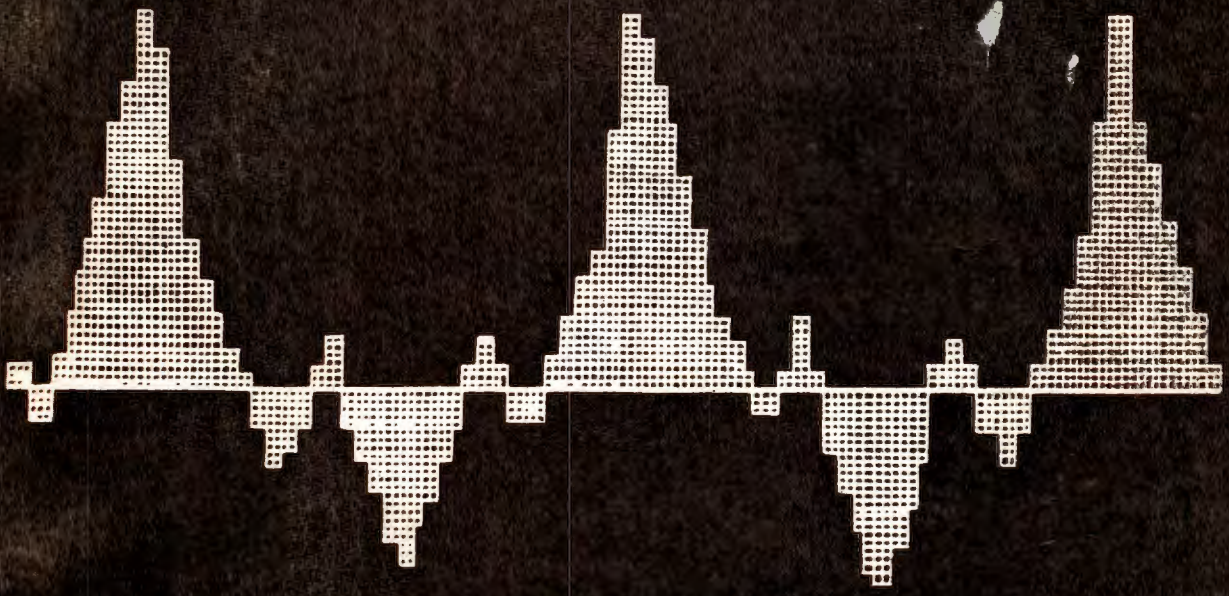
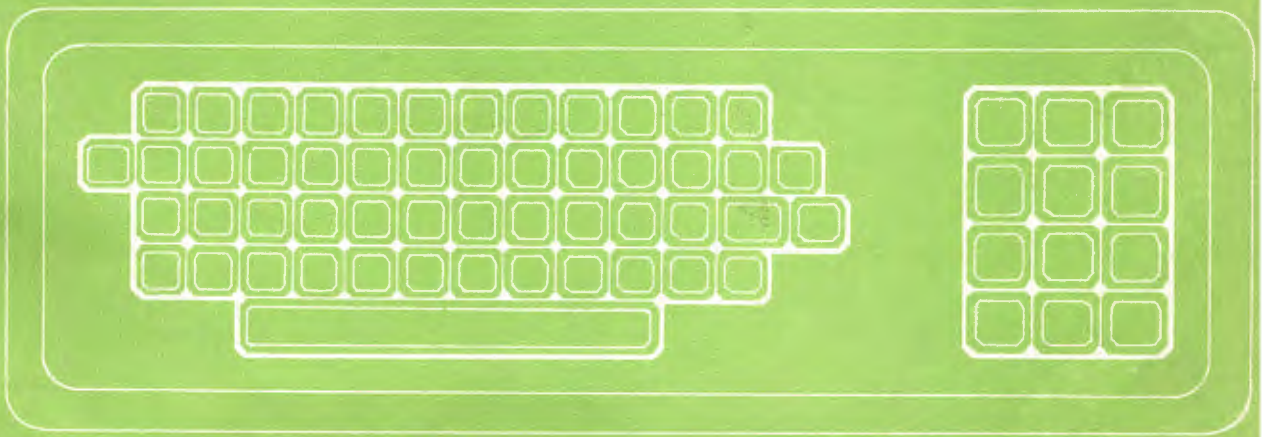


TRS-80 微计算机磁盘操作系统

—NEWDOS/80 2.0

周宝兴　梁祖威　编译

海洋出版社



责任编辑：刘莉蕾

TRS-80 丛　书

- 微计算机实用手册　　定价　6.40 元
- TRS-80微计算机驻机解释程序——LEVEL Ⅱ ROM剖析　　定价　3.70 元
- TRS-80微计算机磁盘操作系统——NEWDOS/80　2.0　　定价　3.50 元

统一书号：13193·0437
定　价：3.50 元

TRS-80微计算机磁盘操作系统—NEWDOS/80 2.0

周宝兴　梁祖威　编译

海洋出版社

1985年·北京

内　容　简　介

本书从对系统的使用入手，首先在前四章中介绍了最新的、功能最强的NEWDOS/80 2.0的系统库命令，特殊的功能以及操作系统子程序调用。接着，在第五章介绍了NEWDOS/80 2.0的系统程序模块和数据结构，它为愿意深入分析本系统的读者提供了必要的系统信息。第六章详细说明了独立于NEWDOS/80系统的一些实用程序。第七章至第十章的内容是有关NEWDOS/80 2.0的磁盘BASIC的内容，它比TRS-DOS BASIC的功能更强。本书末的附录列出了错误信息表、英汉对照词汇表和对NEWDOS/80与TRSDOS兼容性的讨论。

本书读者对象：从事微计算机设计和应用的科技人员和大专院校师生以及使用TRS-80微计算机及软件兼容的同类机的广大用户。

TRS-80 微计算机磁盘操作系统

——NEWDOS/80 2.0

周宝兴　梁祖威　编译

海　洋　出　版　社　出　版　（北京市复兴门外大街）

新华书店北京发行所发行　八九九二〇部队印刷厂印刷

开本：787×1092　1/16 印张：17¼　字数：350千字

1985年5月第一版　　1985年5月第一次印刷

印数：20000册

统一书号：13193·0437　　定价：3.50元

前　言

近年来，我国对微计算机的应用已普及到各行各业中，这对推进我国四个现代化的进程起着很大作用。在国内众多的微计算机中，美国Radio Shark生产的TRS-80微计算机占有较大的比例，加果包括软件上兼容的PS-80和PS-85(日本TEAC与Radio Shark联合生产)和国内组装的上述型号的微计算机，以及国内自己研制生产，在软件上兼容的DJS-043, DPS-80, DPS-85, MDR-Z80, YEE8100以及JSR-80等，则数量相当可观。为了帮助广大的TRS-80微计算机及同类机的用户用好这些微计算机，使它们充分发挥作用，海洋出版社先后出版了《微计算机实用手册》和《TRS-80微计算机驻机解释程序——LEVEL II ROM剖析》。在这些书中，对TRS-80微计算机的硬件、接口，各种语言及TRSDOS磁盘操作系统和驻机BASIC解释程序作了介绍和分析，受到了广大读者的欢迎。

最近几年TRS-80微计算机的磁盘操作系统有了很大的改进，相继出现了NEWDOS 2.1, NEWDOS/80 1.0版本和NEWDOS/80 2.0版本。使得TRS-80机的操作系统的功能越来越强。同时广大TRS-80微计算机的用户随着工作的开展以及小型磁盘驱动器价格的下降，对磁盘操作系统的兴趣也与日俱增，但是对于TRS-80微计算机的这些新出现的功能强大的磁盘操作系统，对于它们的使用以及内部结构等，目前尚无一本比较完整的、实用的参考书，为此，我们根据Apparat公司出版的"NEWDOS/80 FOR THE TRS-80 Model I MICRO COMPUTER"和H. C. 潘宁顿著的"TRS-80 DISK & OTHER MYSTERIES"等书籍编译了这本《TRS-80微计算机磁盘操作系统——NEWDOS/80 2.0》。

本书第一章至第六章由梁祖威编译，第七章至第十章和附录由周宝兴编译。由于编译者水平有限，难免有错误与不当之处，欢迎读者批评指正。

编译者

1984年5月

目　　录

第一章 概 述

几乎所有以磁盘为基础的计算机系统均使用磁盘操作系统（通称为DOS），DOS提供执行磁盘I/O的用户程序与实际的磁盘驱动器及其控制器之间的软件接口。通常这些操作系统还要完成许多其它的功能，如控制用户程序的执行、定位磁盘文件和分配文件空间。总之，DOS能够为用户提供强有力的使用功能和灵活方便的使用环境。本书要介绍的是TRS-80微计算机使用的一种性能比较好的磁盘操作系统NEWDOS/80 2.0。

1.1 APPARAT公司的NEWDOS/80 2.0

NEWDOS/80是TRS-80微计算机上使用的操作系统中的一种，但是它只适用于TRS-80 I型和TRS-80 Ⅲ型这两种机型。

NEWDOS/80 2.0版本可以代替NEWDOS/80 1.0版本（1980年6月发表）和NEWDOS/21（1979年3月发表）。NEWDOS/80 2.0是为TRS-80 I型和TRS-80 Ⅲ型微计算机设计的磁盘操作系统。但每个具体的NEWDOS/80 2.0系统盘片只能适用于这两种型式的TRS-80微计算机中的一种。如果你想在TRS-80 I型和Ⅲ型上均用NEWDOS/80 2.0，就必须为每一种机型购买不同的NEWDOS/80系统盘片。要运行NEWDOS/80 2.0磁盘操作系统，TRS-80微计算机至少要有32K RAM和一个5英寸、单面、35（TRS-80 Ⅲ型为40）磁道的磁盘驱动（0号驱动器）。TRS-80 I型用的NEWDOS/80 2.0录制在一个35磁道、单面、单密度的磁盘片上，而Ⅲ型用的NEWDOS/80 2.0则是录制在一个40磁道、单面、双密度的磁盘片上。当然使用时必须有一个能够管理该磁盘片的磁盘驱动器。

TRS-80 I型和Ⅲ型的NEWDOS/80 2.0与NEWDOS/80 1.0，NEWDOS/21以及TRS-80 I型的TRSDOS 2.3基本上是向下兼容的，但是由于存在不兼容的地方，因此这四种版本的操作系统仍然保持着某些不同的程序和文件。NEWDOS/80 2.0与TRS-80 Ⅲ型机的TRSDOS之间的兼容性不如它与I型的TRSDOS之间的兼容性好。过去的TRSDOS在很长一段时间内稳定下来没有更动，因此APPARAT公司在设计NEWDOS时有意把它同TRSDOS之间的不兼容处降到最低限度。但是后来TRSDOS开始修改了，而NEWDOS也在逐步改进，于是两种系统出现了越来越大的不兼容之处。TRSDOS走的是一条路，NEWDOS/80走的是另一条路。如果这两种操作系统的不兼容性使用户感到受了限制，这也没有办法，因为它们毕竟是两种操作系统，NEWDOS不能也不应该作为TRSDOS的影子而存在。NEWDOS填补了TRS-80 I型的TRSDOS的不足，它有一些功能是TRSDOS中没有的，但TRS-80 Ⅲ型的TRSDOS中的一些性能在NEWDOS/80中却没有。本书的附录B列出了NEWDOS/80 2.0与NEWDOS/80 1.0的某些不兼容处，以及NEWDOS/80 2.0与TRS-80 I型、Ⅲ型的TRSDOS之间的不兼容性。

NEWDOS/80的DOS和磁盘BASIC是根据NEWDOS/21的有关内容全部重新写过的。

对于NEWDOS/80的用户来说，并不需要象使用NEWDOS/21那样预先购买TRSDOS。但是我们仍然推荐用户购买TRSDOS，并希望NEWDOS/80的用户购买TRSDOS使用手册和了解其内容，这是因为使用NEWDOS/80要用到其中某些用户必须了解的知识(请参阅计声编译海洋出版社出版的《微计算机实用手册》中TRSDOS的有关章节)。使用NEWDOS/80 EDTASM程序的用户仍然需要购买Radio Shack的磁带编辑/汇编程序的说明书（请参阅《微计算机实用手册》中编辑/汇编程序的有关章节)。

虽然NEWDOS/80 2.0版本已经进行了比1.0版本更为广泛的错误纠正工作，但仍然难免还有错误，许多程序中的错误还有待于在NEWDOS/80 2.0使用过程中去排除。有关修正错误的一些情况在附录C中讨论。

1.2 系统的一些规定及副本的制作

NEWDOS/80比较复杂，因此当用户首次使用NEWDOS/80时，在使用前应该先用1—2小时学习一下资料。

学过有关资料后，把“写保护片”贴在你的NEWDOS/80 2.0磁盘片的保护缺口处，这里用的磁盘片是5英寸软盘片，贴上“写保护片”以后，在用错了命令时可以防止破坏系统盘片上的内容。然后打开计算机电源，把NEWDOS/80系统盘片放入0号驱动器，并按下复位键。这时应该出现NEWDOS/80的标题，标题出现以后，DOS要询问用户日期和时间。如果需要设定日期和时间，请打入相应的数值。接着显示NEWDOS/80 READY，表示DOS正等待用户打入命令。

使用带磁盘的计算机要养成良好的习惯，即除非要复制系统盘片、或非常小心地进行修补工作（见附录C)决不要把NEWDOS/80的原版盘片装入磁盘驱动器内。修补时正确的作法是：应该首先对版本2的系统盘片副本作修改，并经测试确认可靠后，才能去处理原版系统盘片。务必把原版系统盘片保存在一个安全的地方，不要把它夹在你的NEWDOS/80手册内；平时工作中也不要使用它。

NEWDOS/80的功能很强，现在先简单介绍几个常用的命令，让大家有个初步的印象。

通过键盘打入DOS命令：

```
LIB
```

这时将会显示所有的DOS库命令的清单（在第二章中将介绍这些命令，并附有例子)。

打入DOS命令：

```
DIR, 0, S, I
```

将会把NEWDOS/80 2.0系统盘片上所有文件的目录清单显示出来。这些文件(除NWD82-V2/ILF和NWD82 V2/XLF以外）将在第五章中讨论。

打入DOS命令：

```
SYSTEM, 0
```

NEWDOS/80为用户提供某些系统选用参数，这些选用参数是经DOS库命令SYSTEM来规定的(见2.46节)，而且在每次计算机复位期间使选用参数生效。DOS命令SYSTEM, 0会显示所有的SYSTEM选用参数的状态，你应该仔细地把这些值与给定的说明作比较。你可以决定你的系统所用的SYSTEM参数。如果确有必要，此时你可以改变它们；通常应该等到你有了几个备份的系统盘片以后再对原版盘片的SYSTEM参数作改动。你可以随时修

改系统盘片，但不要忘了去掉盘片的写保护片。

打入 DOS 命令：

PDRIVE, 0

NEWDOS/80 能够根据规定的若干种磁盘驱动器和接口的组合来工作。必须用 DOS库命令 PDRIVE（见 2.37 节）来为系统规定 0—3 号驱动器各自的特征说明，然后在每次计算机复位期间，由 DOS 读这些特征说明。你刚打入的 PDRIVE 命令能够显示出已有的驱动器 特征说明，并附加六个伪驱动器（即并非实际存在的）的特征说明。你可以改变一个或几个驱动器特征说明。如果确有必要，此时你就可以改变它们；但通常应该等到有了几个系统盘片的备份副本后再去改动原版的系统盘片。

现在你应该制作三个或更多一些的 NEWDOS/80 2.0 的系统盘片副本。如果可能的话，应在不改变 SYSTEM 或 PDRIVE 参数的情况下制作与原来一样的备份副本。如果不 可 能，则应把变动控制在最低限度，改完后对计算机进行一次复位。你作这件工作时，应该先仔细研究 2.14,2.37 和 2.46 节的内容。

NEWDOS/80 内没有供制作备份系统盘片的 BACKUP 程序(在 TRSDOS 中 有 BACKUP 程序)，但是可以由 DOS 库命令 COPY 的格式 5 或格式 6 来代替（在 NEWDOS/80 2.0版本中，COPY 命令有六种不同的格式，分别有其适用的范围，详细情况见 2.14 节)。制作系 统盘片时，主盘片既是系统盘片同时又是源盘片，而目的盘片是要存放 NEWDOS/80 系统的新工作副本的盘片。用于制作 NEWDOS/80 2.0 副本的 COPY 命令实例如下：

COPY, 0, 0,, FMT, USD

这个例子适用于单驱动器系统，可用来制作系统盘片的副本，其中主系统盘片和副本盘片有相同的 PDRIVE 特征说明。

COPY, 0, 1,, FMT, USD

这个例子适用于多驱动器系统，也能用来制作系统盘片的副本，其中主系统盘片和副本盘片（装配在 1 号驱动器上）有相同的 PDRIVE 特征说明。

COPY, 0, 0,, FMT, USD, CBF, DPDN=4

这个例子适用于单驱动器系统,其中目的盘片的PDRIVE特征说明与主系统盘片上记载的目的驱动器原来的 PDRIVE 特征说明不同,你必须预先改变主系统盘片上4号驱动器的 PDRIVE特征说明（改变 4 号驱动器说明以后，不要忘记使用 A 选择参数或复位计算机)。

COPY, 0, 1,, FMT, USD, CBF, DPDN=4

这个例子适用于多驱动器系统，复制完成以后， 0 号驱动器中盘片的内容将传送到 1 号驱动器内，并且目的驱动器的 PDRIVE 特征说明与当前 0 号驱动器内主系统盘片上记载的 1 号驱动器的特征说明不同。所以应该预先更改主系统盘片上的 4 号驱动器的 PDRIVE 特征说明。

上述 COPY 命令中的各项选用参数在 2.14 节中有详细 的 介 绍，这 里 就 不 多 加 说 明了。

每个系统盘片都有它自己的一组 SYSTEM 参数和 PDRIVE 特征说明。所以对于每 一 个制作的 NEWDOS/80 2.0 版本的工作副本，在复制完成以后，必须为其工作状态设置一组合适的 SYSTEM 参数和 PDRIVE 特征说明。

NEWDOS/80 的所有者和用户有办法禁止其它人来复制 NEWDOS/80 及其中 的 某 些 程序。详细情况请看 COPY 命令格式 2 和格式 4 （见2.14节)。

1.3 NEWDOS/80 系统盘片的修补

因为 NEWDOS/80 中存在的错误是逐渐被发现的，所以需要不断的对错误进行修正，并对原来不够完善之处补充一些新的内容。故而在准备运行用户程序之前，要对准备使用的 NEWDOS/80 系统盘片内的一些模块进行排错和修补。一般在购买 NEWDOS/80 时都会提供有关修补的资料。有些修补是非用不可的，还有一些修补可由用户根据需要自行选用。应该用哪些非用不可的、对 NEWDOS/80 模块排错的内容来修补所有的 NEWDOS/80 2.0 系统盘片的副本和 NEWDOS/80 2.0 原版系统盘片。但是不要一开始就进行排错和修补，而要到你至少有两个或三个完好的 NEWDOS/80 系统备份盘片以后才去进行这项工作（有关修补的情况，请看本书的附录 C)。

1.4 着手使用 NEWDOS/80

一旦复制够了 NEWDOS/80 2.0 系统的备份副本，进行了排错修补，确定了系统的选用参数和驱动器特征说明之后，NEWDOS/80 就可以开始使用了。

妥善地保管好原版系统盘片，并把刚复制好的一个系统盘片装在 0 号驱动器内；然后按复位键，使新的盘片重新初始化 DOS；接着将在荧光屏上显示出 NEWDOS/80 READY。这时用户可以打入一个命令，它可以是第二章中讨论的 DOS 库命令，也可以是要装入和运行的用户程序名或文件名/扩展符。如果用户程序没有文件名扩展符，系统就认为它的扩展符为 CMD。如打入

```
BASIC
```

就能装入和执行 BASIC/CMD 程序。打入

```
SCRIPSIT/LC
```

就能装入和执行 SCRIPSIT/LC 程序。

如果 DOS 库命令和用户程序需要一些参数，就必须把一个或一个以上的空位或逗号放在命令名字的后面及参数的前面。例如：

```
BASIC, 5, 65000
DIR  1  A
```

本书将给出在 NEWDOS/80 2.0管理下能执行的所有程序的说明，这些说明告诉用户怎样使用这个程序。对于 NEWDOS/80 2.0 内的程序模块，除 BASIC 的说明在第七、八章内介绍以外，其它的一些附加程序的说明均在第六章内。这些附加程序都是独立的程序，严格说来它并不属于 NEWDOS/80 2.0，但是这些附加程序都是极其有用的，希望读者在使用它们之前仔细地读一下第六章的内容。

第二章内有 NEWDOS/80 2.0 的系统库命令介绍，每个要使用 NEWDOS/80 2.0 的读者必须阅读这部分的内容。每个库命令的介绍包括：该命令的功能说明，命令格式、说明及举例。

为了给用户充分利用 NEWDOS/80 2.0 本身的资源提供方便，在第三章中给出了一些原来只供系统调用的子程序入口，使得汇编语言程序员可以直接调用这些 NEWDOS/80 2.0 中的子程序。在第五章中比较详细地介绍了有关 NEWDOS/80 2.0 的系统模块及数据结构，这些内容对于希望剖析及移植 NEWDOS/80 2.0 的读者是很有用处的。

本书是从实用的观点出发，按由浅入深的方法编写的，先是一般概念，接着是一些使用方法及键盘命令，当用户能够熟练使用以后，就可以进一步阅读系统子程序调用，系统特有功能及系统结构方面的内容，以达到灵活、充分应用系统的目的。

从第七章开始是有关磁盘 BASIC 的内容，这部分内容比 TRSDOS 的磁盘 BASIC 有了较大的扩充，对于 BASIC 语言有兴趣的读者可以先阅读一下 TRSDOS 的磁盘 BASIC 使用手册（请参阅计声编译海洋出版社出版的《微计算机实用手册》的第六篇），然后再读本书的有关内容，这样阅读起来困难就会少一些。

第二章 DOS 库 命 令

DOS 库命令的数量在一定程度上可以反映操作系统功能的强弱，如 TRSDOS 2.1 只有十几个 DOS 库命令，NEWDOS/80 1.0 约有 30 多个库命令，而 NEWDOS/80 2.0 已有 47个库命令。并且每个命令中可选用参数的数量也大大的超过了以前的几种操作系统。这一章将逐个介绍 NEWDOS/80 2.0 中可以使用的命令。

2.1 书写规则和一般说明

所有的 DOS 命令都要用 ENTER 结束。在今后的说明中，为了简化就不再写出 ENTER，但是在实际使用中，用户一定要用按下 ENTER 键来结束 DOS 命令。

DOS 命令的字符总数不得超过 80 个（包括用作结束的 ENTER 在内）。

书写时选用参数放在一对方括号内，但在 DOS 命令中使用选用参数时，不必往计算机内打入这组方括号 [ ]。例如：

[,PROT=xxx] [,ASE=yn][,ASC=yn]

可以按下面的格式用键打入计算机

,PROT=READ, ASC=N

在命令中可以使用大写字母 A—Z 和非字母数字字符（见上例）。

小写字母和一些后带（或不带）十进制数字的字用来表示应该遵循的命令格式，用户在使用时应该代入相应的字符或实际数值。请看上例。

在只用一个字符表示的格式中，则该格式中的字（英文小写形式）包括了该值的所有的合法字符，因此很容易记住该命令参数可用的合法字符。例如：如果 ASC=Y 和 ASC=N 是两个仅有的合法的 ASC 值，则表示格式通常写成 ASC=yn。

在 DOS 命令中用逗号的地方，可以用一个或多个空格来代替。

除非另有说明，不带后缀 H 的数值将看作为十进制数，而十六进制数必须带有 H 后缀。例如：4000H 和 16384 表示的是同样的值。

当确定一个磁盘文件时，要用到'filespec'（文件标识符）这个专用名词。文件标识符的格式为：

name 1 [/ext 1] [. password 1] [: dn 1]

必须按照上述次序来设置各项参数。其中：

* name 1（名字 1）是指文件的名称，它由 1—8 个字符组成，第一个字符必须是 A—Z的英文字母，其余可以是 A—Z 英文字母或 0—9 数字。

* /ext 1（扩展符 1）是文件名的扩展部分（如 CMD, BAS, OBJ, CIM, TXT, DOC, COM 等等）。它对文件进行了分类。一个文件不一定有文件扩展符。如果要给出扩展符，必须是 1—3 个字符，并以 A—Z 字母开头，其余部分可以是 A—Z 或 0—9。如果一个文件有了文件名扩展符，则引用该文件的所有文件标识符中都必须包含文件名扩展符，除非为它提

供的是一个缺项文件名扩展符（如/CMD)。

* password 1（通行字1）是由1—8个字符组成的，第一个字符必须是A—Z字母，其余部分可以是A—Z或0—9。通行字1是在文件建立时给定的存取通行字和修改通行字的值。当打开一个已经存在的文件，在进行通行字检查时要用到通行字1的值。如果通行字起作用和文件分配有通行字，则在文件标识符中就需要有通行字。否则就不需要通行字。

* dn1（驱动器编号）是磁盘驱动器的编号，在这个驱动器内有存放该文件的盘片。例如：

MYFILE80/BAS. YOURPW80:0（表示文件在0号驱动器内的盘片上）

MYFILE:3（表示文件在3号驱动器内的盘片上）

YOURFILE. YOURPW（表示文件在缺项驱动器内的盘片上）

NEWDOS/80允许在所有的DOS库命令以及为DOS需要而提供的输入信息中使用英文小写字母。

本书对于每个DOS库命令，给出了命令的关键字（如DIR, LIB, COPY等等）与简要定义。接着，如果命令中有参数，则在给出的命令格式中列出了所有要求的和可供选用的参数。下面就是关于该命令的参数和选用项的说明。最后给出该DOS命令的一些例子。

由于编排印刷上的原因，在本书内有些命令格式要分写成多行来表示，但用户应该把每个命令看作为一行连续的语句。

除非另有说明，DOS库命令是能够在MINI-DOS（即缩小规模以后的磁盘操作系统）管理下执行的（见4.2节)。

NEWDOS/80不同于TRSDOS，不用括号把参数括起来。在NEWDOS/80 1.0版本中，用于操作数的括号对于BREAK, CLECK, DEBUG, DIR, PROT和VERIFY是可以随意选用的，而在2.0版本中就不允许使用括号。

同样，1.0版本允许用ON或OFF去代替DOS命令BREAK, CLOCK, DEBUG和VERIFY中的Y和N在2.0版本中这是不允许的。

2.2 APPEND　把一个文件添加在另一个的后面

1. 命令格式

APPEND, filespec 1, [TO,] filespec 2

2. 说明

(1)这个命令将把filespec 1文件放在filespec 2文件的结束处。filespec 2文件的FPDE（文件主目录登记项）的EOF（文件结束）决定了filespec 1文件将要放置的地方。如果filespec 2文件有显式的EOF字符（如BASIC程序文件或汇编源文件）可能会引起麻烦。

(2)此命令不会影响filespec 1文件的内容。filespec 2文件原来的内容也不会被替换，只是把一个文件添加到另一个文件的后面去。

(3)APPEND命令不能在MINI-DOS管理下执行。

3. 举例

(1)APPEND, XXX:1, YYY/DAT:0

1号驱动器内盘片上的文件XXX的内容附加在0号驱动器上的文件YYY/DAT的后面。

(2)APPEND　AAA　TO　BBB

文件 AAA 的内容附加在文件 BBB 的后面。DOS 搜索当前装配的磁盘寻找这两个文件。

2.3 ATTRIB　规定一个文件的属性

1. 命令格式

ATTRIB, filespec 1 [,INV] [,VIS] [,PROT=xxx] [, ACC=password 1] [, UPD=password 2] [,ASE=e] [,ASC=c] [,UDF=u]

2. 说明

(1)这个命令把属性赋予 filespec 1 文件。至少必须规定一个选用参数。如果在你的系统中通行字起作用（即 SYSTEM 的参数 AA=Y），那末 filespec 1 中必须列出该文件现有的修改通行字（如果有的话）。

(2)ATTRIB 命令中的参数说明

* INV 参数规定该文件具有隐性属性。除非在 DIR 命令中使用 I 选用参数，否则DIR将不会列出该文件的目录。

* VIS 参数用于撤消文件的隐性属性，不论该文件原来是否具有隐性属性。这个参数可以很方便的把隐性文件变成可见文件。

* PROT=xxx 指定文件输入/输出时的存取级别，这是指在系统中允许使用通行字（见 SYSTEM 的选用参数 AA），而且用存取通行字（不是修改通行字）来打开该文件的情况才用 PROT=xxx。

用 xxx 定义的存取级别有下列几种：

① LOCK　级别7。该文件根本不允许用户存取，只有系统覆盖装入程序可以存取它。

② EXEC　级别 6。该文件只能作为一个程序执行时才允许访问。在 BASIC 中要用带有 R 选用参数的 RUN 或 LOAD 命令，并且不允许使用 BREAK 键，因此用户不能中断程序的执行，同时禁止直接语句的执行。

③ READ　级别5。在执行或读取该文件内容时允许访问。

④ WRITE　级别4。执行或读、写该文件时允许访问。

⑤ RENAME（或 NAME）　级别 2。在执行、读、写或对该文件改名时允许存取。

⑥ KILL　级别 1。在执行、读、写、改名或删去该文件时允许存取。

⑦ FULL　级别 0。对该文件允许进行任何操作。

* ACC=password 1(通行字 1)为文件分配 password 1 作为存取通行字。如果 password 1 空缺，则假定它全是空格，否则就必须放 1—8 个字符，由 A—Z 字母开头，其余字符可为 A—Z 或 0—9。通过 ATTRIB 命令的这个参数为文件分配存取通行字，这是允许使用 PROT=xxx 保护措施的唯一方法。此外，只有这时才能使存取通行字的作用不同于修改通行字。使用存取通行字打开的文件要受到保护级别的限制。如果在文件建立时只规定一个通行字，则这个通行字既是存取通行字，又是修改通行字，在文件打开时修改通行字具有较高的优先级。如果通行字有效，则在打开文件时文件标识符中必须指明的通行字不应该是修改通行字，而应该用存取通行字，于是当前的保护级别将被存入FCB（文件控制块），供 DOS 以后读、写、装入等等子程序用。这就限制了用户对文件的使用范围，如只能执行或只能执行、读等等。其次，如果想违背存取级别来存取文件，将会显示出 ILLEGAL ACCESS TRIED TO A PROTECTED FILE（非法存取被保护的文件）错误信息。

* UPD=password 2（通行字2）为文件分配 password 2 当作修改通行字。修改通行字的结构与存取通行字相同。文件中通行字有效，在打开文件时首先要把文件标识符中规定的通行字与此文件的修改通行字进行核对。如果一致，就允许该文件作 FULL（级别0）等级的存取。因此，用修改通行字来存取文件就很方便，不受任何限制。但是对某些专门的文件还是用存取通行字比较妥当，使这些文件可以受到某种保护不致于出差错。

* ASE=e 其中 e 可以是 Y 或 N。这个参数加进去以后，如果 ASE=Y，则 DOS 会为文件自动分配磁盘空间；如果 ASE=N，则 DOS 就不允许进一步为文件分配磁盘空间。当文件建立时，缺项参数是 ASE=Y。

* ASC=c 其中 c 可以是 Y 或 N。这个参数加进去以后，如果 ASC=Y，则在 CLOSE 操作期间，DOS 自动分配超出 EOF 范围的文件的磁盘空间；如果 ASC=N，则没有这种重新分配性能。在建立文件时，缺项参数设置成 ASC=Y。

* UDF=u 其中 u 可以是 Y 或 N。这个参数加进去以后，如果规定 UDF=Y，则把该文件标记为待修改文件；如果规定 UDF=N，则清除修改标记。DOS 系统把一个文件标记为待修改文件，就随时准备为那个文件修改扇区，并能很快找到未标修改标记的文件目录项。

3. 举例

(1)ATTRIB, XXX/CMD:1, UPD=ZXCVB, ACC=NMLKJ, PROT=EXEC

为装配在1号驱动器内盘片上的文件 XXX/CMD 分配修改通行字 ZXCVB，存取通行字 NMLKJ 和保护级别为6级，这就使得这个程序只能执行，而不能对它读或写。因为本例中文件标识符 XXX/CMD 没有给出通行字，所以我们必须假定在 ATTRIB 命令之前不作通行字检查（SYSTEM 的选用参数 AA=N）或者文件没有修改通行字。

(2)ATTRIB YYY/DAT . QZBV INV ASE=N ASC=N UDF=N

这个命令要测试文件 YYY/DAT 是否有一个修改通行字 QZBV，如果确有 QZBV 修改通行字，就要为该文件分配隐性属性，并标记上不允许额外的磁盘空间分配和超出 EOF 范围的磁盘空间重新分配，最后，清除文件修改标记。

2.4 AUTO 规定复位时要自动执行的 DOS 命令

1. 命令格式：

AUTO [,doscmd]

2. 说明：

(1)这个命令允许用户规定一个1—31个字符组成的 DOS 命令，使得在系统复位时该命令能自动执行。这个命令保存在当前系统盘片的GAT 扇区的最后32个字节（见5.6节）。

(2)如果没有规定 doscmd，则在 GAT 扇区就不存放该命令，表示复位或打开电源时没有 AUTO 命令要执行。

(3)如果 SYSTEM 的选用参数 AB=N 和 BC=Y，则就象在复位的同时按下 ENTER 键一样，将忽略存放在 GAT 扇区内的 AUTO 命令，系统进入 DOS READY 状态。

(4)AUTO 命令对于那些系统复位时通常要执行某一固定的命令及链接命令(见2.9节、4.3节 CHAIN 命令和2.19节 DO 命令)的用户特别有用。用设置 SYSTEM 的选用参数AB=Y，或 BC=N 的办法，可强制用户进入这个命令或链接命令，这样计算机的系统操作员就可以限制计算机用户的使用范围。

3. 举例

(1)AUTO BASIC RUN"XXX/BAS"

使计算机复位或打开电源后立即进入 BASIC，并开始执行 BASIC 程序 XXX/BAS。

(2)AUTO DO RSACTION

使计算机复位或打开电源后立即链接文件 RSACTION/JCL，于是就可执行包含在其中的DOS 和其它程序命令。

(3)AUTO

使计算机复位或打开电源后立即进入通常的 DOS READY，等待从键盘输入的下一个 DOS 命令。这个例子完成了撤消盘片上原有 AUTO 命令的功能。注意：执行 AUTO 命令，必须先去掉 5 英寸盘片上的写保护片。

2.5 BASIC2 进入非磁盘的 ROM BASIC（只适用于 TRS-80 Ⅰ型）

1. 命令格式

BASIC2

2. 说明

这个命令不需要任何参数，它使系统进入非磁盘 BASIC，这时，NEWDOS 不再管理计算机系统。

2.6 BLINK 启动或禁止光标闪烁

1. 命令格式

BLINK [,yn]

2. 说明

(1)BLINK 或 BLINK, Y 使得显示光标闪烁。

(2)BLINK, N 使得显示光标不闪烁。

(3)SYSTEM 的选用参数 BH 能用于设置计算机在复位或打开电源后光标是否闪烁。

2.7 BOOT 复位计算机

1. 命令格式：

BOOT

2. 说明：

在 TRS-80 Ⅰ型计算机中，BOOT 命令在驱动器停止转动后执行 Z80 的 HALT 指令，以实现硬件和软件的复位。而在 TRS-80 Ⅲ型计算机中，因为 HALT指令不能产生硬件复位，所以只是转移到 0 单元去实现软件复位。

2.8 BREAK 启动或封锁 BREAK 键

1. 命令格式：

BREAK [,yn]

2. 说明：

(1)BREAK 或 BREAK, Y 使 BREAK 键作为普通的输入键（16 进制代码 01），在下一

次出现 DOS READY 时才恢复其 BREAK 功能，那时它就被置成相应于 SYSTEM 选用参数 AG 的状态。

（2）BREAK,N BREAK 键不能作为普通的输入键用，直到下一次出现 DOS READY时为止，那时它就被置成相应于 SYSTEM 选用参数 AG 的状态。

（3）BREAK 命令对于那些想使 BREAK 键作为普通输入键来用的程序是很有用的，BREAK 键可以经过 DOS-CALL（向量为 4419H）使它起普通输入键的作用。同样，它也适用于那些不想使它成为普通输入键的那些程序。注意：在 DOS READY 状态执行 BREAK 命令是不起作用的，因为立即要返回到 DOS READY，以致把 BREAK 复位到相应的SYSTEM 选用参数 AG 所规定的状态。

（4）在 NEWDOS/80 中也可以把 C9H 存入 TRS-80 Ⅰ型微计算机的 4312 单元（Ⅲ型为 4478H）以启动 BREAK 键成为普通输入键功能，或者把 C3H 存入同样的单元而封锁BREAK 键。在 NEWDOS/80 1.0 版本中，也可以用改变 4369H 单元（仅指 TRS-80 Ⅰ型）的位 4 控制 BREAK 键；在 2.0 版本中，对于 TRS-80 Ⅰ型微计算机来说，置位或清除这一位不起作用，但也不会有什么有害的后果。然而在Ⅲ型中，不能更动这一位，因为现在这个单元是处于系统缓冲区内。

2.9 CHAIN　用磁盘文件代替键盘输入

1. 命令格式：

CHAIN, filespec 1 [,sectionid]

2. 说明：

DOS 命令 DO 能完成与 CHAIN 完全相同的工作。

（1）CHAIN 命令的用途是使预先定义的一组字符作为键盘输入来对待。这组字符事先已经存在文件标识符为 filespec 1的文件内。

如果 NEWDOS/80 尚未处于链接方式中，则此命令可使它置于链接方式。NEWDOS/80 打开 filespec 1 文件，如果发现没有确定 sectionnid（段标记），则将定位于该文件开始的地方。如果确定了段标记，则将搜索该文件，寻找与段标记相一致的记录，并定位于该段的 ID（标识符）记录后面的那个字节。

接着，可以想象为从键盘来的输入是来自于链文件，并一直到遇到一个文件结束或段结束才终止链接，或者由执行 DOS 命令 CHNON, N 暂停链接。

（2）键盘数据是以下列两种方式中的一种从链接文件中获取的。

* 如果 SYSTEM 选用参数 AT=N，则链接工作于记录方式。在这种工作方式中，每当 NEWDOS/80, BASIC 或任何程序需要从键盘经标准 ROM 键盘记录输入子程序(在 05D9H)得到一个新的记录时，这个记录将来自链文件。任何其它需要由键盘输入的皆由键盘提供，而不由链文件提供。

* 如果 SYSTEM 选用参数 AT=Y，则链接工作于字节方式。在这种工作方式中，所有要求经标准键盘输入子程序提供的键盘输入字符均由链文件提供。

（3）CHAIN 命令可以经 DOS-CALL 或 BASIC 的 CMD 发出。这样做时，DOS 不立即返回到调用它的程序，而代之以继续执行从链文件取来的命令，直到遇到下列情况之一时为止：文件结束，段结束，CHNON, N 命令或 CHNON, Y 命令。

(4)在 MINI-DOS 管理下，CHAIN 命令无效。

(5)链接文件的建立者要保证对于系统或用户程序链接是正确的，不在链接中产生不合法的情况。

(6)NEWDOS/80 一次只能有一个链接文件起作用，而不能有多于一个的链文件。如果从当前链接文件得到的新的 DOS 命令本身就是一个 CHAIN 命令或者是一个 DO 命令，则停止当前文件的处理，而去打开这个新的链接文件，于是产生一个新的当前链接文件。

(7)当系统打开一个链接文件时，如果文件标识符中未给出文件名扩展符，则文件名扩展符就认为是 JCL。

在 4.3 节中将要进一步讨论 CHAIN 命令。

3. 举例

(1)CHAIN, XXX:0

链接开始于文件 XXX/JCL:0 的起点。

(2)DO, YYY/CHN:1, QQQ

链接开始于文件 YYY/CHN 中的名为 QQQ 的链接段的第一字节。

2.10 CHNON 改变链接状态

1. 命令格式

CHNON [,ynd]

2. 说明

在链接中要用到 CHNON 命令。如果链接文件没有打开，将会发生错误。CHNON 命令不应该是不分段链接文件中的最后一项，或链接文件段中的最后一项，因为这时该命令是没有意义的。

(1)CHNON,N 记住链接文件中的当前位置并暂停链接工作，接着的键盘字符就来自键盘。如果链接是在 DOS-CALL 下进行的，则会退出当前的 DOS-CALL 级。

(2)CHNON, Y 使接着的键盘字符来自链接文件，开始于该链接文件中的当前位置。如果 CHNON,Y 是作为一个 DOS-CALL 来执行的，则要退出当前的 DOS-CALL 级。

(3)CHNON,D 使接着的键盘字符来自链接文件，开始于该链接文件中的当前位置。如果 CHNON,D 是作为一个 DOS-CALL 来执行的，则不退出 DOS-CALL，仍留在这个级别中，并执行链接文件中的下一个命令，直至遇到下列情况之一时为止：CHNON, Y命令，CHNON,N 命令，段的结束或文件的结束。

在 2.9 节和 4.3 节有链接的进一步讨论。

2.11 CLEAR 清除用户存储器、路径、定时和逻辑排队

1. 命令格式

CLEAR [,START=addr 1] [,END=addr 2] [,MEM=addr 3]

2. 说明

CLEAR 命令有下列功能：

(1)完成 DOS 命令 ROUTE, CLEAR 的功能。

(2)撤消在定时中断子程序链中的所有用户子程序的排队，用户子程序排队是由 4410H

(TRS-80 Ⅰ型) 或 447BH (TRS-80 Ⅲ型) 转入 DOS 的。包括关闭时钟显示。

(3)撤消所有名字带“*”号的子程序排队，由 4461H 转入 DOS。包括撤消 NEWDOS/80 假脱机程序（如果它在起作用），但这不是它的正常的终止。假脱机程序（如果在使用）应该在执行 CLEAR 命令前完全终止。

(4)复位 HIMEM 成 addr 3 (地址 3)，如没有规定 addr 3，则要复位到最高的存储器地址。

(5)把 addr 1 (地址 1) 或 5200H (addr 1 通常比 5200H 大) 到 addr 3 或 HIMEM (addr 3 通常低于此值) 的存储器清零。addr 1 必须大于或等于 5200H 而小于或等于 addr 3。

3. 举例

(1)CLEAR, START=6000H, MEM=0DFFFH

清除所有路径，撤消全部定时和名字带“*”号的子程序队列。HIMEM 设定为 0DFFFH。6000H 到 0DFFFH 的主存储器被清零。

(2)CLEAR

清除所有路径，撤消所有定时和名字带“*”号的子程序队列，HIMEM 设定为最高主存单元，从 5200H 到 HIMEM 的所有存储器单元都被清零。

2.12 CLOCK 每秒显示一次时间

1. 命令格式

CLOCK [,yn]

2. 说明

(1)CLOCK 或 CLOCK, Y 当前的时钟数值显示于荧光屏顶行的第53—60列的位置，每秒显示一次，显示格式为时:分:秒。

(2)CLOCK, N 停止时钟显示。

用户要注意这个时钟有时会停止一段时间，因为系统内没有产生时、分、秒的专用硬件时钟电路。在 TRS-80 Ⅰ型机中，时钟时间的计算是由 25 毫秒中断链来完成的 (在 TRS-80 Ⅲ型中，是由 ROM 中的定时中断来完成的)，不论何时，只要其它来源的中断超过 25 毫秒 (Ⅲ型为 33 或 40 毫秒)，就会损失一个或多个 25 毫秒实时钟中断，而每丢失一个就使时钟少走 25 毫秒 (Ⅲ型则少走 33 或 40 毫秒)。当进行磁盘 I/O 时，这种实时钟中断的丢失就很频繁，而在磁带 I/O 时会丢失得更多，如果其它的子程序使这种 25 毫秒中断链停止超过几毫秒时，也会经常丢失这种 25 毫秒中断，从而影响机内时钟的计时正确性。

2.13 CLS 清除荧光屏显示

1. 命令格式

CLS

2. 说明

CLS 命令可以清除荧光屏上的显示，并把显示恢复成 64 字符的模式。在 TRS-80 Ⅲ型中，保留荧光屏顶行不被清除。

2.14 COPY　复制单个文件、多个文件或整个磁盘

1. 命令格式

COPY 命令有六种格式，列出如下：

(1)COPY, filespec 1 [,TO] filespec 2 [,SPDN=dn 3] [,DPDN=dn4]

(2)COPY, $filespec 1 [,TO] filespec 2 [,SPDN=dn 3] [,DPDN=dn4]

(3)COPY, [:] dn1, filespec 1 [,TO], filespec 2 [,SPDN=dn 3] [,DPDN=dn 4]

(4)COPY, [:] dn1, $filespec 1 [,TO], filespec 2 [,SPDN=dn 3] [,DPDN=dn 4]

(5)COPY, [:] dn 1 [=tc 1] [,TO], [:] dn 2 [=tc 2], mm/dd/yy [,Y] [,N] [,NDMW] [,FMT] [,NFMT] [,SPDN=dn 3] [,DPDN=dn 4] [,SPW=password 1] [,NDPW=password 3] [,DDND] [,ODN=name 1] [,KDN] [,KDD] [NDN =name 2] [,SN=name 3] [,USD] [,BDU] [,UBB]

(6)COPY, [:] dn 1 [,TO], [:] dn 2 [=tc 2], mm/dd/yy, CBF [,Y] [,N] [,USR] [,/ext] [,UPD] [,ILF=filespec 3] [,XLF=filespec 4] [,CFWO] [,NDMW] [,FMT] [,NFMT] [,SPDN=dn 3] [,DPDN=dn 4] [,SPW=password 1] [,ODPW =password2] [,NDPW=password 3] [,DDND] [,ODN=name 1] [,KDN] [,KDD] [,NDN=name 2] [,SN=name 3][,USD] [,UBB] [,DDSL=ln1] [,DDGA=gc 1]

2. 说明

NEWDOS/80 2.0 版本已对 COPY 命令作了很大的改动，所有用户（包括新、老用户）都应该仔细地读这一节的内容。这里提供了 6 种格式的 COPY 命令，其中格式 5 及格式 6 是用于复制备份盘片的，格式 1 和格式 2 用于多驱动器工作环境，而格式 3 和格式 4 用于单驱动器工作环境。

COPY 命令不能在 MINI-DOS 下执行；但是复制单个文件时，DOS 库命令MDCOPY是可以用的。

(1)下面介绍 COPY 命令中的一些参数。

* dn1 和 dn2 是驱动器的编号，两者可以是相同的。dn1 和 dn2 前面的冒号是选用的。

* filespec 1 是源文件的文件标识符。filespec 2 是目的文件的文件标识符。前面带有 $ 号的文件标识符表示源文件或目的文件，或者两者都在 0 号驱动器上，而且盘片上或许是：

① 包含的 NEWDOS/80 系统与 COPY 开始时 0 号驱动器内盘片上的系统不相同。

② 不包含 NEWDOS/80 系统。

③ 根本就不包含有任何操作系统。

在处理格式 2,3,4,5 和 6 时，系统可能要求装配各种盘片，这时请按照系统的提示去做!!

① 当提示 "SYSTEM DISKETTE"（系统盘片）时，请装配开始执行 COPY 命令时在 0 号驱动器上的那个 NEWDOS/80盘片。

② 当提示 "SOURCE DISKETTE"（源盘片）时，请装配已包含有 filespec 1（格式 1,2, 3和 4）或要复制的数据（格式 5 和 6）的盘片。

③ 当提示 "DISTINATION DISKETTE"（目的盘片）时，要装配待装的 filespec 2（格式 1,2,3 和 4）或待接收复制数据（格式 5 和 6）的盘片。

* SPDN=dn 3 源驱动器的编号。SPDN=dn 3 告诉系统在所有的源驱动器 I/O 中，用系统盘片上驱动器 dn3 的 PDRIVE 特征说明（见 2.37 节 DOS 命令 PDRIVE）来代替源驱动器原来的 PDRIVE 特征说明。dn3 是一个 0—9 的值，可以参看由 PDRIVE 命令列出的驱动器编号。

* DPDN=dn4 目的驱动器的编号。DPDN=dn4告诉系统在所有的目的驱动器 I/O 中，用系统盘片上驱动器 dn4 的 PDRIVE 特征说明来代替目的驱动器原来的 PDRIVE 特征说明。dn4 是一个 0—9 的值，可以参看由 PDRIVE 命令列出的驱动器编号。

注意：对于 0 号驱动器，单驱动器 COPY（格式 4,5 或 6）使用 SPDN 和 DPDN 参数意味着在 COPY 期间，即使只用了一个驱动器，也会有三个不同的 PDRIVE 特征说明（一个用于系统盘片，一个用于源盘片，另一个用于目的盘片）。

（2）**格式1** 是复制单个文件用的。它是用于把 filespec 1 的文件内容复制成 filespec 2的文件。COPY 命令中所涉及到的那些盘片在 COPY 命令执行以前必须早已装配好，系统不给出装配提示信息。COPY 命令执行以后，filespec 1 的文件内容不会改变。filespec 2 的文件即使原来有内容也将会被破坏。如果 filespec 2 书写格式的前面部分与 filespec 1 的前面部分相同，则 filespec 2 可以省略前面的相同部分，但 filespec 2 的剩余部分要用 / 或 · 或 : 来开始。例如：

COPY, USERFILE/DAT:0, TO, USERFILE/DAT:1 能缩写成

COPY, USERFILE/DAT:0, TO, :1

要记住：TO 是选用的，并且其中的逗号可用空格来代替。于是上述命令也可以写成

COPY USERFILE/DAT:0 :1

（3）**格式2** 与格式 1 相同，仅是在 filespec 1 前带有 $ 号以表示在 0 号驱动器（即系统盘驱动器）上放置盘片时会发生抵触，但 DOS 会向用户提示应该把哪一个盘片装配在 0 号驱动器上。如果源和目的驱动器编号两者均是 0（即只用一个驱动器），而源和目的文件又是分开放在两个盘片上的，这时要用格式 4 来代替格式 2。

（4）**格式3** 与格式 1 相似，适用于用户只有一个驱动器可用来复制，而 filespec 1 的文件是放在与 filespec 2 的文件不同的盘片上。这两个文件标识符都不能规定驱动器编号。DOS 将会对装配盘片作提示，用户应该按照 DOS 的要求装配源和目的盘片。如果指定 0 号驱动器，则源和目的这两个盘片都必须包含有 NEWDOS/80 系统，而且这个 NEWDOS/80 必须与开始执行 COPY 命令时装在 0 号驱动器上的那个一样，否则请用格式 4。

（5）**格式4** 能够完成类似格式 3 的功能，只是允许一个或两个文件是在具有不同的 NEWDOS/80 系统的盘片上，或在非 NEWDOS/80 系统盘片上，也可以在根本没有任何操作系统的盘片上。DOS 将会对系统盘片、源盘片和目的盘片的装配作出提示。格式 4 只适用于用户只有一个驱动器(dn1=0)时使用，如有多个驱动器，就没有必要把时间白白地浪费在不必要的反复装配盘片的工作上。

格式 2 和格式 4 允许程序的提供者（不论是免费还是收费）把他们的软件产品放在一个不带 NEWDOS 系统的盘片上。除提供者的程序和数据外，盘片上只需要包含目录和 BOOT/SYS 文件，这两个文件是在格式化期间建立在每个盘片上的。

NEWDOS/80 中没有制作备份盘片的程序（如 TRSDOS 中的 BACKUP 程序），而用 COPY 命令的格式 5 或 6 来代替。格式 5 是整盘的扇区对扇区的复制，不涉及到文件的数

目和类型。格式 6 是把源盘片上的一部分或全部的文件复制到目的盘片上。在传送相同数目的数据时，这两种格式相比，格式 5 比较快，而格式 6 则允许源盘片和目的盘片型号之间有比较大的差别，并力求把文件分配在连续的空间。

（6）**格式5** 是全盘复制。两个驱动器的缺项说明是 DOS 目前正在使用的 PDRIVE 特征说明。用于复制工作的驱动器必须每个磁道有相同的扇区数，每个 gr. 组有相同的 gr. 数，并且每个 gr. 有相同的扇区数（目前 NEWDOS/80 的标准是每个 gr. 有 5 个扇区），否则必须用格式 6。目的盘片的磁道可以比源盘片多，如果是这样，就要调整目的盘片的目录，以便记入这些额外的空闲 gr.（如果规定了 BDU 选用参数，就不必这样做）。格式 5 的选用参数定义如下：

＊ =tc1 复制期间 DOS 用 tc1 值作为源盘片的磁道计数，而不用源驱动器的缺项值。

＊ =tc2 复制期间 DOS 用 tc2 值作为目的盘片的磁道计数，而不用目的驱动器的缺项值。

＊ mm/dd/yy 是放置在目的盘片日期字段中的日期。不给定 mm/dd/yy 时，就用系统盘上的日期。只有当命令只给出两个驱动器号的参数（例如 COPY 0 1）时，格式 5 的COPY 命令中才能把 mm/dd/yy 全部省略掉。否则，即使没有 mm/dd/yy 或者用 KDD 或 USD 这两个参数中的一个来代替，也必须把 mm/dd/yy 作为第三个参数放入 COPY 命令中。如果空缺 mm/dd/yy，则必须用分隔逗号来表示（中间不留空格），例如：

COPY, 0,1,, FMT, CBF

＊ Y 用户不管目的盘片上原来有什么内容。Y 不能与 N, ODN, ODPW, DDND, KDN 或 KDD 等参数同时使用。如果没有指定上述与 Y 不能同时存在的参数，则 Y 就是 COPY 命令的缺项参数。

＊ N 在 COPY 或 FORMAT 命令开始时，目的盘片上必须不包含有可以识别的数据，也就是说应该处于全部擦除干净的状态。如果发现目的盘片上有数据，将会终止 COPY命令的执行。N 不能与 Y, ODN, ODPW, DDND, KDN 或 KDD 等参数同时存在。

＊ NDMW 不等待磁盘装配。一旦用户设定这个参数以后，DOS 就认为所有需要的盘片已经装配在指定的驱动器上了，不再显示出装配盘片的提示信息及错误信息。如果有错误发生（不设定 NDMW 参数时会给出错误提示信息），将会终止复制工作。如果确定了NDMW 参数，但既不指定 FMT 选用参数，也不指定 NFMT 参数，则 DOS 就认为是 FMT 参数有效。NDMW 参数用于下列情况比较合适：COPY（或 FORMAT）命令是经 DOS-CALL（也就是说从 BASIC）引用，并且调用的程序不希望用户干预的情况。因为 NDMW 参数使COPY 或 FORMAT 命令不作错误提示和磁盘装配的询问，所以建议在用户用键打入COPY（或 FORMAT）命令时（即人干预的复制工作）一般不要选用 NDMW 参数。

＊ FMT 格式化操作。DOS 在复制数据之前要对目的盘片格式化。FMT 不能与NFMT 选用参数同时用。如果既不给定 FMT，也不给定 NFMT，同时又不指定 NDMW，则 DOS 要问用户 FORMAT DISKETTE? (Y OR N)（要格式化磁盘吗？请用 Y 或 N 回答）。如果不指定 FMT 或 NFMT，但是指定了 NDMW 参数，则 DOS 就认为是 FMT 有效。

＊ NFMT 不进行格式化。DOS 在复制数据之前不对目的盘片格式化。用户必须保证已经对目的盘片作了合乎要求的格式化处理。NFMT 与 FMT 参数不能同时使用。

＊ SPW=password 1 源通行字。如果系统中已使通行字有效（SYSTEM 选用参数 AA

=Y)和 SYSTEM 选用参数 AR=N, 则 COPY 命令需要一个与之匹配的源盘片主通行字。如果password 1 与源盘片的通行字不一致，将会终止复制功能。

* NDPW=password 3 新目的盘片的通行字。password 3 必须符合通行字规则，并分配作为目的盘片的新通行字。NDPW 不能与 BDU 参数同时使用。

* DDND 显示目的盘片的老的名字和日期。目的盘片的老名字和日期在荧光屏上显示出来，使得用户决定是否要在这个盘片上复制新的内容。DDND 不能与 Y, N 和 NDMW等参数同时使用。

* ODN=name 1 老的目的盘片的名字。如果目的盘片的老名字不是 name 1，则系统要询问用户是否要进行复制。ODN 不能与 Y, N 和 NDMW 等参数同时使用。

* KDN 保持目的盘片的名字。目的盘片保持它原来的老名字，而不接受源盘片的名字。KDN 不能与 Y, N, BDU 和 NDN 等参数同时用。

* KDD 保持目的盘片的日期。目的盘片保持它原来的老的日期，而不接受 COPY 命令中给出的 mm/dd/yy 参数。KDD 不能与 Y, N, BDU 和 USD 等参数同时使用。

* NDN=name 2 新的目的盘片的名字。目的盘片取 name 2 为它的名字，而忽略接收到的源盘片的名字。name 2 必须符合盘片名字的规则。NDN 不能与 BDU 和 KDN 同时使用。

* USD 采用源盘片的日期。目的盘片把源盘片的日期作为它的日期，而忽略接收到的由 COPY 命令发出的 mm/dd/yy 参数。USD 不能与 KDD 和 BDU 参数同时使用。

* SN=name 3 源盘片的名字。如果源盘片的名字不是 name 3，DOS 就要给出提示信息，用户可以决定是否进行复制。

* BDU 不对目的盘片的目录进行修改。除了原封不动地把源盘扇区的内容复制到目的盘上以外，格式 5 的 COPY 命令还将修改目的文件 BOOT/SYS 中的引导和 PDRIVE 数据，如果有必要，还可以把名字、日期、通行字和额外的 gr. 信息等放进文件 DIR/SYS 中。然而有时不想对这个文件进行修改，这时就要指定选用参数 BDU，以便不作这种修改。BDU 参数通常用于：源盘片的目录有问题，源盘片有一个非标准化的目录（如 TRS-80 Ⅲ型 TR-SDOS 的目录），或者用户想不作任何修改地全盘片复制，但是源盘片根本就没有目录，这时就要用 BDU 参数，以便不对目的盘片的目录进行修改。BDU 不能与 KDN, NDN, NDPW 和 USD 等参数同时使用。

* UBB 使用大的缓冲区。在 NEWDOS/21 和 NEWDOS/80 1.0 版本中，除了单驱动器多个盘片的复制以外，都限定 COPY 命令使用 7000H 以下的所有内存储器，而在单驱动器多盘片复制的情况下，直到 HIMEM 的所有内存储器都要用上。如果在复制时用户想要强制使用 HIMEM 以下的所有内存储器，则要指定 UBB 参数。然而在 NEWDOS/80 2.0 版本中，直到 HIMEM 的全部存储器都要用到，除非是由 DOS-CALL 引用的 COPY 命令（也就是由 BASIC 引用的 COPY 命令），在这时只用 7000H 以下的内存储器。所以，在 NEWDOS/80 2.0 中，UBB 是一个没有作用的参数，现在保留它只是为了与版本 1.0 兼容罢了。

（7）**格式6** 用于多文件的复制，它与格式 5 的差别仅仅是多包含一个 CBF（按文件复制）的选用参数。虽然格式 5 是一种制作备份盘片比较快的手段，但是格式 6 提供了更大的灵活性：允许在特性有很大差别的盘片和驱动器之间进行文件的复制。待复制文件的选择可由选用参数 USR, /ext, UPD, ILF, XLF 和 CFWO 的组合来限定，如果确定了一个或多个

准则，则只有符合全部准则的文件才能被复制。除了 BDU 以外，格式 5 的选用参数以及下面这些附加的选用参数都可以用于格式 6 中。

* 如果指定了 NFMT 参数，则 Y, N, KDN, KDD, NDN, USD, NDPW, DDSL, DDGA 或 tc2 中任何一个不能再选用了（可以用 OPDW 参数），另外，除非系统文件已经存在目的文件目录中，否则系统文件不能被复制。

* 如果不指定 NFMT 参数，则就象用 FORMAT 命令一样，把目的盘片格式化，包括建立 BOOT/SYS 和 DIR/SYS 文件。然后，在文件复制以前，把所有待复制的文件登入目的盘片的目录内。必须这样做的原因是系统文件必须占有相同的目录文件主目录登记项，以使 DOS 能正常工作。

* CBF 按文件复制。CBF（格式 6 需要这个参数，且只用于格式 6）表示复制按文件进行，而不是按盘片扇区的次序顺序进行。在把一个 NEWDOS/80 2.0 系统复制到另一个盘片上去的时候要注意新盘片上是否有 BOOT/SYS 和 DIR/SYS（即该盘片是否可引导），如果该盘片上有这两个文件，则使用 CBF 选用参数时不必用 FMT 参数，否则一定要用 FMT 参数，以便建立 BOOT/SYS 和 DIR/SYS 文件。

* USR 复制用户文件。只复制用户的文件，拒绝复制系统的和隐性的文件。

* /ext 复制有文件名扩展符 ext 的那些文件。使用这个参数以后，只复制带有文件名扩展符 ext 的文件。ext 是一个由 0—3 个字符组成的文件名扩展符。例如 /CMD, /BAS 等。

* UPD 复制要修改的文件。只复制在源盘片目录上修改标志打开的那些文件。这个标志是由标准的 DOS 扇区写子程序打开的，表示从上一次清除修改标志以来至少有一个扇区已经写入和重写到这个文件，这个标志要通过 PROT 或 ATTRIB 命令来清除，它并不由 COPY 命令来清除。因为标准的 DOS 扇区写子程序是把文件的扇区写到目的盘片上，所以，对复制的目的文件来说，修改标志是打开的。

* ILF=filespec 3 包括待复制文件的清单文件。filespec 3 确定了一个包含有待复制文件清单的文件。如果一个文件不在这个清单中，则它不被复制。如果一个包含在清单内的文件不在源盘片上，这不算一个错误。在这个清单中，每个待复制的文件由文件名/扩展符后跟 EOL 字符(0DH) 来表示。如果一个说明由一个分号（;）开始，它就被看作一个解释。每个说明（除解释以外）限定最多是 13 个字符（包括 EOL 在内）。在读出时，文件的字节要变换成 0—127 范围内的 ASCII 码。文件能用 SCRIPSIT 程序来加工，但是用户要保证，除 00H 以外没有别的字符跟在最后一个 EOL 字符的后面。SCRIPSIT 程序容易留下无关的附加字符，所以应该删除文本的结束字符 EOL 后面的部分。ILF 不能与 XLF 参数同时使用。

* XLF=filespec 4 记有不要复制的文件的清单文件。filespec 4 与上面 ILF 有相同的结构，它确定了不要复制的文件。如果一个不要复制的文件不在源盘片上，这不算是一个错误。XLF 与 ILF 参数不能同时使用。

* CFWO 必须经用户核对后才复制的文件。对于这种要核对的文件，DOS 要询问用户（每次一个文件）该文件是否要复制到目的盘片上。如果该文件要复制，则回答 Y，如果不要复制，则回答 N，如要重新开始整个 CFWO 询问过程，请回答 R，如果不再有文件要复制，就回答 Q。一直到询问完毕才开始复制文件。

* ODPW=password 2 老的目的盘片通行字。如果指定了 NFMT 参数，并且通行字有效和 SYSTEM 选用参数 AR=N，那就需要一个与之一致的目的盘片通行字。如果password 2

与目的盘片的通行字不一致，则终止复制。

＊ DDSL=ln 1　目的盘片目录开始的 gr. 组 (lump)编号。如果指定了 DDSL 参数，格式化将从 gr. 组 ln 1 的第一扇区上的目录开始，否则将使用驱动器的缺项开始 gr. 组的编号(见 PDRIVE 命令)。DDSL 不能与 NFMT 参数同时使用。

＊ DDGA=gc 1　目的盘片的目录 gr.分配。如果指定了 DDGA 参数，则格式化将分配 gc 1（取值在 2—6 范围）个 gr. 给目录，否则将把驱动器的缺项 gr. 数分配给目录。DDGA 不能与 NFMT 参数同时使用。

如果在 COPY 格式 6 工作期间，目的盘片有的空间不足以放一个文件，于是就会在荧光屏上显示出 DISKETTE FULL=name/ext （磁盘已满——文件名/扩展符），同时目的文件的 EOF 被置成 0。虽然 EOF 置成了 0，但是已分配给文件的任何空间不能再重新分配了。

在 DOS-CALL（即由 BASIC 调用）的情况下，不能执行单驱动器的格式 5 或格式 6 的 COPY 命令，因为这时复制被限制在内存储器低于 7000H 的部分进行，这就需要频繁地反复倒换盘片，这是极不方便的。

(8)如果不指定 NDMW 参数，在 COPY 命令和 FORMAT 命令执行期间，任何时刻按⊡键将使命令暂停执行，等到按下 ENTER 键时再继续下去。而按下⊡键将会取消这个命令。在任何时刻按⊡键会终止这个命令。但是要注意，这样处理以后，目的盘片的状态将是不可知的，特别在格式化期间突然取消命令，其后果就更难预料。

(9)对于标准的 40 磁道，双密度，单面 5 英寸 TRS-80 Ⅲ型的 TRSDOS 盘片来说，可以用 COPY 命令把Ⅲ型的 TRSDOS 的磁盘文件传送到（或取自）NEWDOS/80 系统。但是这个操作有下列几点限制：

① NEWDOS/80 不能格式化一个 TRS-80 Ⅲ型的 TRSDOS 盘片，然而，用户只要有一个已格式化的空白Ⅲ型 TRSDOS 盘片，就能够在 NEWDOS/80 管理下用带有参数 NFMT 和 BDU 参数的格式 5 COPY 命令，反复地复制出另外一些格式化盘片，这样就可以得到格式化的空白的Ⅲ型 TRSDOS 盘片。

② 用户必须保证源和（或）目的盘片（Ⅲ型的 TRSDOS）放在具有合适的 PDRIVE特征说明的驱动器内，该特征说明可以是隐含的也可以是用 SPDN 和（或）DPDN 参数直接给出的（见 PDRIVE 命令的例 3 和 2.37 节关于 PDRIVE 命令的详细说明)。

③ 要复制的文件不需要预先存在于Ⅲ型 TRSDOS 盘片上。NEWDOS/80将会分配给合适的目录项和磁盘空间。

④ COPY 命令的格式 1,2,3,4 或 6 中的任何一个均可用于复制文件到（或来自）Ⅲ型 TRSDOS 盘片上。要记住，必须不指定 FMT 参数。如果要用格式 6，并且源或目的盘片之一是一个Ⅲ型 TRSDOS 盘片，则系统文件（系统文件的标记为 FPDE 的第一个字节的位6＝1）是不能被复制的。

⑤ 在 NEWDOS/80 和Ⅲ型 TRSDOS 之间互相复制的那些文件总是可读的，然而在你的系统中它们并不一定是可以使用的。

3. 举例

(1)COPY　XXX:1　YYY:1

在这个格式 1 COPY 命令中，已装配在 1 号驱动器内盘片上的文件 XXX 被复制为同一盘片上的文件 YYY。

(2)COPY, AAA, BBB:2

在这个格式1 COPY 命令中，在目前已装配好的盘片上搜索文件AAA，如果找到，则把它复制到已装配在2号驱动器内的盘片上，文件名定为BBB。

(3)COPY SUPERZAP/CMD:0 :3

在这个格式1 COPY命令中，把文件名为SUPERZAP/CMD的文件从已装配在0号驱动器的盘片上复制到已装配在3号驱动器的盘片上。因为两个文件的文件名和文件名扩展符是相同的，所以第二个文件标识符可以省略。

(4)COPY XXX:1 2 SPDN=9

在这个格式1 COPY命令中，SPDN=9表示只有在COPY执行期间，所有源文件输入/输出都假定1号驱动器具有PDRIVE特征说明当中给9号驱动器所规定的特性。如果我们假定PDRIVE特征说明中规定9号驱动器符合TRS-80 Ⅲ型的TRSDOS盘片的要求(见2.37节PDRIVE例3)，则此COPY命令将把已装配在1号驱动器上的Ⅲ型TRSDOS盘片上的文件XXX复制到已装配在2号驱动器上的NEWDOS/80盘片上。

(5)COPY, $XXX:1, YYY:0

在这个格式2 COPY命令中，包含文件YYY的目的盘片并不是COPY命令开始时装配在0号驱动器上的同一盘片。DOS将根据它的要求向用户提示是装配目的盘片还是系统盘片。

(6)COPY, $XXX:0, YYY:1

在这个格式2 COPY命令中，现在放有源文件XXX的源盘片不是COPY命令开始时装配在0号驱动器上的同一个盘片。DOS将根据它的要求向用户提示是装配源盘片还是系统盘片。

(7)COPY 1 XXX YYY/DAT

在这个格式3 COPY命令中，虽然源和目的盘片都是用的1号驱动器，但是包含有文件XXX的源盘片与包含有文件YYY的目的盘片不是同一个盘片。DOS将根据它的需要要求用户装入源或目的盘片。要注意：按照格式3和格式4的要求，这两个文件标识符均不要包含驱动器编号。

(8)COPY 0 XXX/DAT /DAT

在这个格式3 COPY命令中，在一个盘片上的文件XXX/DAT要以XXX/DAT同样的文件名复制到另一个盘片上。两个盘片都要装配在0号驱动器上，DOS将按它的需要要求用户装配它们。因为使用0号驱动器，而且这是格式3而不是格式4，故而源和目的盘片这两者都必须包含有当COPY命令开始时装配在0号驱动器上的同样的NEWDOS/80系统。

(9)COPY 0 $XXX/DAT /DAT

这个格式4 COPY命令实际上完成上面例子同样的工作，其差别仅在于不论是源盘片还是目的盘片均不包含NEWDOS/80系统，于是DOS将按其需要，要求用户装配系统，源和目的盘片。

(10)COPY 0. $XXX XXX SPDN=9

这个格式4 COPY命令与例4完成同样的工作，其区别是现在只用0号驱动器。在这个COPY命令期间，0号驱动器使用两组PDRIVE特征说明。标准的0号驱动器说明是供系统和目的盘片的I/O用的，系统盘片的PDRIVE特征说明中的9号驱动器说明是源盘片I/O

用的。注意，在这个例子中，第二个文件标识符没有缩短，因为没有东西可以省略。

(11)COPY 0 1 06/01/84 FMT

这个格式5 COPY命令是一个最简单和最常用的整盘复制的例子。这个COPY命令把一个盘片上的内容复制到另外一个盘片上，用0号驱动器作为源驱动器而1号驱动器作为目的驱动器。有关驱动器的缺项磁道计数被用作为磁盘的磁道计数。除了可以有不同的磁道数（目的盘片的磁道数必须大于或等于源盘片磁道数）外，两个驱动器要有相同的特性。如果发生错误，则磁盘装配和错误选择项会提示给用户。缺项参数是Y有效，表示用户不必去管目的盘片上以前是否有数据。整个源盘片复制到目的盘片上以前，目的盘片要格式化，并接受源盘片的名字和通行字。目的盘片的日期将设定为06/01/84。如果目的盘片的磁道比源盘片多，它们也将被格式化和相应注册于目录中，这样，目的盘片就可以使用了。

(12)COPY, 0, 1,, NFMT

这个格式5 COPY命令是最简单和最常用的整盘复制的另一个例子。此例与上面那个例子之间的唯一差别是

假定目的盘片已完成格式化，当前系统的日期将成为目的盘片的日期。

(13)COPY, 0, 0, 06/01/84, NFMT, USD, KDN, ODN=WATCHDOG, SN=GOODDATA

这个格式5 COPY命令与前面的例子有些相同，差别之处为：

* 这是单驱动器两个盘片的复制。
* 如果源盘片没有指定名字，将会给出提示信息。
* 如果目的盘片没有给出名字，将会给出提示信息。
* 目的盘片将保留它的老名字。
* 目的盘片将从源盘片上接收它的日期。

单驱动器对两盘片进行复制，要比两个驱动器复制给出更多的装配提示信息。又因为单驱动器复制要进行多次的磁盘装配，所以不能由DOS-CALL（即由BASIC调用）来执行。

(14)COPY, 0, 1,, FMT, CBF

这个格式6 COPY命令是一个最简单和最常用的多文件复制的例子。目的盘片（要装配在1号驱动器上）要格式化，并从源盘片（装配在0号驱动器上）取得它的名字和通行字，以系统日期为它的日期。接着，把所有的源盘片文件（除BOOT/SYS和DIR/SYS）复制到目的盘片上。

(15)COPY, 0, 1,, NFMT, CBF

这个格式6 COPY命令是最简单和最常用的多文件复制形式的又一个例子。它和例(14)的不同之处是：

* 目的盘片不格式化。
* 它的名字，通行字和日期不被改变。
* 在目的盘片上原来不存在的源盘片系统文件（除BOOT/SYS和DIR/SYS)不复制。

(16)COPY, 0, 1,, NFMT, CBF, USR

这个格式6 COPY命令是类似于上面的例子，只是不复制系统文件和隐性文件。

(17)COPY, 0, 1,, NFMT, CBF, UPD

这个格式6 COPY命令类似于上面的例子，只是要复制的源文件是那些标有修改标志的

文件，不复制系统的或隐性的文件。在这种方式中，现在要复制的文件只是从上一次复制以来所改变过的文件。注意，COPY 命令不清除源盘片上的修改标志，只有 DOS 命令 PROT 或 ATTRIB 才能清除这个标志。

(18)COPY, 2, 3=60, 06/01/80, FMT, NDMW, CBF, DDSL=29, DDGA=4

在这个格式 6 COPY 命令执行期间，不显示盘片装配提示信息和错误选择项，系统认为盘片已经按规定装配好了。把目的盘片格式化为 60 磁道。目录将起始于第 29 gr. 组，并分配 4 个 gr.。源盘片上所有的文件（除 BOOT/SYS 和 DIR/SYS ）将被复制到目的盘片上。

(19)COPY, 2,3,06/01/80, CBF, CFWO, NFMT

对于这种格式 6 COPY 命令，目的盘片假定已经格式化了，并且可能包含有文件。对于每个源盘文件（除 BOOT/SYS 和 DIR/SYS)将要求用户认可该文件是否复制到目的盘片上。当所有的提问均已回答后，就要复制选中的文件，但不复制那些预先并不存在于目的盘片上的系统文件。如果某个文件已经在目的盘片上了，则命令执行以后，该文件原有内容要被冲掉。

2.15 CREATE　预先分配磁盘文件

1. 命令格式

CREATE, filespec 1 [,LRL=ln 1] [,REC=count 1] [,ASE=yn] [,ASC=yn]

2. 说明

(1)CREATE 命令允许用户建立一个文件和随意地把确定数目的空记录写到该文件内，因为只有在磁盘上安排有空间，才有可能为文件分配邻接的空间。

有时候一个用户程序希望调用一个或多个已经存在的文件（即使该文件中可能没有可用的数据），所以在该程序首次使用以前，用户必须先建立这些文件。如果一个文件的内容散布在整个盘片上，这时程序的效率就会降低，为了使效率不下降，用户应该预先分配需要的文件空间，以减少这种分散性。

(2)CREATE 命令可以用于建立新的文件或改变已经存在的文件的状态。CREATE 命令中的选用参数有下列几项：

* LRL=ln 1　规定该文件每个记录的长度（用字节表示）。ln1 必须是 1—256之间的一个值，缺项值是 256。

* REC=count 1　规定预先分配给一个文件的记录数。

* ASE=yn　这个参数表示在 CREATE 命令以后，DOS 是否可以按需要自动分配更多的磁盘空间给这个文件。ASE=Y, 表示可以自动分配；ASE=N, 表示不可以。如果命令中没有指定这个参数，就认为是 ASE=Y。

* ASC=yn　这个参数表示是否允许 DOS close 功能自动重新分配超出EOF范周的磁盘空间。ASC=Y,表示允许重新分配；ASC=N,不允许重新分配。缺项值是 ASC=Y。

如果为一个文件分配了足够的磁盘空间，假如分配了 count 1 个记录，每个记录的长度为 ln 1，接着就要把 count 1 个全零的记录写到该文件内，并在这些记录的结束处建立该文件的 EOF。如果规定了 ASE=N, 就禁止文件的磁盘空间作进一步的分配，如果 ASC=N, 就禁止文件自动的重新分配超出 EOF 的磁盘空间。

3. 举例

(1)CREATE, XXX:1, LRL=30, REC=2000

如果XXX文件不存在，就要在驱动器1内的盘片上建立这个文件。记录长度是30，2000个全是00H字节的记录被写入该文件。把EOF放置在60000处。以后还允许DOS为该文件自动分配空间和重新分配空间。

(2)CREATE, YYY:2, 200, ASE=N, ASC=N

如果YYY文件不存在，则把它建立在2号驱动器内的盘片上，记录长度是256，200个全部是00H字节的记录被写入该文件。把EOF放置在51200处。以后不允许DOS为该文件自动分配空间和重新分配空间。

(3)CREATE, ZZZ:0

如果文件ZZZ不存在，则把它建立在0号驱动器内的盘片上。记录长度是256，EOF被置成0。以后允许DOS为该文件自动分配空间和重新分配空间。

2.16 DATE　设定日期

1. 命令格式

DATE [,mm/dd/yy]

2. 说明

如果打入命令时不指定参数，DATE命令就以月/日/年的格式显示目前系统的日期。

如果给定mm/dd/yy参数，则该值就成为系统的日期，并置入实时时钟。mm是月（取值是01—12)。dd是日（取值是01—31)。yy是年（取值是00—99)。除了限定取2位十进制数以外，并不对这三个值表示的时间真实性作检查。当机内时钟达到24:00:00，接着就恢复成00:00:00,设定的日期中的“日”就要加1。对于TRS-80 I型机，月的结束和年的结束不作调整。对于Ⅲ型机，月和年的结束由ROM进行调整。

在复位时，日期被置成相应于SYSTEM命令选用参数AY或AZ所规定的状态。

3. 举例

(1)DATE

显示系统的日期。

(2)DATE, 05/01/84

设定系统日期为1984年5月1日。

2.17 DEBUG　允许或禁止DEBUG功能

1. 命令格式

DEBUG [,yn]

2. 说明

(1)DEBUG或DEBUG, Y　允许DEBUG程序工作（但不运行)。此后，只要用户程序（如BASIC, SCRIPSIT, PROFILE, EDIT等）开始工作，就可使DEBUG进入运行。进入DEBUG发生在用户程序装入完成以后,而在第一条指令执行以前。预先输入DEBUG程序的目的是使得程序的调试者在程序开始执行以前，就可以用DEBUG功能去改变程序的状态或它的初始参数。

(2)DEBUG, N　使得DEBUG处于不工作状态。在复位及打开电源时，DEBUG是处

于不工作状态（即禁止状态）。

这个命令不影响'1 2 3'联键操作，即同时按下 1,2,3 三个键时可以进入 DEBUG 功能。详细情况请看 4.1节 DEBUG 功能说明。

2.18 DIR　显示磁盘目录信息

1. 命令格式

DIR [:] [dn1] [,A] [,S] [,I] [,U] [,/ext] [,P]

2. 说明

（1）这个命令可以显示装配于 dn1 驱动器上的盘片的目录信息，如果不指定 dn1，就认为是 SYSTEM 选用参数 AN 规定的驱动器。

最先显示的一行包含该驱动器的编号，盘片名，它的日期，磁道数目，空闲的 FDE 数和空闲的 gr. 数。例如：

DRIVE 0　NEWDOS80 00/00/00 80TRKS　28FDES　93GRANS

磁道数和空闲 gr. 值是根据该盘片目前有效的 PDRIVE 特征说明，由 DOS 扣除了已占用的 FDE 和 gr. 后算出的。如果该特征说明与实际不符，则显示值可能是错的。

（2）接着显示的是该盘片上的文件的目录。如果不给定 A 参数，每行显示 4 个文件，每个文件只给出文件名和文件名扩展符（如果有的话）。如：

TL1/CMD	PASCAL/CMD	ZZZ	SARGON/CMD
MOONBASE/BAS	NWD80V2/XLF	NWD80V2/ILF	EDTASM/CMD
CHAINTST/JCL	LMOFFSET/CMD	INVADER/CMD	URANAI/BAS
ASPOOL/MAS	CHAINBLD/BAS	STARWARS/BAS	DISASSEM/CMD
DIRCHECK/CMD	GOLF/BAS	LUPIN/BAS	MARJONG/BAS
SUPERZAP/CMD	FIXSEC/CMD	HAN30/BAS	

如果指定了 A 参数，DIR 命令将在每个显示行上只列出一个文件，显示行还包括其它的一些内容，下面是一个实例：

	EOF	LRL	RECS	GRANS	EXTS	SIUEC····UAL
TL1/CMD	21/123	256	22	5	1	··U·····0
PASCAL/CMD	24/000	256	24	5	1	··U·····0
ZZZ	0/244	256	1	1	1	··U·····0
SARGON/CMD	35/000	256	35	7	1	··U·····0
MOONBASE/BAS	13/056	256	14	3	1	··U·····0
NWD80V2/XLF	2/054	256	3	1	1	··U·····0
NWD80V2/ILF	2/002	256	3	1	1	··U·····0
EDTASM/CMD	35/000	256	35	7	1	··U·····0
CHAINTST/JCL	1/050	256	2	1	1	··U·····0
LMOFFSET/CMD	10/000	256	10	2	1	··U·····0
INVADER/CMD	17/000	256	17	4	1	··U·····0
URANAI/BAS	36/176	256	37	8	2	··U·····0
ASPOOL/MAS	10/000	256	10	2	2	··U·····0
CHAINBLD/BAS	19/058	256	20	4	1	··U·····0

STARWARS/BAS	14/026	256	15	3	1	··U·····0
DISASSEM/CMD	25/000	256	25	5	1	··U·····0
DIRCHECK/CMD	15/000	256	15	3	1	··U·····0
GOLF/BAS	24/086	256	25	5	1	··U·····UA5
LUPIN/BAS	13/249	256	14	3	1	··U·····0
MARJONG/BAS	38/102	256	39	8	1	··U·····0
SUPERZAP/CMD	30/000	256	30	6	1	··U·····0
FIXSEC/CMD	1/183	256	2	1	1	··U·····0
HAN30/BAS	39/144	256	40	8	1	··U·····0

其中各项的含义如下：

* 文件名。

* 文件名扩展符（如果有的话）。

* 以 xxx/yyy 格式表示的文件 EOF 值，其中 xxx 是文件中的相对扇区值(表示文件占了几个扇区)，而 yyy 是最后一个扇区中的相对字节数。

* 以字节表示的文件逻辑记录的大小(LRL)。

* 文件中的逻辑记录数(RECS)。

* 分配给该文件的 gr. 数(GRANS)

* 分配的磁盘空间范围元数(EXT)。

* 提供文件信息的 12 个标志，定义如下：

① S=系统文件。

② I=隐性文件，见 DOS 命令 ATTRIB。

③ U=在上一次由 PROT 命令清除修改标志以后修改过的文件。

④ E=不允许为文件分配比现有更多的空间。

⑤ C=在 DOS CLOSE 期间，超出 EOF 范围的剩余文件空间没有自动释放。

⑥—⑨ 留给以后使用。

⑩ U=存在非空格修改通行字。

⑪ A=存在非空格存取通行字。

⑫ L=保护级别。见 ATTRIB 命令。

只有指定了 S 参数，才能显示系统文件。

只有指定了 I 参数，才能显示隐性文件。

在 DIR 命令中用了 I,S 参数以后显示出的部分内容如下：

	EOF	LRL	RECS	GRANS	EXTS	SIUEC····UAL
BOOT/SYS	5/000	256	5	1	1	SI·····UA6
SYS6/SYS	35/000	256	35	7	1	SIU·····UA7
SYS14/SYS	5/000	256	5	1	1	SIU·····UA7
BASIC/CMD	18/000	256	18	4	1	.IU·····0

如果指定了 U 参数，只有标有修改标记的文件才显示。带有修改标记的文件就是指自上一次目标盘片上的修改标志由 PROT 或 ATTRIB 命令清除以后，要通过标准的 DOS I/O 写子程序改变的那些文件。

如果指定了 /ext，则只有那些有文件扩展符 ext 的文件被显示出来。ext 可以取 0-3 个字符。例如：DIR, 1, /CMD 将列出所有带有 CMD 扩展符的文件，如 EDTASM/CMD, DISASSEM/CMD 等。

如果同时指定了 U 和 /ext 这两个参数，则只有满足这两个条件的文件才被列出。

如果显示内容已占满了整个屏幕，DIR 命令就会显示出一个'？'号，并等待用户打入 ENTER 作回答，以便继续显示后面的内容，或者打入 BREAK 键，以终止 DIR 功能。

如果指定了 P 参数，将会把有关目录的信息送到打印机去打印，不再在显示屏幕上显示。请注意：如果打印机处于没有准备好的状态，系统将会挂起来，一直等到打印机准备好为止。

如果指定 $ 参数，DIR 在列出目录清单之前将要求用户装上目标盘片，在退出 DIR命令以前，系统要用户重新装入系统盘片。$ 参数只有在系统内只有一个驱动器（驱动器编号 dn1＝0）时才用。DIR 命令本身没有改变 PDRIVE 特征说明的能力。

（3）用户必须记住，如果不指定 dn1，缺项驱动器编号是由 SYSTEM 选用参数 AN 决定的，而 AN 不一定是 0。

3. 举例

（1）DIR 0

显示目前装配在 0 号驱动器上的盘片的所有非系统的，非隐性的文件的文件名和文件名扩展符。每个显示行将列出四个文件。

（2）DIR, 0, S, I, P

与上例相同，只是系统文件和隐性文件也要列出，并且把清单送到打印机去打印，以代替荧光屏的显示。

（3）DIR, 1, /DAT, U

显示装配在 1 号驱动器内的盘片上的所有记有修改标志和文件名扩展符为 /DAT 的文件的文件名和文件名扩展符。

（4）DIR, 2, A

显示 2 号驱动器内盘片上的所有非系统和非隐性文件，每行显示一个文件。由于每行只显示一个文件，因而通常要显示若干页画面，用户只要按 ENTER 键，就可以逐页的显示，如果按下 BREAK 键，就会终止 DIR 功能。

（5）DIR $0

与例 1 相同，只是系统将要求用户把目标盘片装配于 0 号驱动器，而且当 DIR 命令执行完后，系统将要求用户重新把系统盘片装入 0 号驱动器。本例适用于系统内只有一个驱动器，而又要查看另一个不带 NEWDOS/80 系统的盘片目录的情况。

2.19 DO 用磁盘文件代替键盘输入

1. 命令格式：

DO, filespec 1 [,sectionid]

2. 说明：

DO 命令完成的工作与 DOS 命令 CHAIN 完全相同（见 2.9 节）。

2.20 DUMP　把内存储器内容转储到磁盘上

1. 命令格式

DUMP, filespec 1, start-addr, end-addr [,entry-addr [,relloc-addr]]

2. 说明

DUMP 命令把主存储器内的数据写到 filespec 1 文件内，从 start-addr（开始地址）的字节开始，并以 end-addr（终止地址）的字节结束。

（1）start-addr（开始地址），end-addr（终止地址），entry-addr（入口地址）和 relloc-addr（浮动地址）是小于 65536（十进制）或10000H（十六进制）的一个数值。如果该值是十六进制，则必须带有 H 后缀（如 8000H），否则就认为是一个十进制值。开始地址和浮动地址可以是 0 到 FFFFH 之间的任意一个值。

（2）根据入口地址值的不同，这个命令可以用两种方式工作。如果入口地址值等于65535（即 FFFFH），则把内存储器内的内容一个不漏地转储到盘上。开始地址值被存放在文件的第一、第二字节内，文件的其余部分是没有插入任何控制字节的存储器转储内容。这个存储器转储文件可以用 SUPERZAP 程序的 DMDB 命令在荧光屏上显示，或在打印机上打印出来，这样就可以把它留在以后再作调试，或者拿到另一台 TRS-80 计算机上去调试。

如果入口地址小于 65535，或者没有指定入口地址，则认为指定的存储器区内放的是可执行的机器代码，并以装入程序的格式送到文件中。于是，以后能用 NEWDOS/80 的装入程序把这个程序读回内存储器，或者执行这个程序（见 LOAD 命令）。如果用户不指定入口地址，则系统就用 402DH 作为入口地址（使得返回 DOS READY）。注意：如果用户企图执行或装入一个开始地址低于 5200H 的文件，就会破坏 DOS 的内容，使得系统不能正常工作。

（3）浮动地址确定了用 LOAD 命令把起始地址到终止地址之间的那些字节应该放在什么地方，或者是当程序执行时应该放在什么地方。在写目的文件期间，浮动地址减去开始地址得到的差值要加到目的文件中每个装入地址上。如果入口地址处于开始地址到终止地址的范围内，则差值也要加到入口地址上。实际的目的代码是不改变的，所改变的仅是装入程序的控制信息。

（4）如果 filespec 1 没有指定文件名扩展符，则不会象 TRSDOS 那样自动的提供一个。

3. 举例

（1）DUMP, PROGRAM/CMD:1, 5200H, 9ABCH, 54EDH

把 5200H（包括 5200H）到 9ABCH（包括 9ABCH）的存储器内容转储到目前放在 1 号驱动器内的盘片上的文件 PROGRAM /CMD 中。转储的文件是装入程序格式，并具有 54EDH 的入口地址。今后该文件可由 DOS 命令

LOAD, PROGRAM/CMD

装回存储器内，或用 DOS 命令

PROGRAM〔, 参数〕

执行。

（2）下例是假定用户程序因某种原因进入了死循环或者出了错误，而用户的查错不能立即奏效，但是该计算机必须接着作其它使用时的一种处理方法。

这时如果有一个具有足够空间的已格式化备用盘片，就可以用'DEF'（即同时按下D,E,F这三个键）进入MINI-DOS状态，如果此时计算机已在 DOS READY 状态，就不必进入MINI-DOS状态，接着可以用下面的命令：

DUMP,TROUBLE/MEM:2,0,65535,65535把主存的65536个字节（包括ROM，显示RAM和所有其它RAM的内容）转储到文件TROUBLE/MEM内。该文件最前面的2个字节将是0000H，它是转储文件的开始地址，文件的其余部分是不含控制代码的存储器内容。一旦转储完成，用户应该取走记有转储文件的盘片，留作该程序的调试者以后使用，并复位计算机，以便运行其它的程序。以后，调试者可用SUPERZAP程序的DMDB命令去显示或打印TROUBLE/MEM文件的内容，好象该文件当时就在内存储器中一样，并继续查找错误。但是用户应该记住：在错误发生以后和转储开始以前，DOS工作区4000H—51FFH已被DOS的工作（包括DUMP命令）所改变。

2.21 ERROR 显示DOS错误信息

1. 命令格式

   ERROR, xx

2. 说明

显示相应于错误代码xx的DOS错误信息，其中xx是一个0到63之间的整数。

3. 举例

(1)ERROR, 24

将在荧光屏上显示FILE NOT IN DIRECTORY（目录中没有这个文件）信息。

2.22 FORMAT 格式化一个NEWDOS/80用的盘片

1. 命令格式

   FORMAT, dn2[=tc2], name 2, mm/dd/yy, password 3 [,N][,Y][,NDMW] [,DDND] [,OLD=name 1] [,KDN] [,DDSL=ln 1] [,DDGA=gc 1] [,DPDN=dn 4] [PFST=tn 3 [,PFTC=tc 3]]

2. 说明

从制造厂买来的磁盘片不能立即在NEWDOS/80系统中使用。首先必须把盘片分割成许多磁道，每个磁道又分割成若干个扇区，每个扇区可以容纳256个字节。该磁盘的总容量的15—30%用来放置格式控制信息，不能再用来放用户数据。

DOS命令FORMAT就是进行这种磁盘格式化工作的，并设定合适的磁道和扇区，还要建立两个系统文件（BOOT/SYS和DIR/SYS），这是每个盘片上都必须要有的系统文件。格式化完成以后，这个盘片就可以供NEWDOS/80作为数据盘片用。

COPY命令（格式5和格式6）中也包含有这种格式化处理。

要注意：FORMAT命令不能在MINI-DOS状态执行。

(1)在NEWDOS/80 2.0中，在对磁道格式化后和磁盘臂步进到下一个磁道之前，就立即读该磁道的各个扇区。然后在所有磁道格式化以后，如果SYSTEM选用参数BM=Y,则在VERIFYING（复核）阶段要读整个盘片。如果设定BM=N,就跳过这个复核阶段。用户如不满足于磁道格式化时作的检验，就应选定复核选用参数BM。

（2）FORMAT 命令不允许用户指定要封锁的磁道，当碰到一个不可复核的扇区时，并不把相应的磁道封锁字节置成 FF（FF 表示封锁）。只是为了要与 TRSDOS 兼容，才在标准的磁盘目录中装有封锁表，实际上，NEWDOS/80 是不用它的。要记住：NEWDOS/80 不在目录上登录磁道，只登录 gr. 组 (lump)，gr. 组可以覆盖若干个磁道。NEWDOS/80 是在这种特定的情况下工作的，带来的缺点是：如果一个盘片不能完全通过格式化，就不能使用（即不允许有损坏或通不过格式化处理的磁道）。

（3）使用 FORMAT 命令时要求给出命令中规定的全部参数。该命令不会对用户提出任何提示信息。

dn 2 是格式化期间所用的目的驱动器编号。name 2（名字 2）是要赋予该盘片的名字，只有在指定了 KDN 参数，保留老名字时，才不用 name 2，但是在这种情况下 FORMAT 命令中仍然要给出 name 2，只是不用它罢了。mm/dd/yy 是用来给该盘片指定日期的，如果指定了 KDD 参数，则不用 mm/dd/yy。但是 FORMAT 命令中还得给出这个参数值。password 3（通行字 3）是赋予该盘片的通行字。它必须符合通行字的有关规定。

（4）当盘片名字、日期和通行字等参数空缺时，就引用它们的缺项值。它们的缺项值分别是 NOTNAMED，系统日期和 PASSWORD。要用几个空缺参数可以由用户决定，但是用空缺参数的地方必须要用逗号作分界，不准用空格来代替。具体格式可以参看后面的例2，3和 4。

（5）为了尽可能的使 FORMAT 和 COPY 使用相同的 NEWDOS/80 的程序段，所以选用参数尽量与 COPY 命令的格式 5 和格式 6 保持一致。这两个命令的主要差别仅在于FORMAT 只进行格式化处理，而 COPY 命令既要进行格式化，还得进行复制。读者应该读一下 COPY 命令的格式 5 和格式 6 的有关内容（见 2.14 节），两个命令中相同的参数不再列出，只给出不同的地方及两个新增加的参数。

如果既不给定 N 参数，也不给出与它互斥的参数，就认为 N 是它的缺项值。

如指定了 tc 2，则该盘片将被格式化为 tc2 数目的磁道，否则该盘片将被格式化为那个驱动器的缺项磁道数目（见 PDRIVE 命令）。如果 tc2 值大于该驱动器能管理的磁道数，企图步进到实际上并不存在的磁道时，就会使格式化进行不下去。

（6）新增加的 PFST=tn3 和 PFTC=tc3 这两个参数使得可以格式化一定区域内的磁道，而不是整个磁盘。如果指定了 PFST，就不能再指定 tc2，但是指定了 PFTC 后，就必须指定 PFST，这两个新增加的参数必须配对使用。PFST 参数的含义是局部格式化的起始磁道，也就是说 tn3 规定了要格式化的第一个磁道。如果驱动器 dn1 可以使用 PDRIVE TI 标记的 J或 K 参数，则 DOS 要把 1 加到 tn3 内。PFTC 参数的含义是局部格式化的磁道计数，因而 tc3就规定了要格式化的编号连续递增的磁道数目。如果不规定 PFTC，而规定了 PFST，则认为 tc3 等于1。完成了 tc3 数目的磁道格式化以后，同时 SYSTEM 参数 BM=Y，则将会复核整个盘片。如果全盘片复核过程中发现有问题，则按下↑键，就可以勾消所做的格式化。请记住：每个磁道格式化以后，该磁道内的那些扇区已经即刻作了一次检验，这不包括在全盘片复核操作中。

3. 举例

（1）FORMAT, 0, AAA, 08/01/82, PSWD, Y

根据 DOS 要求装配在 0 号驱动器内的盘片将按目前该驱动器的 PDRIVE 特征说明进行

格式化。DOS 并不管该盘片格式化以前是否包含有数据。盘片命名为 AAA，日期为 1982 年 8 月 1 日，盘片主通行字设定为 PSWD。

(2)FORMAT, 0, , , , Y

这个例子与上例极其相似，只是盘片名字，日期和通行字用了缺项值。也就是说盘片命名为 NOTNAMED，用当前系统盘片的日期作为它的日期，并把 PASSWORD 作为它的通行字。

(3)FORMAT, 1, XXX,, PSWD, N, NDMW, DPDN=4, DDSL=40, DDGA=6

已经装配在 1 号驱动器内的盘片必须不包括可识别的数据。它是根据系统盘片的PDRIVE 规定的 4 号驱动器说明进行格式化（而不是根据现有的 1 号驱动器说明来格式化的）。赋予它的名字为 XXX，通行字为PSWD，它的日期取自当前系统的日期。目录起始于gr.组40 的开端，并且 gr. 组包含有 6 个 gr.（目录最多允许有 222 个文件）。由于设定了参数 NDMW，DOS，不对格式化盘片的装配作提示，发生了错误也不给出显示，不允许重试，而是终止当前的操作。

(4)FORMAT, 1,,,, Y, PFST=22, PFTC=2

假定发生了严重的错误，使得一个盘片上的磁道 22 和 23 的格式被破坏了；使用SUPERZAP 程序已经检查出确实在这两个磁道的每一个上，至少有一个扇区发生 SECTOR NOT FOUNT（找不到扇区）的错误，而且你已经使用 SUPERZAP 程序的 CDS 或 SCOPY命令，把这两个磁道的可读扇区保存在另外的空扇区中（可以在同一盘片上，也可以在另外一个盘片上）。执行本例的 FORMAT 命令，将只对这两个磁道格式化，该盘片上其余部分的信息不受影响。当完成上述操作后，就能够把保存的扇区内容移回来，这样就恢复了有问题的扇区。

2.23 FORMS　设置打印机参数（只适用于 TRS-80 Ⅲ型）

1. 命令格式

FORMS [,WIDTH=xxx] [,LINES=yyy]

2. 说明

FORMS 命令可以随意改变打印机的参数，也可用于列出打印机的参数。

* WIDTH=xxx　规定了每行的字符数，其中 xxx 必须是 9—255 之间的一个值，而 255 表示不限制每行的字符数。如果不规定 WIDTH，则不改变每行的字符数。

* LINES=yyy　规定了每页的行数，其中 yyy 必须是 1—254 之间的一个值，而 254 表示每页不限制行数。如果不规定 LINES，则不改变每页的行数。

3. 举例

(1)FORMS, WIDTH=80, LINES=60

每行字符数设定为 80 个，每页为 60 行。

(2)FORMS, WIDTH=255, LINES=254

不限定每行的字符数和每页的行数。

(3)FORMS

显示当前打印机设定的每行字符数和每页的行数。

2.24 FREE 显示当前装配好的每个盘片的空闲 gr. 数和空闲的 FDE 数

1. 命令格式

FREE [,P]

2. 说明

有时为了把一个用户程序存放到磁盘上去，就需要知道该盘片是否还有空余的空间，空余磁盘空间是否容纳得下这个程序，这时就要用到这个命令。

对于每个装有盘片的驱动器，FREE 命令将显示该驱动器编号，盘片名字，盘片日期，该盘片的磁道数，空闲 FDE 数和空闲 gr. 数。

如在命令中指定了 P 参数，信息将发送到打印机去，不再在荧光屏上显示。

3. 举例

(1)FREE

对于目前装配的每个盘片，在荧光屏上显示出空闲 gr. 数和空闲 FDE 数。

(2)FREE, P

可以把有关信息送到打印机去打印，打印结果的格式如下：

DRIVE 0 NEWDOS80 00/00/00 80 TRKS 16 FDES 30 GRANS

2.25 HIMEM 设定 DOS 可使用的最高内存储器地址值

1. 命令格式

HIMEM [,addr 1]

2. 说明

DOS 在 TRS-80 I 型的 4049H 单元（Ⅲ型为 4411H 单元）保存着 2 字节的最高内存储器地址。这个最高内存储器地址值是供 COPY, BASIC, EDTASM, DISASSEM和 LMOFFSET 等程序用的，它作为这些程序能够使用的存储器上限。用户程序也使用这 2 字节 HIMEM 值作为内存储器的上限。注意！程序装入期间，装入程序不用它作为内存储器上限。

如果不指定参数，HIMEM 命令以十六进制格式显示当前最高存储器地址值。

如果指定了 addr1，则 DOS可用的最高存储器地址被限定为 addr 1，addr 1必须是28672和65535（十进制）之间的一个整数（即 7000H—FFFFH)。

3. 举例

(1)HIMEM

显示当前 DOS 使用的最高存储器地址值。

(2)HIMEM, 49000

设定 DOS 使用的最高存储器地址为 49000（即 BF68H)。

2.26 JKL 把当前荧光屏上的内容发送到打印机

1. 命令格式

JKL

2. 说明

JKL 命令没有参数。这个命令与同时按 J,K,L 三个键的联键操作（见 4.5 节),用的是相

同的子程序。JKL 命令可以把当前荧光屏上显示的内容传送到打印机去打印。如果 SYSTEM 参数AK＝Y,则大于或等于 80H 的代码（包括代表图象的代码）将按原样传送；若 AK＝N，则上述代码将用圆点来代替。所有小于 20H 的代码均显示为圆点。在 JKL 命令执行期间，可以按 BREAK 键去终止 JKL 命令的执行。

JKL 命令主要是在 BASIC 中通过 CMD "JKL"使用，或在用户程序中通过 DOS-CALL 来使用。

2.27 KILL　删除文件

1. 命令格式

KILL, filespec 1

2. 说明

这个命令的功能是从盘片上删除一个文件。删除以后，该文件不能用通常的方法来存取，而且 DOS 也不再能识别它们。

（1）从装配于规定驱动器的盘片上删除具有 filespec 1 的文件。如果不指定驱动器编号，则从 0 号驱动器开始搜索所有装配好的盘片，把找到的第一个具有指定文件名和文件名扩展符的文件删除。

（2）KILL 命令的作用如下：

＊ 假如在盘片上已为一个文件分配了空间，则只有把这个空间释放出来以后，才能再分配给其它的文件用。KILL 命令就起着把属于某一文件的空间取回来的作用，KILL 命令本身并不能改变盘片上文件的数据（如果有的话）。这些数据（虽然不再可存取）在与之有关的文件空间尚未重新分配给其它文件之前是不被覆盖的，所以这些扇区上实际仍写有这些数据。

＊ 使每个文件目录项的第一个字节的位 4 成为 0，并使与每个文件相联系的 HIT（散列检索表）扇区字节成为 0，就可使文件的 FPDE 和 FXDE 释放出来。在 DOS 重新把文件目录项(FDE)分配给另一个文件以前，除了这个位4以外，用通常的 DOS 操作不能改变与FPDE 和 FXDE 有关的其它任何信息。关于 FDE, HIT, FPDE 和 FXDE的说明请参阅第五章，那里有详细的描述，这里不作介绍。

（3）如果用户不小心删除了一个不应该删去的文件，由于该文件的 FDE 和占用的磁盘空间要到 DOS 需要用它们时才改变，所以有可能重新恢复 FPDE 和 FXDE，以及文件空间。当然，要做到这一点，必须十分熟悉目录的结构及其工作情况，在 Apparat 公司提供的资料中没有这方面的内容，因为这不是一下子说得清楚的。为了使读者学会恢复文件的方法，本书第五章的 5.10 节将会教你如何去完成这项艰难的工作。

如果你一次要从盘片上删除几个文件，请用 PURGE 命令。

3. 举例

（1）KILL XXX/BAS:1

删除装配在 1 号驱动器内盘片上的 XXX/BAS 文件。

（2）KILL　YYY

由 0 号驱动器开始，依次搜索装配好的盘片，一直到从中找到第一个 YYY文件时为止。然后删去这个文件。如果其它装配好的盘片上还有 YYY 文件，则不再删去。

2.28 LC　设定键盘的 a—z 按键为指定状态

1. 命令格式

LC [,yn]

2. 说明

(1)LC 或 LC,Y　把键盘小写字母 a—z 的按键设定为小写字母 a—z。

(2)LC,N　把键盘小写字母 a—z 的按键设定为各相应的大写字母 A—Z。

对于 TRS-80 I 型微计算机，只有在使小写字母驱动程序有效时，LC 命令才起作用(见下面 LCDVR 命令)。

2.29 LCDVR　小写字母驱动程序（只适用于 TRS-80 I 型）

1. 命令格式：

LCDVR [,x [,s]]

2. 说明：

在 NEWDOS/80 1.0 版本中，英文小写字母的驱动程序（处理键盘小写字母和发送小写显示字符到显示器的一种程序）是一个在存储器高区执行的独立的程序。在 2.0 版本中，小写字母驱动程序是完整的 I 型 NEWDOS/80 的一个组成部分。

(1)LCDVR 命令中各个参数的定义是：

* 如果 x=Y，则英文小写驱动子程序处于有效状态,如果 x=N，则该子程序无效。当小写字母驱动子程序处于有效状态时：

① 根据选择小写字母的锁定按键位置来处理键盘输入的 a—z 字符。

② 把 ASCII 代码 96—127（即60H—7FH）显示成它的相应字符，而不由 ROM显示子程序把它们改变为 64—95（即40H—5FH）的相应大写字母。

* 只有在 x=Y 时，LCDVR 命令中的第二个参数 s 才有意义,这个参数完成了 LC 命令的功能，如果 s=Y，则初始设定 a—z 锁定按键为 a—z 状态，如果 s=N，则把 a—z 变换成 A—Z。

(2)一旦小写字母驱动程序处于有效状态，按下 SHIFT 0 键，将使驱动程序在接受小写字母和把小写字母变换成大写字母这两种工作方式之间来回切换。将来，要把 LC 命令改成可以明确设定这两种方式中的任意一个。

(3)为了使用小写驱动程序，必须使 NEWDOS/80 的键盘和显示子程序有效。其它一些要禁止这两个 NEWDOS/80 子程序的程序（不包括 ROUTE）也将禁止小写字母驱动程序的工作。

(4)如果不指定 LCDVR 命令中的参数，DOS 就认为是 LCDVR,Y,N。

(5)这个小写字母驱动程序与 1.0 版本提供的 LCDVR 程序有些地方是不同的。在 NEWDOS/80 1.0 版本中，如果正在把小写字母 a—z 变换为大写字母 A—Z，则同时会把大写字母变换为小写字母。NEWDOS/80 2.0 版本不会把大写字母变换为小写字母，也就是说，真正做到了大写字母的锁定。

3. 举例

(1)LCDVR

使小写字母驱动子程序有效，同时把小写字母开关设定为把小写字母 a—z变换为大写字母 A—Z。

(2)LCDVR, Y, Y

使小写字母驱动子程序有效，初始设定为小写字母开关处于接受小写字母 a—z的状态。

(3)LCDVR, N

使小写字母驱动程序无效。

2.30 LIB 显示 NEWDOS/80 2.0 的库命令

1. 命令格式

LIB

2. 说明

LIB 命令不需要任何参数。它可以显示 NEWDOS/80 2.0 的全部库命令。

在库命令中的 FORMAT, COPY 和 APPEND 命令在存储器 5200H 及以上区域内执行，它们与 CHAIN 命令一起都不能在 MINI-DOS 状态执行。而另外一些在 DOS 覆盖区(4D00H—51FFH)执行的命令 (CHAIN 命令除外) 是能够在 MINI-DOS 状态执行的。

3. 举例

(1)用LIB命令显示NEWDOS/80 2.0的库命令。

只要用键打入LIB及ENTER键，即可得到如下的显示信息：

APPEND	ATTRIB	AUTO	BASIC2	BLINK	BOOT	BREAK	CHAIN
CHNON	CLEAR	CLOCK	CLS	COPY	CREATE	DATE	DEBUG
DIR	DO	DUMP	ERROR	FORMAT	FREE	HIMEM	JKL
KILL	LC	LCDVR	LIB	LIST	LOAD	MDBORT	MDCOPY
MDRET	PAUSE	PDRIVE	PRINT	PROT	PURGE	R	RENAME
ROUTE	STMT	SYSTEM	TIME	VERIFY	WRDIRP		

2.31 LIST 在显示器上列出文本文件

1. 命令格式

LIST, filespec 1[, start-line [,line-count]]

2. 说明

(1)这个命令把 filespec 1文件内容发送到显示器。filespec 1 文件未必一定是文本文件，所以如果它不是一个文本文件，则得到的显示将是无意义的。属于文本文件的是：用带有 A 参数的SAVE命令存储的BASIC程序，用PRINT写的BASIC文件，汇编、FORTRAN 和 COBOL 的源文本文件，用带有A参数的SAVE命令存储的SCRIPSIT文件和电笔 (Electric Pencil) 文件等。要显示非文本文件，只能利用SUPERZAP程序。

(2)对字符的显示含义不作检查，只是把十六进制值 80H—FFH 之间的字符调整到 00H—7FH范围内，并用一个圆点来代替所有十六进制值小于20H和大于由SYSTEM参数 AX 规定的最高ASCII字符值的字符。

(3)如果指定了 start-line (开始行)，可取值为十进制的 1—65535，则清单将由该行 开始，所谓一行就是由ENTER (回车) 或EOL字节(0DH)为结束的一串字符。

（4）如果指定了 line-count（行数），则显示的行数就被限定为这个数，如果文件中的实际行数小于这个数值，则从文件开始行算起直到显示完为止。如果命令中指定了行数，就一定要确定start-line。

（5）按⊡键将会在碰到十六进制字符 0DH 或显示完256个字符（不论这两种情况中哪一个先出现）时，就暂停显示。按下ENTER键将会继续显示。按⊡键将终止LIST命令。

（6）LIST 命令除了刚才介绍的可列出一个文件的内容以外，对于那些在开始部分标有日期/时间的文本文件还有这样的一种用途：如果一个文本文件有多个副本，可以用 LIST 命令来查看文件的开始部分，找出一个最新的副本。

3. 举例

（1）LIST, BASEPROM/BAS

显示文件BASEPROM/BAS的全部内容。

（2）LIST, XXX, 1, 6

显示文件XXX的最前面的 6 行。

（3）LIST, YYY:1, 200

显示 1 号驱动器内盘片上的文件YYY,从第200行开始一直到该文件结束的全部内容。

3.32 LOAD　把Z80机器语言文件装入RAM

1. 命令格式

LOAD, filespec 1

2. 说明

（1）这个命令可以把filespec 1的Z80机器语言文件装入 RAM，并把它的入口地址装入从4403H（十进制 17411）开始的两个单元内。被装入文件必须具有一种规定的装入格式，如DUMP或EDTASM建立的文件就符合这种格式，因为要从该文件中取得控制数据才能完成装入工作。如果文件装入时覆盖了DOS驻留程序(4000H—4CFFH)的任何一部分或DOS覆盖区（4D00H—51FFH），将会产生严重的灾难性后果，至少也会使系统死锁。

（2）要把一个程序或数据装进RAM，供其它程序使用，就要用到LOAD命令。例如，通过BASIC的USR语句调用的机器语言程序就要用 LOAD命令装入。请记住，入口地址保存在4403H和4404H，但在TRSDOS中入口地址不放在这两个单元中。

3. 举例：

（1）LOAD, OVERLAY/OBJ:1

把装配在 1 号驱动器内盘片上的目的代码程序 OVERLAY/OBJ 装入存储器。在 OVERLAY/OBJ中包含的装入控制信息决定了该装入什么内容和把它装入存储器中的什么地方。

（2）假定BASIC不用整个高区存储器，而且BASIC程序希望把 USR3PGM/OBJ 程序装入高区存储器，并要求以后用BASIC USR3执行它。

这个例题的解答是执行BASIC语句

CMD "LOAD, USR3PGM/OBJ"

DEFUSR3=(PEEK (17411)+256*PEEK(17412))−65536就可以完成题目中的要求。

2.33 MDBORT　终止MINI-DOS并转入DOS READY状态

1. 命令格式

MDBORT

2. 说明

MDBORT命令没有参数。只有当NEWDOS/80处于MINI-DOS状态，才有必要使用这个命令。这个命令用于终止MINI-DOS和清除原来的MINI-DOS状态，使系统返回DOS READY状态。

为了进入MINI-DOS可以同时按下D,F,G键，但是这时运行的程序也被中断了，当用户不想继续执行这个被中断的程序时，就可以使用MDBORT命令。

2.34 MDCOPY 在MINI-DOS状态复制一个文件

1. 命令格式

MDCOPY, filespec 1 [,TO], filespec 2

2. 说明

常规的 COPY 命令不能在 MINI-DOS 状态下执行。MDCOPY 命令使用户在 MINI-DOS 状态可执行一个有限制的和十分缓慢的文件复制操作。

MDCOPY 命令把 filespec 1 文件复制到新的或原有的 filespec 2 文件上。不改变 filespec 1 文件的内容，而 filespec 2 文件原先的内容(如果有的话) 要被破坏。不允许像 COPY 命令那样对 filespec 2 作省略。

3. 举例

(1)MDCOPY XXX/DAT:0 YYY/DAT:1

在 MINI-DOS 状态，把目前装配在 0 号驱动器内盘片上的文件 XXX/DAT 复制到目前装配在 1 号驱动器内的盘片上，并取名为 YYY/DAT。

2.35 MDRET 退出 MINI-DOS 状态，返回主程序

1. 命令格式

MDRET

2. 说明

MDRET 命令没有参数。

(1)打入 MDRET 命令后，系统退出 MINI-DOS 状态，并返回主程序，由断点开始继续执行，这里指的断点就是为了进入 MINI-DOS 状态而把程序中断的地方(用'DFG' 联键操作所打断的地方)。如果 'DFG' 联键操作以前，已经显示出光标，则光标将重新再显示出来。如果 'DFG' 联键操作时，键输入缓冲器内有输入的记录，则该部分记录还在那里，但是它不再在荧光屏上显示出来。在退出 MINI-DOS 状态以后，用户应该在他停下的地方继续打入要输入的字符。

(2)如果在实时钟中断，而不是键盘输入中断时进入 MINI-DOS 状态，会带进不需要的输入字符，于是在执行 MDRET 命令以后，会在荧光屏上显示一个或多个 D,F,G 字符，这是一种用户不需要输入的字符，应该用←键去删除这种多余的字符。用户和 DOS 都没有办法去防止这种不需要的输入字符，所以在打入或修改一个文本文件 (如 SCRIPSIT 或电笔编制文件) 的时候，不应该使用 'DFG' 联键操作。最好在进入命令方式后再使用 'DFG' 联

键操作，这样就可以用←键删除不需要的输入字符，而不使文件发生错误。

2.36 PAUSE　显示信息并暂停执行，等待用户按 ENTER 键

1. 命令格式

PAUSE, msg

2. 说明

如果显示 PAUSE 命令本身，则不再显示 msg 信息。如果不显示 PAUSE (在 DOS-CALL 情况下执行 PAUSE)，则显示 msg 信息。在任何情况下，都会在下一行显示 PRESS″ENTER″ WHEN READY TO CONTINUE (当作好了继续运行的准备时，请按 ENTER 键)。然后，DOS 等待用户按 ENTER 键。PAUSE 命令是链接状态中产生暂停的方法之一，此外，DOS-CALL 正在执行主存储器中的一串命令时，也可能会用到 PAUSE 命令。

3. 举例

(1)PAUSE, MOUNT DISKETTE LABELED ″PRIMARY″ ON DRIVE 1

执行上述命令时，会把 PAUSE 后面的信息送到荧光屏上显示，并在下一行显示信息 PRESS ″ENTER″ WHEN READY TO CONTINUE。DOS 等待用户按 ENTER 键，用户在把合适的盘片装配在 1 号驱动器内以后，就可按 ENTER 键了。DOS 并不检查用户是否做了应做的工作 (目前情况就是把标记有 PRIMARY 的盘片插入 1 号驱动器内)，实际上，DOS 只是等待用户按 ENTER 键而已。

2.37 PDRIVE　为驱动器分配缺项属性

1. 命令格式

PDRIVE [,password1:] dn1, [dn 2 [=dn3]] [,TI=type1] [,TD=type2][,TC=tc1]
[,SPT=sc 1] [,TSR=rc 1][,GPL=gc2] [,DDSL=ln 1] [,DDGA=gc1][,A]

2. 说明

NEWDOS/80 已经限定用 5 英寸磁盘驱动器和 8 英寸磁盘驱动器工作。PDRIVE 命令告诉NEWDOS/80各个驱动器的特征说明。

(1)PDRIVE 命令可以列出 10 个驱动器的特征说明。然而，能够进行输入/输出的实际驱动器数由 SYSTEM 的选用参数 AL 所限定，并且不得超过 4 个。由 AL 规定的驱动器在 PDRIVE 显示清单中用驱动器号后面的星号标记出来，这里给出了一个实际使用的 NEWDOS/80 2.0盘片上的 PDRIVE 特征说明，在清单上可以看到0,1,2和 3 驱动器标有星号，它们是可以进行输入/输出操作的实际驱动器。这 10 个驱动器说明被保留在装配于 dn1 驱动器内的系统盘片上。

```
0*  TI=A,   TD=A, TC=80, SPT=10,  TSR=3,  GPL=2,  DDSL=17,  DDGA=2
1*  TI=A,   TD=A, TC=80, SPT=10,  TSR=3,  GPL=2,  DDSL=17,  DDGA=2
2*  TI=A,   TD=A, TC=35, SPT=10,  TSR=3,  GPL=2,  DDSL=17,  DDGA=2
3*  TI=A,   TD=A, TC=35, SPT=10,  TSR=3,  GPL=2,  DDSL=17,  DDGA=2
4   TI=A,   TD=A, TC=35, SPT=10,  TSR=2,  GPL=2,  DDSL=17,  DDGA=2
5   TI=CK,  TD=E, TC=34, SPT=18,  TSR=2,  GPL=2,  DDSL=17,  DDGA=2
6   TI=CK,  TD=E, TC=39, SPT=18,  TSR=3,  GPL=2,  DDSL=17,  DDGA=2
```

7 TI=A, TD=C, TC=80, SPT=20, TSR=2, GPL=2, DDSL=17, DDGA=2
8 TI=C, TD=E, TC=40, SPT=18, TSR=3, GPL=2, DDSL=17, DDGA=2
9 TI=C, TD=G, TC=80, SPT=36, TSR=3, GPL=8, DDSL=17, DDGA=2

为了提高效率，DOS 通常从存放在主存储器内的一个表格中去取用驱动器特征说明。这个主存储器的 PDRIVE 表包含有 1 到 4 个驱动器的特征说明（具体数量由 SYSTEM 的选用参数 AL 值决定），在打开电源和复位时，只有 10 个驱动器特征说明都无错误，才能自动从 0 号驱动器的盘片上把这个表格重新装入主存储器（否则，复位就会使系统死锁）。使用带有 A 参数的 PDRIVE 命令也能立即重新装入这个表格。

（2）驱动器 dn1 是装有系统盘片的驱动器，我们正准备修改的就是这个系统盘片上的控制信息（在第三扇区）。驱动器 dn2 是驱动器 dn1 上系统盘片控制信息扇区所记载的 10 个驱动器特征说明中的一个，正准备修改的就是这一个驱动器的控制信息。例如，如果给出的 PDRIVE 命令是 PDRIVE, 1, 4, TC=80，则要读装于 1 号驱动器上的盘片（这是一个系统盘片），以取得 PDRIVE 控制信息，并对它进行修改，组成新的 4 号驱动器特征说明。1 号驱动器内盘片上的 PDRIVE 控制信息包含有 10 个驱动器特征说明，dn2 的取值范围是 0 到9，在本例中 dn2=4，所以要改变的是第五个驱动器（即 4 号驱动器）的信息。而另外 9 个驱动器的特征说明不改变。

（3）如果使用了 password 1（通行字1），则命令中必须给出 password 1，而且 password 1 就是驱动器 dn1 上盘片的主通行字（关于通行字的说明，请看5.11节）。否则，该命令中的 password 1 可以省去。

（4）只有相应于命令中指定参数的那些控制数据要改变，未指定参数的相应控制数据不改变。如果显示出盘片上有错误，则在复位或打开电源时用 dn1 内的盘片作系统盘片是不能启动系统的，必须把错误纠正以后才能用。

（5）打入命令 PDRIVE，dn1 将会在荧光屏上列出装配在驱动器 dn1 内的系统盘片上控制数据中的全部 PDRIVE 特征说明（共有 10 个）。

（6）下面介绍 PDRIVE 命令中的各种参数：

＊dn1 和 dn2 是磁盘驱动器编号，如果在命令中指定了除 A 以外的任一参数，则必须确定 dn2，并把 dn2 放在 dn1 的后面，作为 dn1 后的第一个参数。

＊ dn2=dn3 使驱动器 dn2 采用驱动器 dn 3 所有的 PDRIVE 特征说明。在说明其它参数以前，必须先要使 dn2=dn3。

＊ TI=type 1 确定磁盘驱动器接口的型式。type1 由下面列出的一个或多个选用字母标志组成。对于 TRS-80 Ⅰ型，必须选用标志 A,B,C 或 E 中的一个，而且只能选用一个。对于Ⅲ型来说，必须在标志 A 和 D 中选一个，也只能选一个。另外一些标志是根据接口来选用的。有些标志在驱动器内是互相排斥的，这就是说对于一个给定的驱动器 dn1（内装系统盘片），如果其中对 dn2 规定的一个标志与另一个标志之间是互斥的，则不能再为 dn2 规定另一个与之互斥的参数。对 TRS-80 Ⅰ型来说，标志 B, C 和 E 是驱动器内互斥的参数。下面说明 TI 中各个字母标志的意义。

① 标志 A 表示用于驱动器磁盘 I/O 的是一个标准的接口。这个接口适用于 TRS-80 Ⅰ型中 TD 参数规定的 A 型和 C 型驱动器，对于 TRS-80 Ⅲ型可以支持 A 型，C 型和 G 型驱动器。

② 标志 B （只适用于 TRS-80 Ⅰ型)表示安装的是 OMIKRON 型接口。这个接口适用于 A 型，B 型，C 型和 D 型规格的驱动器（见 TD 参数的说明)。

③ 标志 C （只适用于 TRS-80 Ⅰ型）表示安装的是 PERCOM 型接口，并用作它的 I/O。这个接口支持 A 型，C 型，E 型和 G 型驱动器。

④ 标志 D （只适用于 TRS-80 Ⅲ型）表示安装的是 Apparat 的磁盘控制器。它可以支持 A 型到 H 型的各种驱动器（其中 F 型和 H 型驱动器要求Ⅲ型主机采用高速的CPU)。

⑤ 标志 E （只适用于 TRS-80 Ⅰ型）表示安装的是一个 LNW 型接口。这个接口支持 A 型到 H 型的各种驱动器。

⑥ 标志 H 表示每当 DOS 从另外一个驱动器转入到这个驱动器时，需要给出磁头稳定延迟时间。对于 TRS-80 Ⅰ型和Ⅲ型的 5 英寸驱动器，当马达运转时所有的 5 英寸磁盘驱动器的磁头都已经处于工作状态，所以不需要这种附加的时间延迟。但是 8 英寸驱动器就需要设定标志 H。

⑦ 标志 I 表示每个磁道的编号最小的扇区是 1 扇区。这是 TRS-80 Ⅲ型 TRSDOS 磁盘的正常情况。如果不规定标志 I，就假定每个磁道的最小扇区编号是 0，这是 TRS-80 Ⅰ型和Ⅲ型 NEWDOS/80中规定的标准情况。

⑧ 标志 J 表示磁道编号从 1 开始。如果不规定标志 J，就假定磁道编号从 0 开始，这是 TRS-80 Ⅰ型和Ⅲ型的标准状态。

⑨ 标志 K 表示以不同于该盘片其它磁道的密度去格式化（或要格式化）磁道 0。这就使磁道 0 不能用在它原来通常的工作环境。K 标志可以隐含设定 J 标志。把磁道 0 格式化成不同于其它磁道密度的目的是使得 TRS-80 Ⅰ型可以引导双密度的系统盘片，或者使 TRS-80 Ⅲ型可以引导单密度的系统盘片。TRS-80 Ⅰ型的 ROM 已设计成只能读单密度引导扇区，而Ⅲ型的 ROM 设计成只能读双密度引导扇区。设定标志 K 以后，使得FORMAT 命令和具有格式化能力的 COPY 命令把磁道 0 格式化为不同于其它磁道的另一种记录密度，并把需要的引导扇区放在这个磁道上以供 ROM 使用。因为设定了标志 K，所以通常对磁道 0 的 DOS I/O 实际上移到了磁道 1，而磁道 1 移到了磁道 2，依次类推。要读出 NEWDOS/80 1.0 版本或其它 DOS （不包括 NEWDOS/80 2.0 版本及以后更新的版本)管理下由PERCOM 型接口所建立的双密度盘片，则必须为该驱动器指定 K 标志。在 NEWDOS/80 2.0 版本的Ⅰ型系统中，如果双密度数据盘片始终不在 0 号驱动器上使用，就没有必要把磁道 0 留作单密度用。对于标准的Ⅲ型盘片不应该规定标志 K，除非因某种原因，用户希望在 TRS-80 Ⅲ型机上使用一个单密度系统盘片，或者制作一个在Ⅰ型机上可读的双密度盘片（不带 NEWDOS/80 2.0 系统)。使用 K 标志后，设定 TC 参数时，必须比不用 K 标志时应该设定的磁道数减少 1。由于保存在盘片上的邻接扇区的次序不同，所以由修补过的 NEWDOS/80 1.0 版本建立的双面双密度盘片，在 NEWDOS/80 2.0 管理的系统中就无法正确读出。为了使这种规格盘片上的文件能由 2.0 版本读出，首先必须在 1.0 版本的管理下把这些文件传送到一个单面（单、双密度都可以）或双面单密度的盘片上，然后才能由 2.0 版本读出。

⑩ 标志 L 表示磁道之间有两个步进脉冲。这就使一个 35 或 40 磁道的盘片可以在80 磁道的驱动器上读出来。也能按这种方法来写入，但是 35 或 40 磁道的驱动器在读(写)某些扇区时可能会发生故障，所以不推荐用户使用这种方法。

⑰ 标志 M 表示盘片是一个标准的 TRS-80 Ⅲ型 TRSDOS 规格的盘片。标志 M 隐含

I 标志。COPY 命令是 NEWDOS/80 中注意到Ⅲ型 TRSDOS 盘片不同于 NEWDOS/80 盘片的仅有的一个命令，而且只有设定了标志 M 以后才能有这样的功能。

标志 F—G, N—Z 将留待以后定义。

＊TD是驱动器类型的说明。具体定义如下：

① TD＝A　　5 英寸单密度，单面驱动器。
② TD＝B　　8 英寸单密度，单面驱动器。
③ TD＝C　　5 英寸单密度，双面驱动器。
④ TD＝D　　8 英寸单密度，双面驱动器。
⑤ TD＝E　　5 英寸双密度，单面驱动器。
⑥ TD＝F　　8 英寸双密度，单面驱动器。
⑦ TD＝G　　5 英寸双密度，双面驱动器。
⑧ TD＝H　　8 英寸双密度，双面驱动器。

如果在计算机中用了比原来速度快的 CPU 模块，在磁盘 I/O 期间就必须把 CPU 速度恢复到原来的速度，并且不得低于原来的值。NEWDOS/80 的磁盘 I/O 程序(特别是对 TRS-80 I 型的 B, D, E 和 G 型驱动器) 不允许把 CPU 的速度降低到原来的速度以下。只要你耐心地多作几次试验和适当地设置 SYSTEM 参数 BJ, 就可以使 NEWDOS/80 2.0 在不降低加速的 CPU 模块速度的情况下，成功地进行磁盘 I/O，但是 Apparat 公司并不对这种性能作出任何保证。

使用 TD＝H 和 TD＝F 参数，需要在计算机中安装加速的 CPU 模块，在磁盘 I/O 期间，至少使 CPU 的速度加快一倍。

对于 C,D,G 和 H 型等双面驱动器，现有的 NEWDOS/80 接口 (TI 标志 A,B,C,D 或E) 把双面盘片看作为单容积 (即仅有一个目录),在第一面上有低编号的扇区，而第二面上有高编号的扇区。联接驱动器电缆的第 32 引出端用于选择第二面 (需要用专门的电缆)，接在这种专用电缆上的任何一个驱动器必须把这种作为第二面选择用的分路线切断，以防选中另一个驱动器第二面时把这个驱动器也选中。I 型和Ⅲ型的 NEWDOS/80 2.0 适用于双面，双密度的 40 和 80 磁道的驱动器。

＊TC＝tc 1　规定盘片上的磁道数目。如果设定了参数 TI 的标志 K, tc 1就不应该包括磁道0。例如，不设定标志 K,35 磁道的驱动器的 TC＝35, 40磁道驱动器的 TC＝40。如果设定了标志 K, 35 磁道驱动器的 TC＝34, 40 磁道驱动器的 TC＝39。

＊ SPT＝sc 1　规定每个磁道的扇区数目。对于双面，单容积盘片(TD＝C,D,G或H)，sc 1 必须是相应单面盘片的两倍。sc 1 取值范围在 1 到一个最大值之间，这个最大值由磁道可容纳的最多扇区数来决定 (每个扇区包含 256 个字节)。对于上面所定义的各种型号驱动器，每个磁道的最多扇区数分别为：A＝10, B＝17, C＝20, D＝34, E＝18, F＝26, G＝36 和 H＝52。

＊ TSR＝rc 1　规定表示磁道步进脉冲时间的代码，rc 1是给驱动器的控制器用的。rc 1 取值范围在 0 到3，它是发送到控制器去的 SEEK, STEP 和 RESTORE 命令的一个组成部分，TSR＝0给出 5 毫秒的步进脉冲，TSR＝1给出10毫秒的步进脉冲，TSR＝2给出20毫秒的步进脉冲和 TSR＝3 给出 40 毫秒的步进脉冲。TRS-80 I 型原来的标准是 TSR＝3, 对于某些驱动器，用户应该用 TSR＝2 或 TSR＝1。在 TRS-80Ⅲ型中用 TSR＝0 作为标准。如果你的驱

动器有些故障，最安全可靠的步长是 TRS=3 (TRS-80 Ⅰ型最快的步进速率是12毫秒)。

* GPL=gc2 规定每个 gr. 组的 gr. 数，gc2 可以取 2 到 8 之间的一个值。TRS-80 Ⅰ型和Ⅲ型的 TRSDOS 和老版本的 NEWDOS 中，磁盘空间的分配是按 gr.(Ⅰ型中，每个 gr. 包含 5 个扇区，Ⅲ型中，每个 gr. 包含 3 个扇区) 和磁道(Ⅰ型中，每个磁道由 2 个 gr. 组成，而Ⅲ型中，每个磁道由 6 个 gr. 组成) 进行的。在 NEWDOS/80 2.0 版本中，不论是 TRS-80 Ⅰ型还是Ⅲ型，每个 gr. 仍然是由 5 个扇区组成，每个 gr. 组(不是磁道)有 2 到 8 个gr.。现在目录中(GAT 扇区和 FDE 的两字节范围元)的磁道数已由 gr. 组数来代替。这样做以后，一个 gr. 可以起始于一个磁道而终止于另一个磁道，并使双密度和 8 英寸盘片每磁道的扇区数目为最大，而且保持相同的目录格式。取 GPL=2 可以保持与老的 35 磁道单密度盘片兼容，因为目录严格保持一致，所以 TRS-80 Ⅰ型的 TRSDOS 和 NEWDOS/80 2.0 以前的 NEWDOS 之间可以相互传送数据。现在，如果取 GPL=8，目录就能适应 192×8×5=7680 扇区或 1966000 字节容量的盘片。

* DDSL=ln 1 是 DDST 参数的逻辑等值，并代替了 NEWDOS/80 1.0 中的 DDST 参数。ln 1 规定了一个 gr. 组的编号，这个 gr. 组的第一个扇区包含有目录的第一个扇区。表示目录第一个扇区的值在盘片格式化时存贮在引导扇区的第三个字节，寻找目录时要用到它。在盘片格式化时，要根据它的值去确定目录放在什么地方。在比较老的系统中，引导扇区的第三个字节包含有磁道编号，该磁道的第一个扇区是目录的开始。因为 NEWDOS/80 2.0在空间分配和管理中不用磁道，所以，现在引导扇区的第三个字节放有 gr. 组的编号，该 gr. 组的第一个扇区就是目录的开始。为了确定目录第一个扇区 (GAT 扇区) 的相对扇区编号，就要取引导扇区第三个字节的内容，并把它与 5 倍的 GPL 值相乘。当 DDSL=17 时，就可以保持与标准的 35 磁道、单面、单密度的盘片兼容。应该把 DDSL 设定为 NEWDOS/80 1.0 的 DDST 值。

* DDGA=gc 1 规定了格式化期间建立目录时分配给目录的缺项 gr. 数。gc 1 的取值范围是 2—6 。为了与标准的 35 磁道、单面、单密度盘片兼容，应该规定 DDGA=2。当 gc1>2 时，在一个数据盘片上可以容纳的文件多于 62 个，最多可以达到 222 个文件。

* A 规定了唯有在检验所有驱动器特征说明期间没有发现错误时，SYSTEM 参数 AL 所指定数目的驱动器特征说明才被装进主存的 PDRIVE 表，并立即成为这些驱动器的控制数据，这时不再需要复位。如果规定了参数 A, dn 1 就必须等于0。

(7)PDRIVE 命令不能在 MINI-DOS 状态使用。

3. 举例

(1)PDRIVE, dn1, dn2, TI=A, TD=A, TC=35, SPT=10, TSR=3, GPL=2, DDSL=17, DDGA=2

上面给出的是一个标准的 5 英寸，35 磁道，单密度，单面盘片的 PDRIVE 特征说明，可以在 TRS-80 Ⅰ型中使用。这个说明也适用于Ⅲ型，它为读这个盘片提供了正确的目录地址标记 (见 SYSTEM 参数 AN)。

(2)PDRIVE, dn1, dn2, TI=A, TD=E, TC=40, SPT=18, TSR=3, GPL=2, DDSL=17, DDGA=2

上面是标准的 5 英寸，40磁道，双密度，单面盘片的 TRS-80 Ⅲ型的 PDRIVE 特征说明 (Ⅰ型用 TI=C)，在Ⅲ型的 NEWDOS/80 系统内通用。用这个说明后，如果在扩展接口

中作了双密度修正，这个盘片也能在Ⅰ型中非0号驱动器中读出。

(3)PDRIVE, dn1, dn2, TI=AM, TD=E, TC=40, SPT=18, TSR=3, GPL=6, DDSL=17, DDGA=2

这是用于读出或写入到一个Ⅲ型 TRSDOS 标准的5英寸，双密度，单面盘片的 PDRIVE 特征说明（Ⅰ型用 TI=CM 或 EM)。40磁盘，双密度，单面5英寸磁盘是 NEWDOS/80能管理的仅有的一种Ⅲ型 TRSDOS 盘片。必须规定 GPL=6。因为Ⅲ型 TRSDOS 盘片不能由 NEWDOS/80 格式化，所以 DDSL 和 DDGA 参数是无意义的。在 NEWDOS/80 中（对于 TRS-80 Ⅰ型必须安装双密度部件），除了能够使用 SUPERZAP 程序的 DD,DM,DTS,VDS, CDS,CDD 等功能（不作为文件来处理，也不用 DFS 功能）来处理这个Ⅲ型 TRSDOS 盘片以外，DOS 命令中只有 COPY 命令可以使用。

(4)PDRIVE, dn1, dn2, TI=A, TD=C, TC=80, SPT=20, TSR=2, GPL=8, DDSL=20, DDGA=6

这是一个5英寸，80 磁道，单密度、双面、单容积磁盘的说明，该盘片使用20毫秒的步进脉冲，每个 gr. 组有8个 gr.，并且目录放置在盘片的中央，目录具有最大的容量。在 TRS-80 Ⅲ型中，单密度0号驱动器使用要受到限制。

(5)PDRIVE, dn1, dn2, TI=A, TD=G, TC=80, SPT=36, TSR=2, GPL=8, DDSL=35, DDGA=6

这是一个5英寸，80磁道，双密度，双面，单容积Ⅲ型的磁盘说明（Ⅰ型用 TI=C 或 E)，它使用20毫秒的步进脉冲，每个 gr. 组有8个 gr.，最大容积的目录放置在磁盘的中央位置。在Ⅰ型中，双密度0号驱动器的使用要受到限制。

(6)PDRIVE, dn1, dn2, TI=CK, TD=E, TC=39, SPT=18, TSR=3, GPL=2, DDSL=17, DDGA=2

这是5英寸，40磁道，双密度，单面的 TRS-80 Ⅰ型的磁盘说明（Ⅲ型用 TI=AK)，盘片的0磁道按单密度格式化，因此只有39个磁道可以使用。这个磁盘说明适用于 PERCOM 接口，运行由 TRSDOS 和 NEWDOS/80 1.0 格式化的双密度盘片。本例还可用于制作一个双密度盘片，作为Ⅰ型0号驱动器上的系统盘片。对于Ⅰ型的 LNW 接口要用 TI=EK。

(7)PDRIVE, dn1, dn2, TI=CK, TD=G, TC=79, SPT=36, TSR=3,GPL=8, DDSL=35, DDGA=6

这是5英寸，80磁道，双密度，双面，单容积 TRS-80 Ⅰ型的磁盘说明（Ⅲ型用 TI=AK),该盘片的磁道0格式化成单密度。对于 LNW 接口要使用 TI=EK。

请注意!!! 在修补后的 NEWDOS/80 1.0 中用的双面，双密度盘片不能在NEWDOS/80 2.0 中使用（见 TI 标记 K 的说明)。

(8)PDRIVE, dn1, dn2, TI=AL, TD=A, TC=35, SPT=10, TSR=3, GPL=2, DDSL=17, DDGA=2

这是5英寸，35磁道，单面，单密度磁盘说明，但是该盘片能在80磁道的驱动器上读出。对于80磁道驱动器的每个数据磁道来说，每步仅为35和40磁道的一半，设置标志L使得每步进一个数据磁道要两个步进脉冲。

(9)PDRIVE, dn1, dn2, TI=BH, TD=B, TC=77, SPT=17, TSR=3, GPL=3, DDSL=17, DDGA=6

这是8英寸，77磁道，单面，单密度 TRS-80 I 型的磁盘说明。注意，NEWDOS/80 1.0 使用 SPT=15 和隐含的 GPL=3，因此，为了使得 NEWDOS/80 1.0 能读这些磁盘，必须使用 SPT=15 和 GPL=3。为了提高利用率，建议用 COPY 的方法，把那些 SPT=15 的盘片变换成 SPT=17，这样可以增加12%的磁盘空间。标志 H 使得有磁头加载稳定延迟时间，大多数8英寸驱动器都需要有这种延迟。

(10)PDRIVE, dn1, dn2, TI=BH, TD=D, TC=77, SPT=34, TSR=3, GPL=8, DDSL=17, DDGA=6

这是8英寸，77磁道，单密度，双面，单容积 TRS-80 I 型的磁盘说明，并具有磁头加载稳定延迟时间。

(11)PDRIVE, dn1, dn2=dn3

把驱动器 dn3 的特征说明作为驱动器 dn2 的特征说明。

(12)PDRIVE, dn1, dn2=dn3, TC=40, TSR=2

把驱动器 dn3 的特征说明作为驱动器 dn2 的特征说明，然后使用新的 TC 和 TSR 值。

(13)PDRIVE, 0, A

只有当全部磁盘说明没有错误时，才把由 SYSTEM 选用参数 AL 规定数目的驱动器的 PDRIVE 数据装进主存的 PDRIVE 表。

(14)PDRIVE, 0, dn2=dn3, A

读装在0号驱动器上的磁盘的驱动器特征说明，把 dn2 的特征说明改成 dn3 的特征说明，然后完成上例规定的任务。

2.38 PRINT　在打印机上打印文本文件

1. 命令格式

PRINT, filespec 1 [,start-line　[, line-count]]

2. 说明

PRINT 命令执行与 LIST 命令相同的任务，其差别是 PRINT 命令把清单送到打印机，而不是送到显示器。参数说明和举例请看 LIST 命令。

2.39 PROT　改变磁盘的某些控制数据

1. 命令格式

PROT, [password 1:] dn 1 [,NAME=name 1 [,DATE=mm/dd/yy] [,RUF] [,PW=password 2] [,LOCK] [,UNLOCK]

2. 说明

PROT 命令中至少要规定一个选用参数。各个参数的说明如下：

* dn 1（驱动器编号）　目标磁盘就装在这个驱动器内。

* password 1（通行字1）如果通行字有效，则必须规定　password 1，并使它等于该盘片的主通行字（关于通行字的说明请看5.11节）。

* NAME=name 1（名字1）　这是给盘片定的一个名字。

* DATE=mm/dd/yy（月/日/年）　给盘片规定一个日期。

* RUF　复位修改标志。这个参数关闭该盘片上所有文件的修改标志。如果一个用户仅

对那些修改标志打开（参看 COPY 命令的 UPD 参数）的文件制作备份，则在复制完成以后，执行带有 RUF 参数的 PROT 命令，就能关闭修改标志，在接着的备份制作中这些文件在再次被修改之前就成为不能复制的文件，不能制作备份，只要写或重写文件的一个扇区（不管实际上有否改变）就会使 DOS 去打开文件修改标志。

* PW=password 2（通行字 2） password 2 必须符合通行字规则，命令中不写 password 2，就认为它全部是空格。磁盘把 password 2 作为它的通行字。

* LOCK 这个参数使盘片上的所有文件（不包括系统文件和隐性文件）都以盘片的主通行字作为它的存取通行字和修改通行字。如果给定了 password 2，就用它作为通行字。这个特性在通行字有效的系统中，当用户忘记文件的通行字时，为用户提供了存取文件的一种方法，这时用户只要知道磁盘的主通行字就可以了。该方法的缺点是要改变磁盘上所有文件的通行字（不包括系统文件和隐性文件）。所以用户要重新为所有的文件，也包括用户忘记通行字的那个文件分配通行字。还有一种比较方便的办法：用户只要知道一个 NEWDOS/80 系统盘片的通行字，最好是有一个通行字无效（即 SYSTEM 的参数 AA=N）的 NEWDOS/80系统盘片。把这个系统盘片放入 0 号驱动器内，因为通行字无效，用户可以用 ATTRIB 命令直接为忘记通行字的文件重新分配一个新的通行字，这样就不会影响盘片上其它的用户文件。上述方法可以恢复忘记通行字的文件的通行字功能。

* UNLOCK 该盘片的所有文件（除系统文件和隐性文件）的存取和修改通行字都设定为全部空格，即这些文件均没有通行字。

3. 举例

（1）PROT, 2, RUF

清除现在装配于 2 号驱动器内的盘片上的每一个文件的修改标志。

（2）PROT, OLDPSWD:1, NAME=DSDC, DATE=05/01/84, PW=NEWPSWD

在这个例子中，通行字是有效的，所以需要在命令中放入磁盘的主通行字 OLDPSWD。本例要改变装配于 1 号驱动器内盘片的控制信息，其名字改为 DSDC，日期改为1984年 5 月 1 日，新的磁盘主通行字是 NEWPSWD。

2.40 PURGE 有选择地删去磁盘上的一些文件

1. 命令格式

PURGE, [password 1:] dn1 [,/ext] [,USR]

2. 说明

（1）这个命令只对装配于 dn1 驱动器内的磁盘有效。如果系统中通行字有效，必须在命令中指定 password 1，并使它等于磁盘的主通行字。

（2）对盘片上除 BOOT/SYS 和 DIR/SYS 以外的每一个文件，DOS 要询问用户该文件是否要删去。如果要删去该文件，则打入一个 Y 作回答，这时文件立即被删去，就像发出一个 KILL 命令一样。如果该文件不要删去，就回答一个 N。如果要退出 PURGE 命令，只要回答 Q 就可以了。这个命令对于要同时删去多个文件的用户特别有用。

* /ext 如果指定了这一个选用参数，则 PURGE 命令就限定于删去那些带有文件名扩展符 ext 的文件，其中 ext 是 0 到 3 个字符组成的文件名扩展符。

* USR 如果规定了这一个选用参数，则系统文件和隐性文件不包括在 PURGE 命令

内。

3. 举例

(1)PURGE, 1

对于目前装配在1号驱动器的盘片上的每一个文件（除BOOT/SYS和DIR/SYS），DOS都要询问是否要删除。如果用Y回答，则该文件就会被删除。

(2)PURGE, 0, /DAT

对于目前装配在0号驱动器上的盘片的每一个带有文件名扩展DAT的文件，DOS都要提出询问，询问该文件是否要删除，如果回答Y，就会把该文件删除。

(3)PURGE, 0, USR

对于目前装配在0号驱动器内盘片上的每个非系统和非隐性的文件，DOS都要提出询问，询问是否要删除，如果回答Y，就会把该文件删除。

2.41 R 重复前面的DOS命令

说明

(1)这个命令用来重复执行前面的一个DOS命令（不包括R命令）。

例如：DIR 1 接着使用R命令，其效果就像执行下面的两个DOS命令一样。

```
DIR 1
DIR 1
```

(2)R命令不能在BASIC中经CMD "doscmd"来执行，因为CMD "doscmd"要求命令必须由2个或2个以上的字符组成。

(3)R命令没有参数，而且必须严格的按照规定在R之后打入ENTER键。如果在R之后还把其它字符（即多于2个字符）打入缓冲器中，然后用退格键把多余的字符删去，于是DOS只能看到R和ENTER，在R命令之前存放在命令缓冲器中的DOS命令必然会改变，这时R命令就会失败，极偶然的情况会出现用户预想不到的结果。

如果前面要被重复执行的DOS命令在DOS命令缓冲器中不再是完整的，则R命令的结果是不可预测的。

(4)如果SYSTEM的选用参数BE=N，则R命令不执行前一个DOS命令，而是返回到DOS READY状态。

2.42 RENAME 更改一个文件的名字

1. 命令格式

```
RENAME, filespec 1 [,TO], filespec 2
```

2. 说明

把文件的filespec 1改名为filespec 2，其中filespec 2只包含有一个文件名和一个文件名扩展符。如果filespec 1没有规定驱动器编号，则搜索所有已装配好的磁盘，并把第一个遇到的，与filespec 1的文件名和文件名扩展符一致的文件改名。RENAME命令只改变文件名和文件名扩展符，不改变文件的其它内容。

3. 举例

(1)RENAME XXX/DAT:1 YYY/OBJ

把目前装配在1号驱动器内磁盘上的文件XXX/DAT的名字改为YYY,文件名扩展符改为OBJ。

2.43 ROUTE 选择数据传送的路径

1. 命令格式

这个命令有下列三种格式:

(1)ROUTE

(2)ROUTE, CLEAR

(3)ROUTE, dev 1 [,dev 2][,dev 3]...

2. 说明

设立ROUTE命令的目的是为了增加使用的灵活性,可以选择从键盘和(或)RS-232输入口接收数据,并发送到显示器,打印机和RS-232输出口。

(1)ROUTE命令结束时会显示所用的传送通路,如果不存在这种传送通路,则什么信息也不显示。不带参数的ROUTE命令只是把已经存在的传送通路显示出来,此外不执行任何其它的操作。

(2)ROUTE, CLEAR 清除全部传送通路。

(3)ROUTE, dev 1 [,dev 2] [,dev 3] ...

dev 1指定了要改变传送通路的设备。当dev 1是一个输出设备时,dev 2,dev 3,…等参数规定了传送到达的接收设备,而当dev 1是一个输入设备时,dev 2, dev 3, … 表示发送数据的设备。对于TRS-80 I型来说,键盘用的设备代码是KB,显示器的设备代码是DO,打印机的设备代码是PR,空缺代码是NL(表示没有东西要传送)。在TRS-80 Ⅲ型中,增加了RS-232接口,其输入代码是RI,输出代码是RO。一个输入设备(KB或RI)不可以替换一个输出设备(DO, PR, RO),同样,一个输出设备也不能替换一个输入设备。

只要在命令中指定了dev 1, ROUTE命令首先要清除这个设备原来存在的通路,如果有dev 2, dev 3, ……等参数,就建立由这些参数指定的通路。

dev 2, dev 3等任何一个设备也可以采用MM=addr的形式,其中addr规定了要代替dev 1的用户子程序的内存储器单元。该子程序最前面的12个字节是供DOS用的,不允许用户去改动它,所以实际上addr指的是用户子程序的第13个字节的位置。用户要做的工作就是保存和恢复子程序用的除AF以外的所有寄存器和它调用的子程序。如果dev1是一个输入设备,子程序就在A寄存器中送回一个新的字节,如果送回的是一个0,就表示子程序没有送回新的输入字节。如果dev 1是一个输出设备,就要送字节到子程序中去,这时C寄存器内就放有要输出的字节。

如果dev 1是一个输出设备,则把输出字节依ROUTE命令中的次序发送到通路上所有的接收设备。

如果dev 1是一个输入设备,则要按ROUTE命令中的次序询问通路上的每个发送设备。如果有一个设备提供一个非零字节,就停止询问,并把该字节用作dev 1的输入字节。如果没有设备提供输入字节,就认为0是dev 1的当前输入字节。

(4)通路指向和开始的设备(不包括MM=addr型式)有一定的限制,目前对于TRS-

80 Ⅰ型一次可以有4个，而Ⅲ型一次可以有6个。

警告!!!　在传送路径畅通期间，不得编辑输入或输出字节。因为这样做可能会发生问题（即显示器的控制字符会使得打印机打出不可理解的东西）。

3. 举例

(1)ROUTE, PR, DO

打印机输出不送到打印机去打印，而是改送到显示器去显示。

(2)ROUTE, DO, DO, PR

显示器输出同时送到显示器和打印机。

(3)ROUTE, PR, DO, PR

打印机输出同时送到显示器和打印机，例2和例3的操作和TRS-80 Ⅲ型 TRSDOS的DUAL命令的功能是一样的。

(4)ROUTE, KB, RI（仅适用于TRS-80 Ⅲ型）

键盘输入字符来自RS-232输入设备，而不是来自于键盘的输入。

(5)ROUTE, DO, RO（仅适用于TRS-80 Ⅲ型）

显示输出发送到RS-232输出设备，而不是送到显示器显示。

(6)ROUTE, PR, MM=0FE80H

打印机输出送到内存单元FE80H的子程序（子程序的实际入口是FE8CH,前面的12个字节是留给DOS用的）。

(7)ROUTE, KB, KB, MM=0F800H

键盘输入来自于键盘或位于内存单元F800H的子程序。键盘来的输入有较高的优先级。

(8)ROUTE, PR, NL

不要打印机输出。

(9)ROUTE, PR

撤消所有的打印机通路。打印机输出还是送到打印机。

(10)ROUTE, CLEAR

撤消所有用户规定的通路，并且把所有的设备恢复到它们标准的正常路径。

2.44 SETCOM　设置RS-232接口的参数（只适用于TRS-80 Ⅲ型）

1. 命令格式

SETCOM [,OFF] [,WORD=w1] [,BAUD=br] [,STOP=sb] [,PARITY=pp] [,WAIT] [,NOWAIT]

2. 说明

(1)SETCOM命令可以随意改变RS-232接口的状态和显示这种状态。关于RS-232接口的讨论请看TRS-80 Ⅲ型的使用说明书及BASIC语言参考手册。SETCOM命令只影响标准的RS-232接口控制块和子程序。

(2)SETCOM命令中各项选用参数的说明：

* OFF　如果指定了OFF参数，就会关闭RS-232接口功能。这时不可再规定该命令中的其它选用参数。

如果 WORD, BAUD, STOP 及 PARITY 诸参数中，不论那一个，只要不指定新的值，就表示不改变原来设定的值。

* WORD=w 1 规定每个传送字节的位数。w 1可以取5，6，7或8。

* BAUD=br 规定发送和接收的传送速率(波特率)。允许有 16 种波特率，分别是50，75，110，134，150，300，600，1200，1800,2000，2400，3600,4800，7200，9600和19200波特。

* STOP=sb 规定每个传送字节所用的停止位数。sb 可以取 1 和 2 。

* PARITY=pp 规定传送中使用的奇偶校验情况。当 pp=1 时，采用奇校验,pp=2是偶校验，而 pp=3 表示不用奇偶校验。

* WAIT 或 NOWAIT 这两个参数是互斥的（即不能同时用这两个参数,也就是说用了其中的一个，就不能再用另外的一个)，它可以规定 RS-232 输入子程序是否一直要等待到接收到一个输入字节时才为止，以及输出子程序是否要一直等待到当前的字节发送出去为止。如果命令中没有指定这个参数，就以上次设定的参数为准。

3. 举例

(1)SECTOM, WORD=8, BAUD=300, STOP=1, PARITY=1, WAIT

如果原来没有启动 RS-232 接口，接口就处于不工作状态，执行本例的命令以后，RS-232 接口就可以工作了，并把接口设定为每个传送字节包含有 8 位，300 波特传送率，1个停止位和采用奇校验。在调用 RS-232 子程序时，要使它具有等待功能，也就是说，要等待输入字节接收就绪，或者输出字节送到输出设备时为止。

(2)SETCOM, NOWAIT, PARITY=3, WORD=7

如果原来没有启动 RS-232 接口，现在要使它工作．并设定该接口为每一个字节包括 7 位，没有奇偶校验，RS-232 子程序也没有等待功能。TRS-80 中断子程序将管理 RS-232 接口的输入或输出操作。不改变命令中没有涉及到的其余参数。

(3)SETCOM, OFF

使 RS-232 接口不工作。目前接口所有的参数都保留不变。

2.45 STMT 显示指定的信息

1. 命令格式

STMT, msg

2. 说明

因为通常在打入 DOS 命令的时候，总是会显示出相应信息的，故而这个命令平时没有什么用。但是，如果通过 DOS-CALL 引用这个命令 (DOS-CALL 是不能在荧光屏上显示 DOS 命令的),就可以在调用过程中显示出 msg 信息。

STMT 命令是链接状态中显示信息而不暂停的三种方法中的一种。要显示多行指示信息，最后一行用 PAUSE 命令，其余各行可以用 STMT 命令。

3. 举例

(1)STMT PHASE ONE COMPLETED

上述命令可以在显示器上显示信息 PHASE ONE COMPLETED (已完成阶段1的操作)，这只是告诉终端操作员已完成了一遍操作（不管是干什么事)。DOS并不暂停。

(2)STMT DISMOUNT AND STORE AWAY DISKETTE XXX
PAUSE AND MOUNT DISKETTE YYY ON DRIVE 2.

这个例子是说明 STMT 和 PAUSE 命令组合使用的情况。它可以向用户发出指示：把名字为 XXX 的盘片从驱动器内取出来，并保存好，然后把名字为 YYY 的盘片装到 2 号驱动器内。并等待用户按荧光屏上的指示去完成这些操作。

2.46 SYSTEM 改变系统的选用参数

1. 命令格式

SYSTEM, [password 1:] dn1 [,AA=yn] [,AB=yn] [,AC=yn] [,AD=yn] [,AE=yn] [,AF=yn] [,AG=yn] [,AI=yn] [,AJ=yn] [,AL=al] [,AM=am] [,AN=an] [,AO=ao] [,AP=ap] [,AQ=yn] [,AR=yn] [,AS=yn] [,AT=yn] [,AU=yn] [,AV=av] [,AW=aw] [,AX=ax] [,AY=yn] [,AZ=yn] [,BA=yn] [,BB=yn] [,BC=yn] [,BD=yn [,BE=yn] [,BF=yn] [,BG=yn] [,BH=yn] [,BI=bi] [,BJ=bj] [,BK=yn] [,BM=yn] [,BN=yn]

2. 说明

SYSTEM 命令是选用参数最多的一个命令，它可以用来设定系统的各项参数，作为系统启动时的初始设定值，达到控制系统某些性能的目的。下面详细介绍 SYSTEM 命令及其各项参数的含义。

(1)把要修改（或显示）系统控制信息的 NEWDOS/80 系统盘片装配在驱动器 dn1 内。如果该系统盘片通行字有效，在用 SYSTEM 命令时必须指定 pas word 1（通行字1），并使它等于盘片的主通行字。如果只打入 SYSTEM 命令，不指定任何一个参数，这个命令所起的作用就只是显示这个盘片已有的各项系统参数。如果要指定参数，命令中参数的次序可以由用户自己决定，只有相应于指定参数的控制数据（在该盘片的第三扇区）才改变，SYSTEM 命令中未规定的参数保留原有的值。

(2) SYSTEM 命令的选用参数数量很多，如果要改动的参数很多，就必须多次执行 SYSTEM 命令，这是因为 DOS 缓冲区一次只能容纳 79 个字符。

目前可供选用的参数只有上面命令格式中所规定的那一些，今后将逐步增加。

(3)用 SYSTEM 命令改变系统盘片的一些系统参数以后，并不会影响目前正在工作的计算机的系统状态，只有把改动参数以后的系统盘片插入 0 号驱动器内，并复位计算机以后，这些改动的参数才会影响到计算机的工作。

(4)下面将逐个介绍 SYSTEM 命令的各个选用参数的含义及其功能。

* AA=yn 如果 AA=Y，则系统中通行字有效，如果 AA=N，通行字就无效。

* AB=yn 如果 AB=Y，则系统工作于 RUN-ONLY 方式（只能执行的工作方式）。在这种工作方式中，系统复位就强制设定系统参数 AD=N，AE=N 和 AF=N，而且复位时按下 ENTER 键也不能封锁 AUTO 功能。因而，在这种工作方式中，用户必须有一个合适的 AUTO 命令（见2.4节 AUTO 命令），AUTO 命令将会自动引用一个用户程序或者执行一个 CHAIN 文件(当然最终它仍将引用一个用户程序)。在 RUN-ONLY 方式，如果用户没有用 AUTO 命令设定一个合适的自动运行程序，而是让系统处于 DOS READY 或 MINI-DOS READY 状态，这时荧光屏上就会显示 'RUN ONLY' STOPPED!! KEY 'R' FOR RESET（停止，'RUN ONLY'，要复位可以按 R 键）信息，然后系统进入了无休止的死循环。在系

统接收到一个 R 以后，就会执行 DOS 命令 BOOT（见 2.7 节），重新启动系统。BASIC 语言要保持处于 RUN-ONLY 状态，就要禁止 BREAK 键的功能，而且不能用不带 R 或 V 参数的 LOAD 命令，也不允许用直接语句。如果 AB=N，系统就处于正常的工作方式。

* AC=yn（仅适用于 TRS-80 I 型）如果 AC=Y 和 AJ=Y，NEWDOS/80 内的消除键抖动干扰的子程序就起作用。如果 AC=N 或 AJ=N，就会撤消消除键抖动干扰的子程序，使其不起作用。

* AD=yn 如果 AD=Y，就使联键 'JKL'（即 J,K,L 三个键同时按下）起作用，只要同时按下 J,K,L 键，显示器上的信息就送到打印机去打印。如果 AD=N，就使 'JKL' 联键操作不起作用。

* AE=yn 如果 AE=Y，联键 '123' 起作用，按下 '123' 就可以调用 DEBUG 程序（见 4.1节）。如果 AE=N，则 '123' 联键操作不起作用。

* AF=yn 如果 AF=Y，联键 'DFG' 起作用，按下 'DFG' 可以调用 MINI-DOS（见4.2 节）。如果 AF=N，则 'DFG' 联键操作不起作用。

* AG=yn 如果 AG=Y，就把 BREAK 键看作为一个代码为 01 的通常的输入键。如果 AG=N，就不把它看作为通常的输入键，而把它变成代码为 00 的空键 (null)。在复位或系统返回到 DOS READY 时，就要根据这个参数(AG)来设定 BREAK 键的状态，当然，也可以用 DOS 命令 BREAK 去改变BREAK键的状态，并使状态保持到下一次 DOS READY 出现时为止。另外，程序也可以改变 BREAK 键的状态，其方法是：把 C9H 字节存入 TRS-80 I 型的 4312H 单元（Ⅲ型为4478H单元），就可以把 BREAK 键的代码改成 01H，或者把 C3H 字节放入上述单元，使 BREAK 键的代码成为 00H。

* AH=yn 在 NEWDOS/80 2.0 版本中不必定义这个参数。在老版本中，为了在磁盘 I/O 期间获得较好的时钟精度，要依靠这个参数去关闭时钟中断，新版本的 NEWDOS/80 就不必这样做了。

* AI=yn（仅适用于 TRS-80 I 型）如果 AI=Y，在计算机中已经具有英文小写字母的修正能力，如果 AI=N，就不具备这种能力。用户要知道这个参数的情况，只要测试 436CH单元的位 4 就可以了，如果 AI=Y，则位 4 为 1，如果 AI=N，位 4 就为0。现在，DEBUG 程序和 SUPERZAP 程序要使用这个标志位，以决定存储器的内容是否应该显示成小写字母。

* AJ=yn 如果 AJ=Y，使 NEWDOS/80 的键盘截留子程序可以工作。这个子程序包括重复键功能和防键抖动干扰功能(只有TRS=80 I 型有)，并且有识别 'JKL'，'123'，'DFG' 操作的功能（其它方法要取消定时中断）。如果 AJ=N，则 NEWDOS/80 不去截取 4016H 单元的键盘地址向量，并且：

① 不管 SYSTEM 的参数 AU 如何取值，TRS-80 I 型的重复键功能将不起作用。Ⅲ型恢复了 ROM 内的重复键功能。

② 不管 SYSTEM 的参数 AC 如何设定，将禁止防止键抖动子程序（仅指 I 型）的工作。

③ 'JKL'，'123' 和 'DFG' 等联键操作只能经中断来触发，这样做的结果就会引入许多的假键输入字符（即不是用户有意打入的字符，是联键操作时引入的）。

在整个复位或打开电源期间，保持按下⊡键，就可强制 AJ=N。对于那些可能要覆盖内存储器中 DOS 的程序必须用这种方法强制 AJ=N。

* AK=yn　在NEWDOS/80 2.0中，不必规定这个参数。在老版本的 NEWDOS/80中，这个参数是使 'JKL' 联键操作把图形字符送到打印机去。现在的版本已把这个参数的功能归并到 SYSTEM 的 AX 参数中去了。

* AL=al　al(取值1—4) 规定了系统可用的实际驱动器的个数。如果系统中只有一个驱动器，则设定 al=1，这时系统只限于检查这一个驱动器，虽然 al 可以取值到 255，但是实际上系统最多只能装 4 个驱动器，所以 al 值不得超过4。

* AM=am　am (取值为 0—255,其中 0 表示 256) 是在磁盘 I/O 过程中如果发生错误时，在给出错误信息以前允许进行试验的次数。DOS 中的初始设定值 am=10。

* AN=an　表示 DIR 命令中的缺项驱动器编号,如果 an=0, 在 DIR 命令中不用其它任何的一个参数时，系统就自动显示 0 号驱动器盘片上的文件目录。

* AO=ao　这是一个驱动器编号。当建立一个文件和用户要系统选择文件标识符中没有规定驱动器编号的文件时，系统将首先搜索所有的驱动器，寻找有无该文件。如果没有找到，系统将从编号为 ao 的驱动器开始，向比 ao 编号高的驱动器方向查找，直到找到一个空闲文件目录项 (FDE) 时为止。系统不会去搜索那些比 ao 编号低的驱动器。

* AP=ap　ap 是一个存储器地址,它不能等于0，而且应该位于系统现有的存储器范围内，它将作为 DOS 的 HIMEM 地址值，以两个字节存放在 4049H 和 404AH 单元 (TRS-80 I 型)，在 Ⅲ 型中，存放在 4411H 和 4412H 单元。

* AQ=yn　如果 AQ=Y, 可以用 CLEAR 键去清除荧光屏,如果 AQ=N 和 AJ=Y, 则 CLEAR 键就不能起清除荧光屏的作用。

* AR=yn　如果 AR=Y，即使系统中通行字有效,也不必检查磁盘的主通行字就可以进行复制 (COPY 命令的格式 5 和格式 6)。如果 AR=N, 而且系统中通行字有效，就需要在 COPY 命令中给出通行字，否则不能进行复制。

* AS=yn(仅适用于 TRS-80 I 型)　如果AS=Y,在 BASIC 中将会把输入文本字符串中的英文小写字母转换成大写字母。当计算机中没有安装小写字母的硬件或者没有使用小写字母驱动程序的时候,这个参数是很有用的。因为很可能这时会输入英文小写字母 (即按字母键的同时按 SHIFT 键，就能送出小写字母)，荧光屏上看到的是大写字母，而实际上却是小写字母，但 BASIC 在计算中又把它们看成是不相同的，这很容易发生错误，把 AS 设成 Y 时，就使显示的字母和实际输入计算机的字母统一起来了。如果 AS=N，BASIC 中就不作这种变换。这个参数并不影响作为数据的字符串。

* AT=yn　AT=N 使链接状态进入记录方式,即只有需要全记录时才来自链文件，而需要的单个字符的键输入是由键盘来提供的。AT=Y 使链接状态进入单字符方式，即所有的键输入均来自链文件。

* AU=yn　当 AU=Y 时，打开时钟驱动的重复键功能。延迟一段时间后开始第一次的重复,延迟时间由 AV 参数决定 (延迟时间等于 AV 的值乘 25 毫秒)。以后将以程序扫描该键的速度快速重复，但是重复次数不大于每秒12次。AU=N, 表示关闭重复键功能，在TRS-80 I 型中将会取消重复键功能，而在 Ⅲ 型中，转入驻留在 ROM 内的重复键程序。

* AV=av　当 AU=Y 时要用到 AV 参数。av 是按下任一键以后到该字符第一次重复之间的 25 毫秒间隔的数目。

* AW=aw　AW 是带复核—写的磁盘 I/O 中允许的重试次数。这种 I/O 重试数是与参

数 AM 有关的，只有在扇区检验读已失败 am 次以后，才进行由 AW 参数规定的重试操作。以前，如果扇区写不碰到错误，而复核读有一个错误，就要用户重新再试写一次。现在，如果 aw 大于1，在写盘工作正常，而复核读失败的情况下，就会自动地重新再进行写操作。

＊ AX=ax　这是打印机最高可打印字符的 ASCII 代码。系统程序根据这个值测定什么时候用连续的空格或圆点来代替高于这个值的 ASCII 代码。ax 不能超过 255。这个最高可打印的 ASC II 代码被存放在 4370H 单元（TRS-80 Ⅰ型）及 4290H 单元（TRS-80 Ⅲ型）。

＊ AY=yn　只有在复位前的时刻 DOS 并不工作的情况下的复位（也就是指电源打开及执行非磁盘 BASIC 以后的复位）才用到这个 AY 参数。AY=Y，系统要询问用户关于日期和时间这两个参数。AY=N，不提出这种询问，而自动把日期和时间都置成0。

＊ AZ=yn　只有复位前的时刻 DOS 正在工作（即荧光屏上显示出 DOS READY）的情况下的复位，才用到 AZ 参数。AZ=Y，询问用户日期和时间。AZ=N，保留复位以前的日期和时间。

＊ BA=yn　当 BA=Y时，复位操作可以使 ROUTE，DO，NL 命令进行工作，其结果是荧光屏上没有任何显示信息（包括 DOS 和 BASIC 标题在内），一直要到用户或用户程序执行 ROUTE，CLEAR 或 ROUTE，DO 时为止。BA=N，禁止这种功能。

＊ BB=yn（仅适用于 TRS-80 Ⅲ型）　当 BB=N 时，通知系统每秒发生 60 次时钟中断。BB=Y，通知系统每秒发生 50 次时钟中断。这个选用参数并不设定时钟的工作，只是对时钟的工作认可一下。

＊ BC=yn　当 BC=Y 时，表示用户能手动暂停或取消链接状态。当 BC=N 时，则用户不能手动暂停或取消链接状态。RUN-ONLY 工作方式强制 BC=N。

＊ BD=yn　当 BD=Y 时，表示用户能在复位时用保持按下 ENTER 键来取消 AUTO 命令。当 BD=N 时，用户就不能用这种方法去取消 AUTO 命令，RUN-ONLY 工作方式强制 BD=N。

＊ BE=yn　当 BE=Y 时，使 DOS 命令 R 成为有效，可以用 R 重复前面的 DOS 命令（见2.41节）。当 BE=N 时，R 命令只起返回 DOS READY 的作用。

＊ BF=yn（仅适用于 TRS-80 Ⅰ型）　当 BF=Y 时，可以在复位或打开电源时完成与 DOS 命令 LCDVR，Y（见2.29节）相同的功能。当 BF=N 时，则完成与 LCDVR，N 相同的功能。然而，如果 DOS 发觉没有安装提供英文小写字母的硬件或者硬件不工作，就会强制 BF=N。

＊ BG=yn　当 BG=Y 时，在复位或打开电源时完成与 DOS命令 LC，Y（见2.28节）相同的功能。当 BG=N 时，完成与 LC，N 相同的功能。

＊ BH=yn　在复位或打开电源时，如果 BH=Y，就会启动光标闪烁程序，而 BH=N，则禁止这种功能。

＊ BI=bi　在复位或打开电源时，把 bi 值作为光标字符值，只有在设定 bi=0 时，才使用标准的光标字符值（对于 TRS-80 Ⅰ型为95，Ⅲ型为176）。

＊ BJ=bj　这个参数可以使NEWDOS/80在 CPU 速度提高的情况下，保持磁盘能够正常工作。这个参数可以修正 NEWDOS/80 内部的某些定时程序环，方法是：把这些 Z-80 指令构成的定时程序循环的次数增加 bj 倍。bj 必须是一个大于 0 的整数，并且要等于 CPU 速度提高的倍数（粗略的）。如果不想把这些定时程序循环的时间延长，只需要使 bj=1。如果要想

把定时时间延长，而且新的CPU速度不是原来TRS-80 Ⅰ型或Ⅲ型速度的整数倍，就必须要把 bj 凑成一个合适的值。请注意!!参数 BJ 并不能去改变实际的 CPU 的速度。

＊ BK=yn　如果 BK=Y，表示允许执行 DOS 命令WPDIRP 和 DIRCHECK程序的W和 C 功能。如果 BK=N，则拒绝执行这些功能，并显示 DISK ACCESS DENIED（拒绝磁盘存取）信息。

＊ BM=yn　当 BM=Y时，在磁盘格式化过程中，在所有的磁道已格式化以后，在一个独立的 VERIFYING（复核）阶段中去复核读扇区。按照 DOS 规定，在每个磁道格式化以后，立即要进行磁道扇区的复核读，而这个参数所涉及到的复核读是附加于其上的另外一种复核读。如果 BM=N，则撤消这个VERIFYING 阶段，你如果认为单个磁道格式化时立即进行的复核扇区读已经够了，就选用 BM=N。

＊ BN=yn　（仅适用于 TRS-80 Ⅰ型） BN=N 使得在写单密度磁盘目录扇区时使用 TRS-80 Ⅰ型 TRSDOS 可读的地址标记。如果 BN=Y，在写单密度磁盘扇区时用Ⅲ型 NEWDOS/80 可读的地址标记。BN=Y 只应该用于这种情况：要求这个单密度磁盘是一个可以在 TRS-80 Ⅰ型和Ⅲ型之间互换的 NEWDOS/80 2.0的盘片。

对于单密度磁盘来说，虽然Ⅰ型的 TRSDOS，Ⅰ型和Ⅲ型的 NEWDOS/80 所用的目录中包含的信息是相同的（不包括 NEWDOS/80 附加的那些信息），但是用于表示“被保护的”目录扇区的地址标记字节（磁格式和识别字节的一部分，在软分区的磁盘上每 256 个字节的用户数据的两边就放有这种标记）在Ⅲ型中和Ⅰ型中是不同的。

改变 SYSTEM 的参数 BN 实际上并不能改变任何目录扇区的地址标记。它只是在 DOS 中设定保护扇区写子程序，以便在把保护扇区写入或重写入磁盘的时刻去写规定的地址标记。所以要为一个单密度盘片的目录的所有扇区设置合适的地址标记，就要使用DOS命令 WRDIRP 或带有 W 参数的 DIRCHECK 程序。警告！如果一个单密度盘片已经用于 TRS-80 Ⅲ型或 TRS-80 Ⅰ型（其中 BN=Y）的 NEWDOS/80 系统，而现在必须供 TRS-80 Ⅰ型的 TRSDOS 系统使用，这时用户必须设定 BN=N，并且用 WRDIRP 或带有 W 选用参数的 DIRCHECK 程序重写目录扇区的地址标记。在 BN=N 的情况下，即使用 TRS-80 Ⅰ型 NEWDOS/80的SUPERZAP 程序证明是被保护的目录扇区，也必须重写目录扇区的地址标记，这是因为：虽然系统仅写了用 BN 参数确定的一个地址标记值，但是 TRS-80 Ⅰ型NEWDOS/80可以接受好像“被保护的”另一个地址标记值。

SYSTEM 参数 BO 及往下的一些参数留作将来再用。

3．举例

（1）SYSTEM，0，AL=4，AA=Y，AU=Y，AV=20，AT=Y

这个例子要改变装配于 0 号驱动器内的当前系统盘片的 SYSTEM 控制参数 AL,AA,AU，AV 和 AT。其它的 SYSTEM 参数保持原样不作改变。这个命令执行过以后，会自动显示全部的 SYSTEM 参数。更改过的参数不能立即起控制作用，只有在复位或重开电源时，新参数才能起作用。用户要注意：必须事先把盘片的写保护片撕去，否则不能完成本例中规定的操作。

（2）SYSTEM，2，AP=0FF00H，AN=1，AX=126

这个例子是要改变目前装配于 2 号驱动器内的盘片的控制扇区中 SYSTEM 控制参数 AP，AN 和 AX。不改变其它的 SYSTEM 参数。然后显示该盘片上所有的 SYSTEM 参数，

要使这些参数起作用（假定该盘片是一个NEWDOS/80 2.0系统盘片），就必须把这个盘片从2号驱动器内拿出来，并插入0号驱动器内，然后复位或重开计算机电源。

（3）SYSTEM, 0

这个例子可以显示0号驱动器内系统盘片上的SYSTEM参数，这里给出的是DPS-85微计算机系统用的NEWDOS/80 2.0系统盘片上的典型参数值。

AA=N, AB=N, AC=N, AD=Y, AE=Y, AF=Y, AG=N, AI=N, AJ=N, AL=4/4H, AM=10/AH, AN=0/0H, AO=0/0H, AP=0/0H, AQ=Y, AR=Y, AS=Y, AT=Y, AU=Y, AV=20/14H, AW=2/2H, AX=127/7FH, AY=N, AZ=N, BA=N, BC=Y, BD=Y, BE=Y, BF=N, BG=N, BH=Y, BI=143/8FH, BJ=2/2H, BK=Y, BM=Y, BN=N

2.47 TIME 设置实时钟

1. 命令格式

TIME [,hh:mm:ss]

2. 说明

（1）如果不指定命令中的参数，就以时:分:秒的格式显示当前的时间。

如果指定了hh:mm:ss参数，则时钟被设定为hh:mm:ss，其中hh是一个2位的十进制数字表示的小时值，其取值范围为00—23，mm是一个2位十进制数字表示的分钟值，而ss是一个2位十进制数字表示的秒钟值。系统不对这些值的正确性作检查。这三个值中的每一个要变换成一个单字节的值，并存放在设定时钟的各个字节内。在TRS-80 I型中，设定时钟的3个字节起始于4041H单元（Ⅲ型起始于4217H单元），并以秒、分、小时的次序存放。

（2）当计算机复位或打开电源时，时钟要根据SYSTEM选用参数AY或AZ来设定。时钟每秒修正一次。用户不应该把这个时钟所表示的时间看作为一个很精确的值，因为磁盘I/O操作很频繁，而且有时还要执行中断子程序，这时要使用中断功能，从而一次或多次的关闭时钟中断，结果就使时钟变慢。这里用的实时钟并不是一个硬件时钟，而是用软件来完成的，软件时钟无法测知损失多少时钟中断，因而也无法修正这种误差，故而时钟不太精确。

3. 举例

TIME, 15:25:00

时钟设定为下午3:25。

（2）TIME

显示当前的时间。

2.48 VERIFY 在每次磁盘写以后需要复核读

1. 命令格式

VERIFY [,yn]

2. 说明

（1）NEWDOS/80在它自己的所有目录写入以后和使用逻辑记录或单字节I/O的所有扇区写入以后，都要执行复核读。用4439H调用作整扇区写入时不进行复核读。

* VERIFY 或 VERIFY, Y　这时经 4439H 调用作磁盘写入时要进行复核读。所谓复核读就是在写扇区后再把该扇区读出一次。如果被写的扇区难以辨认或奇偶校验出错，将会产生一个错误。复核并不对数据按字节进行比较。然而，如果复核读检测到一个错误和 SYSTEM 选用参数 AW 不等于1，则写入和复核读将再次进行。假如不进行这种复核读检查，系统就有可能存取已经写入磁盘扇区的数据，从而产生错误。

* VERIFY, N　经 4439H 调用所作的整扇区的磁盘写入不进行复核读。

(2)COPY, EDTASM 和 BASIC 的 SAVE 命令把一个文件写入磁盘的过程中，并不进行复核读，但是，最后要读回整个文件作为复核读。BASIC 语言中所有写入输入/输出(PRINT/INPUT)文件，标记项文件 (marked item file), 固定项文件 (fixed item file) 或字段项文件 (field item file) (其中记录长度不是256) 的磁盘数据均要执行复核读，这是因为实际上用的不是扇区 I/O 而是字节 I/O。记录长度是 256 的字段项文件使用的是扇区 I/O,这时只有使用 VERIFY, Y 命令，才能有复核读操作，否则是不进行复核读的。关于上述的各类文件在本书第八章内有详细的说明，这里不再介绍了。

2.49 WRDIRP 写保护的目录扇区

1. 命令格式

WRDIRP, dn 1

2. 说明

(1)WRDIRP 命令要读出驱动器 dn 1 中盘片的目录扇区,并按目前计算机定义的保护状态进行重写 (见 SYSTEM 的参数 BN 和 BK)。

这个命令用于制作在 NEWDOS/80 2.0 系统管理下可以在 TRS-80 Ⅰ型和Ⅲ型之间进行互换的单密度盘片。

(2)这个命令使用户能在 MINI-DOS 时把目录设置成适当的读保护状态，因为用户想要弄清工作中间发生的问题，这是一个最有效的手段 (用 DIRCHECK 程序是做不到的)。注意!!这个命令要使用从引导扇区第三个字节开始的目录起始 gr. 编号。然后，检查是否是 BOOT/SYS 和 DIR/SYS 的 FPDE。如果通过了这些检查,接着该命令把所有应改变的目录扇区都改变成由 SYSTEM 参数 BN 决定的保护状态。除非你确信发生的问题只是由于 TRS-80 Ⅰ型和Ⅲ型之间的保护状态不同而引起的，否则不要用这个命令，如果你没有把握，最好用 DIRCHECK 程序的 W 功能。

(3)如果 SYSTEM 的参数 BK=N, 就会使 WRDIRP 命令无效。

3. 举例

(1)WRDIRP, 1

处理装配于 1 号驱动器内的盘片，把目录地址标记设定为所用计算机相应的标记，如果所用的计算机是 TRS-80 Ⅰ型，就要根据 SYSTEM 的参数 BN 来设定目录地址标记。

第三章　DOS的子程序

3.1　简要说明

这一章要介绍一些 NEWDOS/80 2.0操作系统使用的子程序，它们也可以供机器语言编程者在他们的程序中使用。如果你既不是汇编语言的程序员，对Z80 机器代码也不感兴趣，就可以不看这一章。我们假定读者是熟悉Z80 机器语言代码并至少了解一种Z80 汇编语言。

为了便于使用，在这些DOS子程序中都给出了入口和出口条件，为了在子程序的说明中不再重复写这些条件，在下面将引用一些英文字母（A,B和C）表示某些条件。

A　表示这个子程序只改变AF 寄存器的内容。其它所有被子程序使用的寄存器在进入子程序时已被保存起来，而在退出子程序时均已恢复。

B　表示在退出子程序时，如果该子程序执行期间没有发生错误，则Z状态标志被置位。如果遇到一个DOS错误，则NZ 状态标志被置位，而且A 寄存器内放有 DOS 错误代码。设置Z和NZ标志要优先于设置其它标志，如 C 和 NC 标志。

C　表示在进入子程序时，DE指向一个打开的FCB（文件控制块）。

有些子程序的使用与TRSDOS不兼容。对这些不兼容的子程序，在介绍到它们的时候会给出简要的说明，希望读者仔细注意这种差别。读者还应该注意到在两种操作系统中FCB字段的NEXT和 EOF 结构形式是不同的（见5.9节 FCB 说明）。介绍每个子程序的时候都给出它的入口地址（在 CALL 或Z80转移指令中要用到这个地址），以及它的标题和说明。

除非另有说明，否则 DOS 子程序就使用引用它的程序所设定的堆栈。除了特定的终止型子程序（即主程序调用该子程序后，整个程序就结束或返回到 DOS 状态）以外，一般情况下，DOS 子程序均要返回到调用它的程序。

其中许多子程序要用到文件控制块(FCB)。TRS-80 I 型和 III 型的 NEWDOS/80 和 TRS-80 I 型的TRSDOS 都使用32字节的 FCB,而 III 型的TRSDOS使用50字节的 FCB。NEWDOS/80 可以运行有50字节 FCB 的用户程序，但是将只用这个 FCB 最前面的 32 个字节。反之，III 型的 TRSDOS运行只有32字节 FCB的程序时将会发生问题。

下面列出的子程序说明并不一定按照入口地址的大小顺序来排列。

3.2　402DH　　无错误退出

这是一种终止型子程序 (Dead end routine)。在没有错误的情况下结束程序就跳转到402DH。DOS按下面的次序检查它自己的状态。

如果在MINI-DOS 或 DOS-CALL 状态，把堆栈指针设置为最后一个DOS命令之前它所在的地方；否则就把堆栈指针设置到DOS的堆栈区，而且根据SYSTEM 的选用参数 AG 来决定 BREAK 键功能是否有效。

如果在DOS-CALL级而且不是链接状态或者链接不在当前的 DOS 级上继续，则除 AF以

外的所有寄存器都要被恢复到它在进入DOS-CALL时原有的状态，Z标志被置位，并且返回到DOS-CALL的调用程序。如果这是最外层的DOS-CALL级，则退出DOS-CALL状态。

如果在RUN-ONLY和非链接状态，则要显示 RUN ONLY STOPPED!! KEY 'R' FOR RESET, DOS等待回答，然后执行 DOS 命令 BOOT（请看2.7节）。

如果在DOS-CALL级，并且在当前的 DOS-CALL 级上继续处于链接状态，则 DOS 要等待来自链接文件的下一个命令。

如果在 MINI-DOS 状态，则要显示MINI-NEWDOS/80 READY, DOS等待下一个命令。

如果链接有效，则DOS要等待来自链接文件的下一个命令。

显示 NEWDOS/80 READY, DOS 等待下一个输入命令。

3.3 4030H 有错误退出

这是一种终止型子程序。结束有错误的程序（这个错误或者已经显示出来，或者还没有被显示），并转移到4030H。DOS 的工作与 402DH 相同，其差别仅在：

如果在链接状态，则取消链接。

如果在DOS-CALL级，要用 402DH 同样的方法退出当前的 DOS-CALL，只是要把C标志置位。

3.4 4400H 同402DH一样

3.5 4405H 进入DOS和执行命令

这是一种终止型子程序。进入 DOS 并且把堆栈指针设置在自己的区域。HL 指向一个命令（由0DH字节结束），DOS把这个命令作为下一个命令。DOS把这个命令传送到它自己的80字节缓冲区，然后执行这个命令。

3.6 4409H DOS有错误则退出

如果寄存器 A 的位 7 等于 0 则成为终止型子程序。因 DOS 的错误而终止的程序 跳转到4409H，这时DOS的错误代码在 A寄存器内，并且A寄存器的位 7 等于 0 。根据 DOS 的状态要发生如下一些操作：

* 如果在链接状态则取消链接。

* 如果在DOS-CALL级，则要用与402DH同样的方法退出DOS-CALL级，只是要把 NZ和NC状态标志置位，同时DOS错误代码在A寄存器里。错误信息并不显示出来。

* 否则要显示错误信息，同时跳转到402DH。

一个程序调用 4409H 子程序就可以在荧光屏上显示出对应于 A 寄存器内错误代码的错误信息，方法是：在调用以前，先把错误代码放入A寄存器，并且把A 寄存器的位 7 置成 1 。在退出这个子程序时，只有 F 寄存器的内容被改变了。

如果A寄存器的位 6 等于 0 ，I 型TRSDOS将会把诊断情况打印出来。而 Ⅲ 型 TRSDOS在位 6 等于 0 时只显示错误编码，只有在位 6 为 1 时才显示错误信息。NEWDOS/80 不检测这一位的值。

把4409H开始的 4 个字节置入CD, 0D, 44, C9，这样，错误显示子程序就能调用DEBUG，

以代替显示错误信息。

3.7 440DH　　进入 DEBUG

用户程序有两种办法进入DEBUG:

(1)使用 Z80 指令 RST 30H。

(2)使用 Z80 指令 CALL 440DH。

当不再使用 DEBUG 时，只要 PC 寄存器的内容没有改变，用 DEBUG 命令 G 就可以返回到 RST 30H 或 CALL 440DH 后面的那一条指令。

3.8 4410H（在 TRS-80 Ⅲ型中是 447BH）放进用户定时中断子程序的队列

这个子程序要改变寄存器 AF, BC 和 HL 的内容。在进入这个子程序时，DE 指向用户中断子程序，用户中断子程序必须符合下面的格式：

* 第 1,2 字节由 DOS 用作正向链指针。在入口点，这两个字节可以是任意的值。

* 第 3 字节放有两次调用用户定时中断子程序之间所需要的25毫秒间隔数。例如，如果每秒要引用一次这个子程序，则第三字节必须设定为40（或28H)。DOS 不改变这个字节。

* 第 4 字节存放计数值，每进行一次中断就减1。在入口点，这个字节应该预置为大于 0 而小于或等于第三字节数值的值。每经一个25毫秒的中断，DOS 就将此值减 1 。如果结果不为0,则对这个25毫秒的中断并不去执行用户的中断子程序。如果结果为0,则要把第三字节内的值移到第四字节，并保存寄存器 HL, DE, BC 和 AF 的值，在第五字节调用用户的中断子程序。用户子程序要用其它一些寄存器时，也必须先把它们保存起来，最后再给予恢复。关闭了中断，用户子程序就不准再打开中断。

当用户子程序处在25毫秒的中断链时，就不允许以任何方式改变它的内容，只有关中断才能对它进行修改；而最前面的两个字节不许改变。

TRS-80 Ⅰ型的 TRSDOS 使用 4 个中断向量(它们是4410H, 4413H, 4416H和 4419H)，实现对用户中断子程序的管理，而 NEWDOS/80 只用 4410H 和 4413H, 所以在TRSDOS中任何一个使用25毫秒中断用户子程序的程序，如果要在 NEWDOS/80 系统中使用，则必须经过修改。这是两种 TRS-80 Ⅰ型系统之间的主要不兼容之处。

TRS-80 Ⅲ型的 TRSDOS 没有为处理用户定时子程序提供任何手段，4410H—441BH 另有其它用途（如用于 HIMEM)， 所以用户不能使用这种25毫秒的内部中断链来构成用户的定时中断子程序。

Ⅲ型的 NEWDOS/80 继续为用户提供 Ⅰ型中具有的用户定时中断子程序的机构，只是用 447BH 作为子程序的入口向量，以代替 Ⅰ型中的 4410H, 在Ⅲ型中实际是以1/30秒或1/25秒来计数的。

3.9 4413H　　把用户定时中断子程序撤出队列

寄存器 AF, BC, DE和 HL 的内容要改变。由寄存器 DE 作指针的用户中断子程序(3.7节中有说明）从25毫秒中断链中脱出来（如果该子程序在25毫秒中断链中)。用户子程序不再参与中断，现在用户可以修改它们。

与 TRSDOS 的不兼容处，请看 3.8 节。

3.10 4416H 保持磁盘驱动器旋转

如果磁盘驱动器正在旋转，则重新选择当前要用的驱动器，并使它继续旋转2.4秒以上。AF寄存器的内容被改变。

在TRSDOS中没有这个子程序。

3.11 4419H DOS-CALL。执行一个DOS命令并返回

这个子程序是DOS调用。DOS不用系统堆栈区，而是使用用户堆栈区的剩余部分。除AF以外的所有寄存器的内容都要保存在堆栈中，当返回时再恢复。待执行的命令由HL指出（命令必须少于80个字符，并且一定要用0DH字节作结束），可以是当前DOS所处状态下的任何合法命令。DOS设置DOS-CALL状态（如果尚未设定），并保存当前堆栈指针，然后执行该命令。命令可以是用户程序的援引项。

现在的版本中，在链接状态DOS-CALL是合法的，而在NEWDOS/80 1.0版本中是不允许的。

DOS-CALL可以使BASIC执行一个包含在BASIC语句CMD "xx" 中的DOS命令，其中xx是一个DOS命令。

DOS-CALL的调用者要确保内存使用不发生冲突，以及有足够的可用堆栈空间。

可以执行DOS-CALL的嵌套调用。从一个DOS-CALL级退出时，就返回到下一层。当退出最外层时，DOS就脱离DOS-CALL状态。

如果DOS命令引用一个程序，这程序可以用它自己的堆栈区，而且必须使用三个出口之一退出，三个出口地址是：402DH，4030H或4409H。在退出时，程序可以把一个两字节的参数存放在4403H和4404H（十进制为17411和17412）单元供调用它的程序使用。

在TRSDOS中使用4419H向量与此不同，其不兼容处请看3.8节。

关于DOS-CALL的进一步说明请看4.4节。

3.12 441CH 提取文件标志符

从HL所指出的文本中提取一个文件标志符，并把它放在由DE所指出的区域中，最后用字节03H作结束。这个子程序要改变AF, BC和HL的内容。

如果文本的第一个字符是A—Z或0—9中的一个，或者文本的第一个字符是*，接着是A—Z或0—9中的一个，则要把文本从HL所指出的区域传送到DE指出的区域，直至遇到一个不是/,.,:,A—Z或0—9的字符为止，或者是32个字节已经传送完成时为止。如果传送字符少于32个字节，则要把03H字节放置在DE所指出的区域最后一个字节后面，表示文件标识符的结束，然后返回，Z标志置1。如果文件标识符多于31个字符，就像下面要讨论的那样，认为这个文件标识符是非法的。

如果第一个字符是非法的，或者第一个字符是*，但第二个字符是非法的，则要返回，并把NZ标志置成1。

在退出此子程序时，如果终止/非法字节等于03H或0DH, 则HL就指向这个字节；否则HL指向下一个字节。

用户应该注意到NEWDOS/80对文件标识符的合理性不作检验；而把检验工作留给

OPEN 子程序（4420H 和 4424H)去完成。

3.13 4420H 打开一个新的或已有的磁盘文件的 FCB

条件 A 和条件 B 有效（见 3.1 节)。入口条件与 4424H 子程序相同，对于 4420H 来说，4424H作为它的一个子程序被立即执行。如果 4424H 成功地打开了一个已有文件，就不再需要作进一步的工作，接着就是退出这个子程序，Z 和 NC 标志要置 1。如果文件没有找到，这个子程序就要着手建立这个文件。

如果在由寄存器 DE 指出的 FCB 中，文件标识符规定了一个明确的驱动器编号，而且装配在这个驱动器内的盘片上有空闲的目录项，则就要在此盘片上建立该文件的目录项，而不管盘片实际上有否空闲的文件空间。如果文件标识符没有规定驱动器编号，则系统就要搜索已装配好的磁盘，从 SYSTEM 选用参数 AO 所指定的驱动器开始，往有较高编号的驱动器搜索直至找到有空闲目录项的磁盘为止。如果没有空闲的文件目录项，就不能建立文件，并采用有错误的退出（即 4030H 子程序)。

建立一个文件就是把一个空闲的文件目录项(FDE)变成一个文件主目录项(FPDE)。这就要在 FPDE 中插入文件名和文件名扩展符（如有的话)，把通行字（如有的话）编码成由2字节组成的散列代码所表示的修改通行字和存取通行字，保存从寄存器 B 内取来的逻辑记录长度（0 即表示 256)，把文件结束标记(EOF)设定为 0，把存取级别设定为 FULL（即允许对该文件进行任何操作)，以及把文件标记为非系统的和非隐性的文件。这时并未为文件分配磁盘空间。实际上，DOS 甚至还没去查看盘片上是否有空闲的文件空间。注意，虽然在建立文件的时候已经把逻辑记录长度(LRECL)存入了 FPDE，但是从来也不去用它。以后每次打开文件用的是由 B 寄存器提供的 LRECL。

文件建立以后，调用 4424H 处的 DOS 子程序完成打开文件的工作。在文件建立和打开成功以后就要退出该子程序，并把 Z 和 C 标志置 1。

3.14 4424H 打开一个已有的文件的 FCB

条件 A 和条件 B 有效（见 3.1 节)。进入 4424H 子程序时，寄存器 DE 指向 FCB,它包含有待打开的文件的文件标识符，寄存器 HL 指向一个 256 字节的缓冲区，该缓冲区是供这个 FCB 在磁盘扇区读、写时用的，而寄存器 B 包含有 LRECL（0 表示 256)。如果在文件标识符中明确地指定了驱动器的编号，则搜索仅限于该驱动器；否则搜索从 0 号驱动器开始，继续进行到较高编号的驱动器，直至找到具有指定文件名和文件名扩展的文件为止。如果没有找到文件，则采用有错误的退出。

假如通行字有效，并且文件有一个非空缺通行字，这时如果文件标识符内既不包含修改通行字，又没有存取通行字，则采用有错误的退出。假如通行字无效或者文件没有通行字、或者只指定了修改通行字，则把 FCB 的存取级别设定为 FULL（即允许对该文件进行任何操作)；否则，就把 FPDE 中的存取级别放置到 FCB 中，以便限定这个文件的存取类型。

把 FCB 从只含有文件标识符的 FCB 变成包含有有关该文件的其它一些信息的FCB,这样在打开 FCB 时就可以减少目录的输入/输出总数。这种 FCB 的变换有如下这些内容：从 FPDE 中复制 EOF 和前面四项的内容，保存从寄存器 B 内取得的 LRECL，如果 LRECL 不等于 0（表示逻辑记录要处理)，就把 FCB 第二字节的位 7 置 1，把 NEXT 置 0，保存驱动器编

号和 FPDE 的目录项代码(DEC), 保存从寄存器 HL 取得的256字节缓冲区指针，设定存取级别，设定 FCB 第二字节的位 5 为 1，以表示缓冲区不包含当前扇区，同时把 FCB 第一字节的位 7 置 1，以表示 FCB 是打开的。

3.15 4428H 关闭一个 FCB

条件 A、条件 B 和条件 C 有效（见3.1 节）。这个子程序取消了 FCB 和文件之间的联系。如果 FCB 第二字节的位 4 等于 1，就要把 FCB 的缓冲区内容写入磁盘，就像调用 4439H 子程序一样。如果 FCB 的 EOF 与 FPDE 中的 EOF 不同，就要把 FPDE 中的 EOF 修改成新的 EOF。如果文件有超出 EOF 的多余的 gr.，而且允许自动重新分配空间，这时就得把多余的 gr. 释放出来。最后把 FCB 变回到文件打开之前的状态，即包含有一个文件标识符的形式，使它包含有文件名、文件名扩展和驱动器编号。这个文件标识符可以为今后再次打开这个文件时使用，而且不必提供通行字。

3.16 442CH 删除一个文件

条件 A、条件 B 和条件 C 有效（见 3.1 节）。把与 FCB 相应的文件删除，与 DOS 库命令 KILL 相同（请看2.27节）。把 FCB 置成全 0 。

3.17 4430H 装入一个程序文件

条件 A 和条件 B 有效，只是寄存器 AF, BC 和 HL 要改变，而且在退出子程序时 HL（和4403H, 4404H)包含有这个程序入口的地址。在进入这个子程序时，寄存器 DE 指向包含有程序的文件标识符的 FCB。这个子程序所完成的工作与 DOS 库命令 LOAD 是相同的（请看2.32节）。

3.18 4433H 装入并执行一个程序文件

这是一个终止型子程序。在进入这个子程序的时候，DE 指向包含有这个程序的文件标识符的 FCB。寄存器 AF 和 BC 的内容要改变；其它所有寄存器的内容在该程序执行时不经改变传送到程序中。打开文件，装入程序，然后开始执行，这与DOS执行一个非DOS库命令是一样的，只是有一个非缺项的文件名扩展。如果在打开或装入过程中发生了错误，DOS就要返回到 4409H。假如 DEBUG 有效（请看 2.17 节），就要在程序执行前进入 DEBUG。

3.19 4436H 从磁盘读一个磁盘扇区或逻辑记录

从磁盘读一个扇区或把一个逻辑记录从 FCB 的缓冲区传送到调用它的程序的缓冲区。

条件 A、条件 B 和条件 C 有效（见 3.1 节）。

如果FCB的第二字节的位 7 等于 0，由NEXT字段的高位两字节所表征的扇区内容就被读进 FCB 缓冲区，假如没有错误或错误代码为6(扇区读保护)，就把 NEXT 字段往高移256字节。如果发生除代码 6 以外的错误，NEXT 字段就不必移高，也就是说，用户可以再试读该扇区。

如果 FCB 第二字节的位 7 等于 1，则长度等于 LRECL（其中 0 表示 256）的逻辑记录就要从 FCB 的缓冲区移到进入这个子程序时寄存器 HL 所指出的缓冲区内。每移动一个字节，NEXT 字段就要加 1 。当 FCB 的缓冲区移空的时候，下一个文件扇区就被自动读进 FCB缓冲

区，同时把字节继续移入 HL 指示的缓冲区。如果有错误发生（包括错误代码 6），就会终止逻辑记录的传送，使 NEXT 字段成为已传送的字节数。

如果 FCB 第一字节的位 1 等于 1，则 NEXT 和 EOF 字段被看作为磁盘中的（而不是文件中的）相对字节地址(RBA)，这就使用户能够去读一个磁盘（而不是一个文件）。FCB 第一字节位 0 的使用在下面3.20节要介绍。DOS 子程序 0013H, 001BH, 4439H, 443CH 和其它一些间接读或写扇区子程序的操作也按照这两位中的任一位是否为 1 来决定如何进行。这两位的使用方法是与 TRSDOS 不同的。

NEWDOS 和 TRSDOS 之间的一个不兼容处是发生在当程序为了测定该文件中的字节数，而从 FCB 中读 EOF 的时候。然而，在大多数情况下，用户不必知道 EOF 是什么样的。在 TRSDOS 和 NEWDOS 这两个系统中，用户都能够逐个扇区地去读文件，并等待两种EOF 错误中的一种出现。如果错误代码是 1CH，荧光屏将显示 END OF FILE ENCOUNTERED（没有遇到文件的结束符），说明文件刚好终止于一个扇区的边界处，而最后读进的扇区就是文件的最后一个扇区。如果错误代码是 1DH，荧光屏将显示：PAST END OF FILE（超过了文件的结束符），则最后读进的扇区也是文件的最后一个扇区，但仅仅是个不完整的扇区，在 FCB+8 处的值就是该文件占有此扇区的字节数。这些情况对于 TRSDOS 和 NEWDOS 两种系统都是适用的，所以相同的程序在两种系统中都能工作。

3.20 4439H 把一个扇区或逻辑记录写入磁盘

不经复核把一个扇区写到磁盘上，或者把一个逻辑记录从调用它的程序的缓冲区传送到 FCB 的缓冲区。

条件 A、条件 B 和条件 C 有效。

如果 FCB 的第二字节的位 7 等于 0，就把 FCB 缓冲区的内容写到由 NEXT 字段所规定的磁盘扇区内。只有 VERIFY 命令（请看 2.48 节）中使用 y 参数时，才进行复核读，否则就不进行复核读。如果没有错误，而且 NEXT 字段的低位字节等于 0，就要把 NEXT 字段往高移256字节。不论 NEXT 是否移高，如果 NEXT 此时超过了 EOF 或者 FCB 第二字节的位 6 等于 0，就要把 EOF 设定得与 NEXT 相等。如果发生错误，则不改变 NEXT 字段，这就使得用户可以重新再试写这一个扇区。

如果 FCB 第二字节的位 7 等于1，则要把一个逻辑记录（长度等于 FCB 的 LRECL）从调用它的程序的缓冲区（进入此子程序时由寄存器 HL 所指出）移入 FCB 的缓冲区。每移动一个字节，NEXT 字段就加1，如果此时 NEXT 超过了 EOF 或者 FCB 第二字节的位 6 等于 0，则要把 EOF 设定得与 NEXT 相等。当 FCB 缓冲区填满时，就要把该缓冲区的内容写到相应的磁盘扇区上，并且带有复核读操作，然后继续移动逻辑记录，去填满 FCB 的缓冲区，以便写下一个文件扇区。只要有错误发生，就要终止逻辑记录的移动，使 NEXT 字段表示已传送的字节数。

FCB第一字节位 1 的作用与3.19节给出的情况一样。如果该字节的位 0 等于1，则这个扇区被写上保护（在扇区读出时错误代码为 6）。

如果在写一个保护扇区以后要作复核读，这当然不是错误操作，因此并不把错误代码 6 送回给调用它的程序。

与 TRSDOS 很主要的一个不兼容处是在 NEWDOS/80 中把一个扇区写到磁盘上，并且

NEXT 的低位字节不为 0 的时候，NEXT 不往高移256字节。在这种情况下，NEWDOS/80就假定调用这个子程序的程序正在写文件的最后一个扇区，而该扇区并不填满，而且还假定 NEXT 已经是原来的 EOF（如果写操作要修改 EOF）相对字节地址(RBA)值了。

NEWDOS 和 TRSDOS 之间另一个不兼容处是对于那些一个一个扇区接着写的文件（通常并不终止于扇区之间的分界处）设置最后的 EOF 不一样。然而如果程序知道什么时候写最后一个扇区(不论该扇区是填满的还是不填满的)，并把所希望的 EOF 低位字节值刚好在写最后一个扇区之前存入 FCB+5 的地方，则 TRSDOS 和 NEWDOS 这两个系统写有相同的 EOF。于是同一个程序就可以在 TRSDOS 和 NEWDOS 这样两个系统中正常工作了，收到了兼容的效果。

3.21 443CH 把一个扇区或逻辑记录写入磁盘后进行复核读

这个子程序是与 4439H 子程序相同，只是每次扇区写操作以后均要作复核读操作。

3.22 443FH 使 FCB 指向文件的起点

条件 A、条件 B 和条件 C 有效（见3.1节）。如果 FCB 有一个扇区正等待写入（FCB 的第二字节的位 4 等于1），就调用 4439H 子程序，把该扇区写到盘上，把 FCB 的 NEXT 字段置 0 。FCB 第二字节的位 5 置 0，表示缓冲区不包含当前扇区的内容。

3.23 4442H 使 FCB 指向一个确定的文件记录

条件A、条件 B 和条件 C 有效（见3.1节）。把 NEXT 设定为逻辑记录的 RBA，其相对记录编号（0 等于第一个记录）在进入这个子程序时要放在寄存器 BC 中。如果新的 NEXT 与老的 NEXT 在相同的扇区内，则不改变当前扇区的状态（即如果 FCB 第二字节的位 4 等于1，则不把该扇区写到盘上）。如果新的 NEXT 与老的 NEXT 并不在相同的扇区内，则：

（1）如 FCB 第二字节的位 4 等于1，就要把老的扇区写回到盘上；

（2）把 FCB 第二字节的位 5 设置为1，以表示新的扇区尚未被读进缓冲区。

3.24 4445H 使 FCB 退回一个记录

条件和执行情况同4442H子程序，只是要把 NEXT 字段减去 LRECL 值。

3.25 4448H 使 FCB 指向 EOF

条件和执行情况同 4442H 子程序，只是要把 NEXT 字段设定为等于 EOF。

3.26 444BH 分配文件空间

条件 A、条件 B 和条件 C有效（见3.1节）。

如果由 FCB 的 NEXT 字段高位两个字节所表征的文件扇区还没有分配给文件，则包含有这个文件扇区的 gr. 和较低扇区的 gr. 一起分配给到目前为止尚未分配到空间的文件。这就使得编程者在文件实际需要以前就能着手分配文件空间，这个子程序对于那些把数据放置到缓冲区以前就想知道扇区能否写入的用户特别有用。如果一个文件的大小在进行写操作之前就能预先确定（就像在 COPY 中所做的那样），就可预先分配需要的 gr.，这比在写文件过程

中再分配gr. 要节省大量时间。

这个子程序的入口地址与 TRSDOS 中的不同。

3.27 444EH 使 FCB 指向规定的相对字节地址 (RBA)

条件和执行情况与调用 4442H 子程序相同，只是新的 NEXT 位置值（三个字节）是取自 HL 和 C，其中 H 放的是高位值，C 放的是低位值。

这个地址规定得与 TRSDOS 不一样。

3.28 4451H 把 FCB 中的 EOF 值写到目录中

条件 A、条件 B 和条件 C 有效（见3.1节）。

如果 FCB 中的 EOF 值不同于文件的 FPDE 中的 EOF 值，就要把 FCB 的 EOF 值写进磁盘上的 FPDE 中去。

这个地址规定得与 TRSDOS 不一样。

3.29 445BH 选择和打开指定的驱动器

条件 A 和条件 B 有效（见3.1节）。

进入这个子程序时，寄存器 A 内放有驱动器的编号。该驱动器就成为当前被选中的驱动器，如有必要就启动它。

3.30 445EH 测试磁盘是否装配好

条件和执行情况与 445BH 相同，只是还须对驱动器进行测试，以判定磁盘是否装配好、磁盘是否在旋转。如果没有旋转(相应错误代码为 8)则要送回 DEVICE NOT AVAILABLE（驱动器不能用）信息。

3.31 4461H 把 * name 1 的子程序放入队列

寄存器 HL 指向主存中要挂入用户逻辑子程序链中的用户子程序。这个子程序的开始12字节规定如下：

4 个字节保留给 DOS 使用。

8 个字节是逻辑子程序名字段，它放有1—8个字符（该子程序的名字），不满 8 个字符在右边填上空格。

如果一个子程序与队列中已有的名字相同，则会送回相应于 FILE ALREADY EXISTS（文件已经存在）信息的错误代码，并使 NZ 标志置1。否则把这个用户子程序加入队列，并退出这个子程序，把 Z 标志置 1。这个子程序要改变 HL, DE, BC 和 AF 的内容。这是 NEWDOS/80 新增加的功能。

今后，只要执行[* name 1]或[* name 1, 参数]形式的 DOS 命令，DOS 就会搜索队列寻找名字为name 1 的子程序，使 HL 指向这个参数(如果有的话)，然后跳到这个子程序的第13字节，当这个子程序结束时，将通过 402DH, 4409H 或4030H退出这个子程序。这个子程序或许会用到所有的寄存器，同时能利用 4403H 和 4404H 两个单元来接收或送回一个参数。如果在队列中没有逻辑子程序名字 name 1, 就要送回 FILE NOT IN DIRECTORY(目录

中没有这个文件）信息所对应的错误代码，并使 NZ 标志置1。

3.32 4464H 把＊name 1 的子程序撤出队列

HL 指向 3.31 节中所规定的一个逻辑子程序。如果这个子程序不在 DOS 的逻辑子 程序队列中，则退出这个功能，寄存器 A 中放有相应于 FILE NOT IN DIRECTORY 错误的错误代码，并使 NZ 标志置1。否则从队列中撤销这个子程序，也就是说，以后用＊name1的命令将无效，并显示出FILE NOT IN DIRECTORY 信息。4464H 子程序要改变寄存器HL，DE，BC 和 AF 的内容。这是为 NEWDOS/80 新增加的功能。

3.33 4467H 把信息发送到显示器

条件 A 有效（见3.1节）。

把由 HL 所指出的字节开始直到 0DH 字节(EOL)、也包括 0DH 字节，或者由 HL 所指出的字节开始直到 03H 字节(EOM)、但不包括 03H 字节的所有信息，发送到显示器去显示。

3.34 446AH 把信息发送到打印机

与 4467H 子程序相同，只是把信息发送到打印机去打印。

3.35 446DH 把时钟的时间变换成 HH:MM:SS（时:分:秒）格式

把 TRS-80 I 型4041H—4043H （TRS-80 Ⅲ 型为 4217H—4219H）单元的当前时钟值变换成 HH:MM:SS 格式，并保存在由 HL 指示的 8 个字节里。这个子程序要改变 AF，BC，DE 和 HL 寄存器的内容。在退出这个子程序时，HL 指向HH:MM:SS下面的一个字节。

3.36 4470H 把日期变换成 MM/DD/YY （月/日/年）格式

这个子程序与 446DH 子程序相同，只是把 TRS-80 I 型的 4044H—4046H (TRS-80 Ⅲ 型为421AH—421CH)单元的日期值变换成 MM/DD/YY 格式。

3.37 4473H 把缺项名字扩展符插入文件标识符

如果由寄存器 DE 指出的文件标识符没有文件名扩展符，则把由 HL 寄存器所指出的 3 个字符作为它的文件名扩展符。最后得到的文件标识符不能超出31个字符。这个子程序要改变 AF 和 HL 寄存器内容。

3.38 0013H 从磁盘文件中读一个字节

这是 DOS 的单字节读出子程序，它的入口地址在 ROM 中。

条件 A、条件 B 和条件 C 有效（见 3.1 节）。

如果包含有要读的字节（由 NEXT 指出）的磁盘扇区不在 FCB 的缓冲区内，这时就要把它读进缓冲区。然后把要读的字节放进寄存器 A，供调用它的程序使用。FCB 的 NEXT 字段加1。

3.39 001BH 把一个字节写到磁盘文件中

这是 DOS 的单字节写入子程序，它的入口地址在 ROM 中。

条件 A、条件 B 和条件 C 有效（见3.1节）。

如果 FCB 的 NEXT 位置所对应的磁盘扇区不在 FCB 的缓冲区内，而且 NEXT 不在扇区的边界，也不等于 EOF，这时就要把它读进缓冲区。在入口时，寄存器 A 内存放的字节要放进缓冲区，并把 NEXT 加 1。如果缓冲区是满的，则把这个扇区写到盘上，就像调用了一次 443CH 子程序一样。

3.40 447BH 与4410H相同（只适用于TRS-80Ⅲ型）

只适用于 TRS-80Ⅲ型，它完成与 I 型 4410H 子程序相同的功能（参看 3.8 节）。对于 TRS-80Ⅲ型，不准使用 4410H 子程序。

第四章 NEWDOS的特有性能

这一章要讨论 NEWDOS/80 特有的一些性能，如 DEBUG, MINI-DOS, CHAINING, DOS-CALL, JKL 和异步操作。DEBUG, DOS-CALL 和异步操作对机器语言编程者和对 Z80程序感兴趣的用户是特别有用的。其它用户只要粗读一下 DEBUG 和 DOS-CALL 部分就可以了。其中的 MINI-DOS 和 JKL 这两个功能是每个用户立即就能使用的。CHAINING功能比较复杂，新用户可以用 BASIC 程序 CHAINBLD/BAS 来了解一下链接概念，首先看一下示范的链接文件 CHAINTST/JCL, 然后建立一个简单的链接文件，逐步熟悉其使用。

4.1 DEBUG 调试、查错程序

DEBUG 是机器语言编程者的一个很重要的工具，而且还能够供高级语言编程者使用，NEWDOS/80 内包含有 DEBUG 程序，它可以中断当前执行的程序，查看和修改存储器内容，查看和修改磁盘以及实行单步操作。这就为调试机器语言程序带来了很大的方便。可以用下面三种方法进入 DEBUG 程序。

1. 同时按下三个键（键盘上的“1”,“2”,“3”键）。为了使“1,2,3”联键起作用，必须满足下列条件。

（1）SYSTEM 选用参数AB=N。

（2）SYSTEM 选用参数 AE=Y。

（3）或者是开中断，或者是主程序正在等待经标准键盘输入子程序来的键盘输入和 SYSTEM 选用参数 AJ=N。

（4）当前 DOS 必须不在使用它的覆盖区（主存中 4D00H—51FFH 单元）。

（5）DOS 必须不禁止覆盖功能。

2. 执行 RST 30H 或 JP 440DH 或 CALL 440DH 指令。

3. 如果打入了 DOS 命令 DEBUG(见2.17节)，接着打入需要调试的机器语言程序的文件名，就在机器语言程序开始执行之前自动进入 DEBUG 程序。

进入 DEBUG 程序将会：

（1）把所有寄存器的内容保存在被中断程序的堆栈中；

（2）接着的堆栈单元就要移交给 DEBUG 程序使用；

（3）上一次设置的所有断点都要被清除；

（4）存储器内容的显示格式（指 DEBUG 功能的显示格式）与上一次退出 DEBUG 时所采用的格式一致，显示区域也不变；

（5）显示器左下角的光标表示 DEBUG 程序正在等待用户发命令。

所有的 DEBUG 命令(包括单个字符形式的命令)都必须用 ENTER （回车键）作结束。如果打错了命令，但尚未按 ENTER 键，这时只要按退格键就可以把它们删去，重新打入正确的命令就可以了。也可以在按下 ENTER 键以前，同时打入“SHIFT”和“←”键，把刚打入

的整行命令删去。

DEBUG 程序中有两种显示方式，一种为X 方式，另一种为 S 方式，这两种方式每行都显示16个字节的存储器内容，并且在屏幕的左半部用十六进制格式表示，右半部显示相应的字符（请看后面的实例）。如果 SYSTEM 的选用参数 AI＝Y，则字符显示部分将会包括小写英文字母。

DEBUG 碰到一个错误时，就会显示出 ERROR 信息，等待用户处理，只要按下 ENTER 键就可以清除这个错误信息。接着用户就可以重新打入命令。

下面介绍 DEBUG 程序可用的命令。除非另有说明，命令中所用到的数字均为16进制格式，不再标注 H 后缀了。

1. X 命令　如果 DEBUG 原来不在 X 显示方式，执行这个命令就使 DEBUG 进入X显示方式。

X 方式在荧光屏上显示15行信息。第 1 行到第 4 行显示第一个64字节的内存储器区。第 5 行显示的是中断时的（或要被置换的）Z80 寄存器 AF，BC，DE 和 HL 的内容。第 6 行到第 9 行显示的是第二个64字节的内存区。第10 行显示中断时的（或要被置换的）Z80 寄存器 AF′BC′，DE′ 和 HL′ 的内容。第11行到第 14行显示的是第三个64字节的内存区。第15行显示中断时的（或要被置换的）Z80寄存器 PC，SP，IX和 IY 的内容。其中 AF 和 AF′的显示还对 F 寄存器内的每位给予标记，表明 Z80 状态标志位的情况，如果该位处于“1”状态，就给出相应的字母字符，如果该位为“0”，则给出“－”标志。各位的含义如下：

位 7　S＝负号位

位 6　Z＝结果为 0

位 5　1＝这位不用

位 4　H＝ 半进位标志

位 3　1＝这位不用

位 2　P＝偶校验或溢出

位 1　N＝减法标志

位 0　C＝进位标志

使用 X 显示方式使用户能够跟踪寄存器，并且一次查看三个独立的内存储器区。

下面是 X 方式的荧光屏显示实例：

```
  00FF  CDA7 28C3 191A 4D45 4D4F 5259 2053 495A    ..(...MEMORY.SIZ
  010F  4500 5241 4449 4F20 5348 414? 4B20 4C45    E.RADIO.SHACK.LE
  011F  5645 4C20 4949 2042 4153 4943 0D00 1E2C    VEL.II.BASIC...,
  012F  C3A2 19D7 AF01 3E80 013E 01F5 CF28 CD1C    ......>..>...(..
AF  = 8145 -Z---P-C   BC  = 2844    DE  = 4480      HL  = 431E
  013F  2BFE 80D2 4A1E F5CF 2CCD 1C2B FE30 D24A    +...J.....+.0.J
  014F  1E16 FF14 D603 30FB C603 4FF1 875F 0602    ......0...O..._..
  015F  7A1F 577B 1F5F 10F8 798F 3C47 AF37 8F10    ..W._....<G.7..
  016F  FD4F 7AF6 3C57 1AB7 FA7C 013E 3047 F1B7    .O..<W.....>.G..
AF' = FFFF SZ1H1PNC   BC' = 0A00    DE' = 0200      HL' = 37EC
  017F  7828 1012 FA8F 0179 2F4F 1AA1 12CF 29C9    .(.....y/O....).
```

```
018F  B118 F9A1 C6FF 9FE5 CD8D 09E1 18EF D715    ................
019F  3A99 40B7 2006 CD58 03B7 2811 F5AF 3299    :.@....X..(...2.
01AF  403C CD57 28F1 2AD4 4077 C384 2821 2819    @<.W(.*.@...(!(.
PC  = 6F00          SP  = 41DE    IX  = 42A0      IY  = 4380
```

2. S命令　如果DEBUG原来不在S显示方式，执行S命令以后就使DEBUG进入S显示方式。

S显示方式把X显示方式的第一显示区基地址以下完整的256字节（一页）显示出来。S显示方式一次可以显示256字节的内存储器区，总共使用16行把这些内存储器区的内容都显示出来。

3. D命令　确定显示区的命令

格式　　[n] D [addr 1]

如果现在处在S显示方式，打入上述命令后，就会显示包括addr 1地址在内的256字节的内存内容；如果给定了参数n，就会改变由n参数指定的区（指X显示方式中、三个64字节内存区中的某一个）的基地址，因为DEBUG正处于S显示方式，因而当前的屏幕显示并不改变。

如果是在X显示方式，addr 1就成了参数n所指定的显示区的基地址（基地址就是用户设定的开始显示的单元地址）。如果不给定参数n，就认为指定的显示区是第一区；如果n=2，就是第二区；n=3，就是第三区。例如：

（1）D7080（不指定n，addr 1=7080H）

如果DEBUG处在S显示方式，就显示7000H—70FFH单元的内容。

如果DEBUG处在X显示方式，则在三个独立的显示区中的第一个显示区将显示7080H—70BFH单元的内容。

（2）3DFFC0（n=3，addr 1=FFC0H）

如果DEBUG处在X显示方式，则三个独立的显示区中的第三区将显示FFC0H—FFFFH单元的内容。

如果DEBUG处在S显示方式，则把新的第三显示区的基地址保存起来，但是目前不改变荧光屏上的显示。

4. [n]；把内存储器显示区的位置移向较高的区域。

如果在S显示方式，并且不指定参数n，当打入';'时，则S显示就要往高移到接着的一个256字节的内存储器区。

如果处在X显示方式，则要把参数n规定的64字节显示区移高64字节。当不指定n时，就把第一显示区移高64字节；如果n=2，就移动第二显示区；n=3，则移动第三显示区。

5. [n]－　显示较低的内存储器区的内容

如果处在S显示方式，并且不指定参数n，当打入'－'时，就会重新显示较低的256字节的内容。

如果处在X显示方式，就要把指定的64字节显示区下移64字节。不给定参数n，即规定第一显示区下移64字节；n=2，则下移第二显示区；n=3，则第三显示区下移。

6. M[addr 1]　进入修改存储器内容的工作方式。

如果原来不在DEBUG程序的S显示方式，打入这个命令就转入S显示方式，并显示包

括 addr 1 地址在内的 256 个字节的内容，并进入修改方式和显示一个闪烁的光标，只要把此光标移到待修改的数字下面，再按 0—9 或 A—F 键中的任一个，就可以修改这一个16进制数字及对应的存储器内容，同时光标向前移动一格。按→或空格键也能使光标向前移动一格，但不影响内存储器的内容。按←键则可使光标后退一格，该操作也不改变内存内容。同时按 SHIFT 键和←键，可使光标后退 4 个十六进制数字的位置，并不改变内存储器的内容；同样，按 SHIFT 键和→键可使光标前移 4 个 16 进制数字的位置而不改变内存内容。按↑可使光标上移一个显示行，按↓键可使光标下移一个显示行，这些操作同样也不影响内存储器的内容。前、后移动光标不得超过当前显示的 256 字节的显示区。按下 ENTER 键就能终止修改方式。按下某些其它的键也能终止修改方式，但是会引起错误。

举例

M6314

DEBUG 如果不在 S 显示方式，就先转入 S 显示方式，并显示出 6300H—63FFH 范围的内存储器的内容，闪烁的光标停留在 6314H 单元所含的第一个十六进制数字的下面。这时用户就可以打入新的数字来修改原有的内容，或者在所显示的256字节范围内移动光标。

7. F[addr1] [,hb1] [,hb2] [,hb3] [,hb4]在内存储器区搜索给定的字节。

从给定地址 addr 1 开始的内存储器区中寻找给定的一组十六进制数字表示的字节。参数 hb1, hb2, hb3, hb4 每一个表示一个字节的两位十六进制数字。只要给定 hb1, hb2, hb3, hb4 中的任一个，就必须给出 addr 1 参数。如果 hb1, hb2, hb3 和 hb4 这四个参数一个也没有给出，就采用上一次 F 命令所用的那一组参数。如果没有给定 addr1, 就使用上一次 F命令所找到的单元地址加 1 作为新的 addr 1, 这样, F 命令就能够为用户继续寻找与最初字节串相同的字节串。为了寻找所要的字节串，F 命令要对内存进行搜索。如果找到了，就把找到的第一字节的地址减去 20H 作为 X 显示方式第一显示区的基地址。这就使找到的字节串刚好在 X 显示方式的第 3 行行首。如果没有找到被寻找的字节串，就把 X 显示方式第一显示区的基地址设定为 0FFE0H。

举例：F5200, CD, 24,44

这个例子是要从内存器的 5200H 地址开始搜索，去寻找第一次出现 CD, 24, 44 这三个字节的单元。找到后，如果再次使用 F 命令，就会继续寻找下一次出现这三个字节的单元。

如果找到的字节串在当前的堆栈区，由于这时堆栈归 DEBUG 使用，因此有可能在这些字节尚未显示出来以前就被 DEBUG 破坏了，这就使用户感到 DEBUG 出了错误。在这里还要再说明一点，DEBUG 程序把用户要寻找的字节串副本保存在内存储器中51××H区域内，供搜索时比较用，因而当搜索到这一区域时，找到的字节串实际上是 DEBUG 程序作比较用的字节串副本，并非你想找的真正的字节串。

8. I 命令 单步执行用户程序

执行被中断程序的当前指令，然后重新进入 DEBUG 程序。这使用户可以单步执行程序。用户可以观察每一条指令执行后的种种变化。单步执行时要注意：

(1)在单步工作方式，允许执行一个完整的定时中断程序。

(2)如果指令位置低于 5200H 或者转移到、或者返回到低于 5200H 地址，则不允许执行单步操作。

(3)DEBUG 执行单步功能是在单条指令执行完成后用 Z80 指令 RST 30H 作为陷阱，以便返回到 DEBUG 程序。所以，要单步执行的指令不应该分枝到它本身，也不应该访问跟在这条要单步执行的指令下面作为源或目的数据的字节。

9. C 命令

执行的情况与 I 命令相同，只是要单步执行的指令是一条 CALL 指令，在这种调用类型的单步指令期间，可以执行整个的被调用程序。

10. R 命令 改变寄存器对的内容。

格式：

R [dreg] [,value 1]

把中断时寄存器对 dreg 的内容更换成 value 1 的值。

例：R DE, C000

把寄存器 DE 原来的值更改成 C000H

R HL', 7100

把寄存器 HL' 原来的值更改成 7100H。

11. L 命令 把磁盘扇区内容装入系统缓冲区。

格式： L [dn1] [,drs 1]

把装配在驱动器 dn1 上的磁盘的相对扇区 drs1 读进 DOS 的系统扇区缓冲区 (TRS-80 I 型为 4200H—42FFH 单元，Ⅲ型为 4300H—43FFH 单元)。然后 DEBUG 进入 S 显示方式，显示该缓冲区内这一扇区的内容。drs1 是一个十进制表示的相对扇区编号值。因为 DEBUG 并不检验 dn1 和 drs 1 等值是否正确，所以用户给定的 dn1 和 drs 1 值一定要正确。一旦该扇区的内容放进了缓冲区，用户就可以像对待内存储器中的其它数据一样，对它们进行处理，可以用 F 命令进行搜索，也可以用M命令去更改它们。然而，更改缓冲区中扇区的内容并不会影响磁盘上该扇区的内容；必须用 WR 命令才能把缓冲区中的扇区内容重新存回盘上。因为几乎所有 NEWDOS/80 系统程序在作磁盘读写操作时都要使用系统缓冲区，所以当中断发生在 DOS 中 (在这种情况下，中断地址通常低于 5200H 要特别注意 COPY, FORMAT 等等)，而且当用户想继续执行被中断的程序时，就不应该使用 L 或 WR 命令。

注意！如果通行字有效，将拒绝执行 L 和 WR 命令，并进入错误状态。

举例 L 1, 150

相对扇区编号是从 0 开始计数的，相对扇区编号 150，即第 151扇区。本例是要把目前装配在 1 号驱动器上的磁盘的第151扇区的内容装入系统扇区缓冲区。

12. WR 命令

格式 WR [dn1] [,drs1]

这个命令可以把系统扇区缓冲区 (TRS-80 I 型为 4200H—42FFH, TRS-80 Ⅲ 型为4300H—43FFH) 的内容写入到装配在驱动器 dn1 上的磁盘的相对扇区 drs1 内。在 L 命令中对参数的规定及限制同样适用于 WR 命令。如果指定的磁盘扇区是读保护的，则读保护也要写到盘上。注意!!!假如你指定了不正确的 dn1 和 drs1 值，就会把缓冲区的数据写到这个不正确的扇区上，给你自己增添许多麻烦。你必须反复核实所进行的一切操作！

举例 WR 1, 150

把系统扇区缓冲区当前的内容写到 1 号驱动器内盘片上的第151扇区。

13．Q命令　退出DEBUG程序

Q命令用于从DEBUG中退出，回到DOS READY状态。不再顾及以前的程序。如果系统原来是在DOS-CALL或MINI-DOS状态，则要清除相应的状态。

14．G命令　恢复各寄存器的内容

格式　G [addr1] [,addr2] [,addr3]

恢复各寄存器的内容并继续执行程序。如果指定了addr1地址，就从这个地址开始执行程序；如果没有指定这个地址，就要从PC寄存器指定的地址开始继续执行程序。如果给定了addr2地址，就在这个地址上设置断点，实际上就是用Z80单字节指令RST　30H去代替这个单元的内容，当程序执行到这一地址时，就重新进入DEBUG程序。断点处原来的内容并未丢失（当重新进入DEBUG时会恢复其内容），但是，从退出DEBUG去执行用户程序到遇到断点重新进入DEBUG这段时间内，断点处原来的内容是不能使用的。addr3是第二个断点的地址。当指定addr2地址时，可以不再指定addr1。addr2和addr3不允许小于5200H。

举例：

（1）G7000,8400,8425

在内存中的8400H和8425H单元设置断点并恢复各寄存器的内容，从内存的7000H单元开始执行程序。

（2）G

这个命令将恢复各寄存器的内容，并由PC寄存器内容所指定的内存单元开始执行程序。如果被中断的程序正在等待键盘输入（如DOS READY或BASIC　READY），则仍将等待键盘输入。虽然不再显示光标（因为DEBUG程序不记光标状态），用户照样可以用键盘输入。

4.2 MINI-DOS　缩小规模的操作系统

在主程序执行的过程中，用户经常希望打断一下主程序的执行，转而去执行一个或几个DOS库命令，然后再回来继续执行主程序，但是要注意，在打断主程序期间不应该改变主程序中的任何参数。NEWDOS/80提供了能满足这种要求的功能，称之为MINI-DOS。

为了使用MINI-DOS，必须满足下列条件：

（1）SYSTEM的选用参数　AB=N。

（2）SYSTEM的选用参数　AF=Y。

（3）允许中断或者主程序正在通过标准键盘输入子程序等待键盘输入和SYSTEM的选用参数AJ=Y。

满足这些条件，并同时按下'D'，'F'，'G'，键就会中断主程序的执行，保存所有寄存器的内容，并进入MINI-DOS状态。荧光屏上显示出MINI-NEWDOS/80 READY。注意！不允许在磁盘输入/输出时进行'DFG'联键操作，这样做会造成磁盘数据出错；使用MDBORT命令可以退出MINI-DOS状态，这不会带来任何不好的后果。

进入MINI-DOS状态后，用户可以执行除APPEND，CHAIN，COPY和FORMAT以外的其它DOS库命令。在MINI-DOS状态下不允许执行非DOS库命令及其它程序。

当你准备返回主程序时，可以打入DOS库命令MDRET。如果在按'DFG'键之前有光标显示，这时还将重显这个光标，恢复主程序各寄存器原来的状态，继续执行主程序。如果主程序被打断时正在等待键盘输入或者已经输入了部分记录，返回时虽然荧光屏没有显示，但

是这部分记录仍然保存在缓冲区内。如果中断时主程序正在等待键盘输入，不论是否打入了字符，在退出 MINI-DOS 时，主程序仍旧处于等待状态。不要有顾虑，请继续输入你的记录。如果中断时主程序没有等待键盘输入，退出时就继续执行主程序。

如果退出 MINI-DOS 状态后用户不希望继续执行被打断的主程序，只要打入 DOS 库命令 MDBORT，就能退出 MINI-DOS 状态并终止主程序的执行，这时，系统就返回到通常的 DOS READY 状态。

虽然在 MINI-DOS 状态不能使用 COPY 命令，但是用另一个 DOS 库命令 MDCOPY，可以完成一些简单的文件复制工作。

NEWDOS/80 不能防止在同时按多个键时（如联键操作 'DFG'）会有一个或两个键信息传送到主程序内。这种情况在 SYSTEM 的选用参数 AJ=Y 时特别容易发生。这些窜入的字符在退出 MINI-DOS 时通常在荧光屏上能够看到。用户应该用退格键←把这些多余的字符清除掉。另外，当主程序是一个文本文件时不要使用三键同时按的命令，以免引起文本出错。

下面是一个使用 MINI-DOS 的例子，系统先处在 DOS READY 状态，接着执行下列程序：

```
BASIC
10 PRINT "HELLO": GOTO 10
RUN
```

这时，上述 BASIC 程序将会在荧光屏上无休止地显示出 HELLO 字样。同时按下 D,F和 G 键，于是打断了这个 BASIC 程序的执行，荧光屏上将会显示出 MINI-NEWDOS/80 READY 信息。此时顺序执行以下的一些 DOS 库命令：

```
DIR 0
FREE
CLOCK
CLOCK, N
LIB
SYSTEM, 0
PDRIVE, 0
MDRET
```

最后的 MDRET 命令使之退出 MINI-DOS 状态，并在中断处继续执行 BASIC 程序。现在，我们想退出正在运行的试验程序进入 DEBUG。同时按下'1','2','3'键，这就再次打断了 BASIC 程序的执行。这时，DEBUG 程序起作用了，可以使用 DEBUG 程序的 X 或 S显示方式。再打入 G 命令和 ENTER 键，就可以退出 DEBUG，继续执行被打断的 BASIC 程序。同时按下 'D', 'F', 'G' 键，再次进入 MINI-DOS 状态。这次我们使用 DOS 命令MDBORT。它将强迫撤销被中断的 BASIC 程序，并退出 MINI-DOS 状态，转入通常的 DOS READY 状态。

4.3 CHAINING 链接状态

DOS 命令 CHAIN 和 DO 只是在写法上有些不同，基本功能是完全一样的，所以在这一节只讨论 CHAIN 命令，这些情况同样也适用于 DO 命令。

对于大多数 TRS-80 的用户来说，有些工作经常要用到同样的一串 DOS 命令或对程序作出相同的一些响应，这就需要用户通过键盘打入一些信息，可以设想，如果能把这一串键盘输入的字符保存在一个磁盘文件中，当用户想要执行这些特定工作时去调用这个磁盘文件，就可以省去许多键输入工作，从而节省了操作时间，并避免了一些键输入中可能出现的错误。

例如，假定每次复位或打开电源后，操作员必须打入下面的一些命令和程序响应：

HIMEM, 0E800H　执行 DOS 命令 HIMEM。

PROGRAM1　执行程序名为 PROGRAM1 的程序。

Y　对 PROGRAM1 程序第一次询问的响应。

50　对 PROGRAM1 程序第二次询问的响应。

PROGRAM2　PROGRAM1 程序完成后，要执行 PROGRAM2 程序。

1　对 PROGRAM2 程序第一次询问的响应。

WORKF1　对 PROGRAM2 程序第二次询问的响应。

WORKF2　对 PROGRAM2 程序第三次询问的响应。

BASIC, RUN "BASPGM1/BAS"

PROGRAM2 完成后，要进入 BASIC，并执行 BASIC 程序 BASPGM1。

Y　对 BASPGM1 程序第一次询问的响应。

假定每次执行时往 BASPGM1 程序送的应答信息是不同的，则它不属于链的组成部分，与我们这里讨论的问题无关。这里所讨论的是每次必须从键盘输入的一组相同的信息。

如果把这一组顺序输入的键盘字符放置在一个磁盘文件(如文件名为XXX/JCL 的文件)内，则只要执行 DOS 命令：

CHAIN, XXX/JCL

就可以启动这一组键盘输入序列。

执行这个 CHAIN 命令（见 2.9 节），就可以从文件 XXX/JCL 中取得键盘输入，并从这个文件的首行开始，在 DOS 要求的时候、或者执行某一程序发出询问的时候，像键盘输入一样发送字符。按照要求发送字符，一直到文件结束，然后把键输入转换成由真正的键盘输入。所以当你把要执行的命令放入 CHAIN 命令后就可以舒舒服服的休息，一直到 BASPGM1 程序收到了第一个响应信息以后，再打入各种需要的命令或应答信息。

另外，我们假定每次复位或开机时均要引用上例中的这一系列键盘输入，这时甚至连这个 CHAIN 命令都可以不打，只要使用 DOS 命令 AUTO（见 2.4 节）预先设置这个 CHAIN 命令就可以了。为此，执行如下的 DOS 命令：

AUTO, CHAIN, XXX/JCL

现在只要复位或开机，就会自动执行 CHAIN 命令，在 BASPGM1 程序收到它的第一个应答信息以前，操作员不必作任何操作。

这种从一个磁盘文件中取得键盘输入的过程和 NEWDOS/80 这时所处的工作方式总称为链接状态。包含有键盘字符序列的文件称为链接文件（如上例中的 XXX/JCL)。

NEWDOS/80 本身并不提供建立链接文件的手段，它只是在对 CHAIN 命令响应时用一下这种链接文件（见 2.9 节)。在链接文件中要包含什么样的键盘字符序列应由用户决定，让用户按其需要来建立链接文件。使用 SCRIPSIT 程序或 PENCIL 程序建立链接文件大概是最

简便的方法，链文件建立以后，以ASCII码的形式把建立的文件保存到磁盘上。对于既没有SCRIPSIT程序也没有PENCIL程序的用户，可以使用 NEWDOS/80 系统盘中已有的、名字为 CHAINBLD/BAS 的 BASIC 程序来建立和编辑简单的链接文件。为了建立通常键盘上未提供的字符所组成的链接文件，必须使用另外一些链接文件的建立程序。

要建立链接文件的用户必须记住：除了 /./ 型式的链接控制记录（下面要讨论）以外，链接文件必须完全由 DOS 或者当前运行的程序所能识别的命令级键盘字符序列所构成。不能凭你的猜测来进行链接工作。

在链接文件的处理中，NEWDOS/80 根据 SYSTEM 选用参数 AT 决定的两种方式中的一种来工作。

（1）如果 SYSTEM 选用参数 AT=Y，则所有需要经标准键盘子程序送入的键盘输入均来自链接文件。此方式应用于两种情况：一种是需要一个记录（如 BASIC 中的 INPUT 或 LINEINPUT），另一种是需要一个字符（如 BASIC 中的 INKEY $）。

（2）如果 SYSTEM 选用参数 AT=N，那么只有需要经标准键盘输入子程序（位于ROM中的 0040H 单元）送入的整个一批记录（如 BASIC 中的 INPUT 或 LINEINPUT）来自链接文件。而需要的单个字符（如 BASIC 中的 INKEY $）则来自于键盘。

在 NEWDOS/80 2.0 版本的盘片上，为用户提供了：

（1）BASIC 程序 CHAINBLD/BAS，用户能够用它建立简单的链接文件。

（2）名字为 CHAINTST/JCL 的示范性链接文件。

关于 CHAINBLD/BAS 程序的使用说明请看本书的6.6 节。现在我们要用 CHAINBLD/BAS 程序来看一下链接文件 CHAINTST/JCL。使计算机处于 DOS READY 状态，通过键盘打入：

```
BASIC  RUN  "CHAINBLD/BAS:0"
```

开始执行 CHAINBLD/BAS 程序。

2	在 CHAINBLD/BAS 程序中，2 表示进行文件装入操作。
CHAINTST/JCL:0	待装内存储器的文件的文件标识符。
L;	在 CHAINBLD/BAS 程序中，打入 L; 表示要列出链接文件的第一页。
;	列出链接文件的下一页。
U	返回到编辑子功能。
Q	返回到显示 CHAINBLD/BAS 程序功能的主清单。
5	退出 CHAINBLD/BAS 程序。

执行每一步时，请仔细地研究一下屏幕显示的是什么内容。这个示范性链接文件内有命令、程序响应和链接控制记录的一些比较好的例子。注意：在 CHAINBLD 程序进行初始化的10秒钟内不给出任何显示信息。当你仔细地看过这个链接文件以后，就可以返回到 DOS 状态，并使用 DOS 命令：

```
CHAIN, CHAINTST:0
```

执行这个链接文件。

因为大多数链接字符序列比较短（一般不多于 100 个字符）所以为每一个这样的字符序列分配一整个 gr.（共 1280 字节）实在太不合适。因此 NEWDOS/80 允许把一个链接文件分

成若干个段(section),组成键盘序列的每一个段前面都有段识别记录、请看下面/./0的说明),但是链接文件的第一段不需要有段识别记录。如果由 CHAIN 命令存取的链接文件段的前面有一个段识别记录，则 CHAIN 命令不仅要指定文件，还必须指出这个段识别记录。

在链接状态，当碰到文件结束或段结束时，NEWDOS/80 不给出任何信息就终止链接状态，并把键盘输入置回到通常的键盘子程序。如果执行了 DOS 命令 CHNON, N 或链接状态/./5 控制功能，也可能发生上述情况。如果当前的程序正在等待输入，则除了一切活动将停止以外，操作员不会感觉到这种变化。一般来说，操作员知道链接状态终止后首先会显示什么东西，并为此作好了准备。

如果在链接状态中，DOS发现有错误，将终止链接状态,并显示出 CHAINING ABORT（取消链接）信息，以便通知操作员。

如果在链接状态执行 CHAIN 命令，则链接将忽略前一个文件,而在新文件中开始链接，新文件很可能就是原来的文件和原文件中的一段。这就是说，CHAIN 命令没有嵌套能力，在链接状态也没有 RETURN 功能。在链接状态，DOS-CALL 是合法的。

在链接状态有五种方法更换键盘字符序列。

（1）当前正在执行的程序可以经 DOS-CALL （如 BASIC 中的 CMD "doscmd"）执行 CHAIN 或 CHNON 命令。

（2）CHAIN 命令本身就可以是链接文件的一个组成部分。然而，要执行 CHAIN 命令，DOS必须是正在等待它的下一个命令，或者是当前正在执行的程序必须是设计得很巧妙的，使得足以检测出正常处理记录中的 CHAIN 命令记录，然后经 DOS-CALL (BASIC中的 CMD "doscmd")去执行 CHAIN 命令。

（3）最容易的方法是让链接文件在要改变键盘字符序列的地方安排一个/./4型式的链接控制记录（有关内容下面还要介绍)。使用 /./ 控制记录使得要更改的链接序列可以不管是在DOS控制下还是在用户程序控制下，屏幕上不给任何提示信息就改变链接序列。这种更改序列的方法有局限性，即链接工作不能转移到另一个不同的文件内进行。

（4）DOS 命令CHNON(见2.10节)可以是链接文件的组成部分。不要忘记，DOS必须是正在等待它的下一个命令，只有这时，CHNON 命令才能起作用。如果规定了 CHNON, N, 就要使链接状态无效(可是并未关闭链接文件，而且还保存着文件的位置供以后的 CHNON, Y, 或 CHNON, D 命令使用)，接着的键输入就来自键盘。如果规定了 CHNON, Y,同时 DOS-CALL 有效，则链接状态继续下去，并退出当前的 DOS-CALL 级。

（5）/./5型式的链接控制记录(有关内容下面要介绍)可以用在链接文件中以代替DOS命令 CHNON。即使 DOS 不是在等待它的下一个命令，也可以执行/./5控制记录的功能。

如果 CHAIN 命令是经 DOS-CALL （如 BASIC 中的 CMD "doscmd"）执行的，编程者必须记住 DOS 仍然在 DOS-CALL 状态中，目前正在执行链接文件中的 DOS 命令，这种情况一直到遇到文件结束、段结束 CHNON, N 命令或 CHNON, Y 命令为止（见2.10节)。这样一来，如果一个程序希望进入链接状态，而且要处理接着来的自己的链输入，则链接文件或链接文件段最前面的部分必须是 CHNON, N 或 CHNON, Y。

在链接文件中有六种控制记录。这些记录中的每一个必须由一个字符或四个字符组成的识别字符串开始，并以行结束字符 (ENTER) 为结束。在 NEWDOS/80 1.0 版本中，只用一个字符记录的识别符；而在 2.0 版本中，推荐使用四个字符记录组成的识别符，因为这四个

字符都是属于ASCII码标准字符，是可以打印和显示的，所以这四个字符在链接文件建立或编辑期间可以看得见。但是，在链接工作期间，不显示控制记录的识别字符。这些控制记录可使链接完成各种不同的工作。下面介绍每一个控制记录的作用，前面给出的是四个字符组成的识别符（在NEWDOS/80 2.0中建议使用的），后面是相应可选用的单个字符的识别符（在NEWDOS/80 1.0中使用）之值。

（1）/./0或一个字节＝128（即80H）。这个识别记录可以区分一个段，它必须是链接文件段的第一个记录，例外情况是：在未命名的文件中的第一段可以不放这个识别记录。链接控制记录的其余部分是段识别字符，它用来与CHAIN命令的段识别字符作比较（如果你已经规定了这个识别字符）或者与/./4控制记录中规定的段识别字符作比较，看它们是否一致。接着一直到EOF为止的文件字符或者下一段识别（ID）记录之前的文件字符都被看作为这个新段的一部分。举例：

/./0XXXXXX 标志下面的字符属于链接段XXXXXX。

（2）/./1或一个字节＝129（即81H）。可以使链接控制记录的其余部分显示出来，然后系统等待用户按ENTER键，以便继续执行。这是一种暂停功能。举例：

/./ 1 MOUNT WORK DISKETTE 显示信息 MOUNT WORK DISKETTE（装配工作磁盘），后面接着显示 PRESS "ENTER" WHEN READY TO CONTINUE（当准备就绪要继续工作时，按ENTER键）。然后DOS等待用户按ENTER键。

（3）/./2或一个字节＝130(即82H)。这个控制记录的功能是忽略链接记录的其余部分。它使链接文件的建立者可以把一些注释，说明放在文件中，链接过程中，这些说明是不显示的，也不起作用。在编辑过程中可以显示出来。

（4）/./3或一个字节＝131（即83H）。它使链接控制记录的其余部分要在荧光屏上显示出来，但是链接工作不会停止下来。这种功能使链接文件的建立者可以通过屏幕的显示告诉操作员，在链接过程中所发生的情况。举例：

/./3 PHASE TWO COMPLETED 显示 PHASE TWO COMPLETED 信息。DOS并不停下来等待，而是继续处理链接文件的输入。

（5）/./4或一个字节＝132（即84H）。链接控制记录的其余部分是31个字符或少于31个字符组成的链接文件段的识别符。搜索当前的链接文件，寻找与这个/./4控制记录中指定的段识别符一致的链接段。当找到时，链接就在这个段的第一个字符处继续。如果找不到一致的链接段，就会显示出错误信息 END OF FILE ENCOUNTERE（遇到文件结束），并取消链接状态。举例：

/./4 XXXXXX 接下来的链接处理字符要转入到目前所在链接文件中的名字为XXXXXX的链接段中去进行（请看上面/./0的例子）。

（6）/./5或一个字节＝133（即85H）。链接控制记录的其余部分可以是字符Y，N或D中的任一个。使用这种一个字符的参数就可以完成CHNOH的功能。/./5功能比CHNON命令优越的地方是DOS不必正处于等待它的下一个命令的状态。而缺点是链接状态的改变难以捉摸。所以/./5控制记录不适合初学者使用。举例：

/./5 N 虽然该文件尚未关闭，但是链接不能进行。

/./5 Y 链接虽然有效，但是要退出当前的DOS-CALL级（如果是在DOS-CALL级的话）。

当初次建立链接文件时，将会感到上面所介绍的链接控制记录都不太容易使用，但是多使用几次后就会慢慢地熟练起来。可以试着用/./3型式的控制记录去显示注释说明，用/./1型式的控制记录去显示信息和暂停执行，等待用户按 ENTER 键。最后，可以试用/./0 型式的控制记录把一个链接文件分成许多链接段，然后用/./4型式的控制记录在一个链接文件中进行链接的分支（转移）。

链接文件的建立者要保证对于系统或用户程序不设置那种不合法的链接状态。

在链接期间和 SYSTEM 选用参数 BC=Y 时，操作员可以按下⊡键来终止链接过程，操作者还可以用按下⊡键去强迫链接暂时停止，当然，操作者也可以用按下 ENTER 链来恢复链接。

4.4 DOS-CALL DOS 调用

NEWDOS/80允许机器语言程序调用 4419H 单元的 DOS 子程序（见3.11节）以便执行一个 DOS 命令或者用户程序。这种能力就称为 DOS 调用 (DOS-CALL)。BASIC 就是使用 DOS-CALL 执行 CMD "doscmd"功能的。

进行调用的程序在缓冲区内建立一个 DOS 命令，并用 0DH 字节来终止这个 DOS 命令。寄存器 HL 指向这个命令，调用 4419H 单元的 DOS 子程序，先把命令移到缓冲区，并把小写字母转换成大写形式，然后让 DOS 去执行这个命令。

如果 DOS-CALL 正在执行的是一个用户程序，DOS 不检查进行调用的程序与被调用的程序间是否有冲突的地方，因而，为了使这两个程序之间不发生任何冲突，编程者必须认真地检查。在 DOS-CALL 状态执行用户程序的一个例子是在 BASIC 状态中 通 过CMD "SUPERZAP" 功能去执行 SUPERZAP 程序。

此外，不能使用寄存器在进行调用的程序和被调用的程序之间来回传送参数。但是在进入被调用的程序时，寄存器 HL 是指向命令参数的。还有在 4403H—4404H 单元内的两个字节可以用于来回传送两个字节的参数。

在 DOS-CALL 状态下工作的一个用户程序自己就可以使用DOS-CALL （但是使用时必须特别小心，别使堆栈溢出）。DOS-CALL 是可以嵌套使用的，每次调用都工作于一个新的 DOS-CALL级。

从 DOS-CALL 返回时，进行调用的程序必须检查三个状态位，如果进位标志为1，则表示发生了一个错误，错误信息已在荧光屏上显示出来了。如果这个程序要继续执行，你就要决定下一步该做些什么。如果该程序要终止，并且该程序是由 DOS-CALL 调用的，它就应该通过转移到 4430H，然后退出，回到下一级较高的进行调用的程序，这时 C 状态标志位是1。

但是，如果返回时的状态为 NZ 和 NC，则表示发生了一个 DOS 错误，这个错误信息到目前为止还没有在荧光屏上显示，错误代码放在寄存器 A 的右边 6 位（位 6 和位 7 为 0)。如果进行调用的程序要继续执行，可以使寄存器 A 的位 7 为1，并调用 4409H 子程序退出，这时将显示出错误信息。否则，使寄存器 A 的位 7 为 0，并转移到4409H退出。后一种情况将使错误信息在荧光屏上显示出来，如果该程序不是由 DOS-CALL 调用的，则系统就要回到 DOS READY 状态，如果该程序是由 DOS-CALL 调用的，则不显示错误信息，退出后回到更高一级的调用程序，寄存器 A 不改变，同时 NC 和 NZ 状态标志位均为 1。

如果返回时的状态为 NC 和 Z，表示正确地完成了被调用功能。因为除寄存器 AF 以外的所有寄存器在 DOS-CALL 入口时要保存起来，在退出 DOS-CALL 时要恢复成原来的值，所以只有一种办法可以把参数带回来，那就是使用 4403H 和 4404H 单元的两个字节。实际上，DOS 命令缓冲区最高部分中不使用的那些字节（4318H—4367H）也可以用于与 DOS 进行通讯，但是用户不应忘记在执行 DOS 命令之前，DOS 要把所有的命令移到这个缓冲区，所以使用时要小心。

4.5 JKL　屏幕硬拷贝

NEWDOS/80 有一个子程序可以把荧光屏上显示的信息送到打印机打印。这就使用户能够把显示后即将消失的那些信息保存下来。

为了使用 JKL 功能，必须满足下列条件：

（1）SYSTEM 的选用参数 AD=Y。

（2）或者是开中断，或者是主程序正在等待经标准键盘子程序来的键盘输入和 SYSTEM 选用参数 AJ=Y。

（3）当前 DOS 必须不在使用它的覆盖区（内存中 4D00H—51FFH 单元）。

（4）DOS 必须不禁止使用覆盖区，也就是说覆盖区是开放的。

满足上述条件并且同时按下 J，K 和 L 键，就会中断主程序，保存现场，把显示内容原封不动地（可打印字符的范围由 SYSTEM 选用参数 AX 设定）转送到打印机去打印。如果打印机没有准备就绪，系统将进入循环，等待打印机准备就绪，此时不给用户任何提示信息。

对于由 SYSTEM 选用参数 AX 所定义的不可打印的字符，JKL 将用一个圆点来代替。

除了 CPU 为了等待打印机而处于挂起状态以外，按下 BREAK 键可以终止 JKL 功能。

当转送完成以后，恢复被中断的程序。由于 JKL 联键操作而造成的不需要字符的输入问题（在 4.2 节中有说明），这里同样存在。

在 NEWDOS 的较早版本中，JKL 子程序始终驻留在主存储器中，要占用一定量的内存储器区，因此不太好。在 2.0 版本中把 JKL 子程序勉强地移到了系统覆盖程序中，这样一来，这个子程序在某些情况下原来可以用，而现在就不能使用了。例如，在 DEBUG 状态，就不能引用 JKL 功能。

4.6 异步执行

NEWDOS/80 像 TRSDOS 一样，只有极有限的异步操作能力。它是通过把一个用户中断子程序插入 DOS 25 毫秒中断链来实现异步操作的。为了把用户的中断子程序插入链中，对于 TRS-80 I 型必须使用 4410H 单元的 DOS 子程序（见3.8 节），对于 III 型必须使用447BH单元的 DOS 子程序，为了从链中除去用户子程序，必须使用 4413H（见 3.9 节）单元的 DOS 子程序。关于用户中断子程序的格式要求和如何引用用户子程序请参看3.8和3.9这两节。

用户要注意，在 NEWDOS/80 中使用的用户中断子程序与 TRSDOS 中使用的是不兼容的。

第五章　DOS模块、数据结构和有关内容

这一章要详细介绍 NEWDOS/80 磁盘的模块、磁盘的目录和文件控制块 (FCB) 等方面的内容。这是为了使有兴趣研究 NEWDOS/80 2.0 操作系统的读者获得有关系统模块和数据结构方面的第一手资料，以便进一步剖析它，分析其优缺点，并加以改进或移植到其它型号的微计算机中去。同时，对于广大的 NEWDOS/80 2.0 操作系统的使用者来说，学习这方面的知识也是有好处的，因为磁盘经反复、频繁的使用，以及偶然的干扰和硬件的故障，都可能破坏有用数据，另外，操作的不小心也可能删除有用的文件，这些都会给使用者带来不可弥补的损失，但是当掌握了本章介绍的内容以后，一些“不可挽回”的损失就有可能得到补救，5.10节将介绍这方面的内容。初次使用 NEWDOS/80 的新用户应该先读一下 5.1节和 5.4 节，其余部分可以以后再看。

5.1　基本文件

NEWDOS/80 使用的磁盘（包括系统盘片和不带系统的数据盘片）上必须具有下列两个文件：

1. DIR/SYS　占 2—6 个 gr.。它是磁盘的目录文件。每个供 NEWDOS/80 使用的盘片必须有这个文件，因为它包含有这个盘片上所有文件的控制信息。FORMAT 命令或 COPY 命令中格式化操作部分能自动建立这个文件，DOS 可以根据需要来修补这个文件，也就是说，可以增加、更改或删除盘片上这些文件的控制信息。具体的目录结构在 5.6 节中介绍。在 NEWDOS/80　1.0 版本中，DIR/SYS 的散列代码是错误的，关于 DIR/SYS 的 HIT 扇区代码的修正可参看 5.6 节中的 2.。

2. BOOT/SYS　占 1 个 gr.。它必须占据每个盘片的第一个 gr.。在数据盘片上，这个文件仅用于禁止把这个盘片放在 0 号驱动器上作引导用。在系统盘片上，BOOT/SYS 文件的第一扇区包含有当打开电源、复位或转移到 0 单元时从装在 0 号驱动器中的盘片上把 DOS 装入机器用的目的代码。在 NEWDOS/80 系统盘片上，BOOT/SYS 文件的第二扇区是第一扇区的副本 (TRS-80　Ⅲ型引导时要用到它)，而第三扇区则放有系统控制信息，这些信息是通过 DOS 命令 SYSTEM 和 PDRIVE 来设定的。FORMAT 命令或 COPY 命令的格式化操作部分会自动建立 BOOT/SYS 文件。

5.2　NEWDOS/80 的 DOS 系统模块

NEWDOS/80 是一个单用户磁盘操作系统，它由 14 个程序模块组成，分别在三个区中执行。常驻模块 SYS0/SYS 驻留在从 4000H 到 4CFFH 的整个非数据区。模块 SYS1/SYS—SYS5/SYS, SYS7/SYS—SYS9/SYS 和 SYS14/SYS—SYS17/SYS 全部共享覆盖区 4D00H—51FFH (一次只能有一个模块在这个区中)。SYS6/SYS 从覆盖区和 5200H—6FFFH 这两个区中执行。下面列出这些系统模块的分布情况及其功能的简要说明。

系 统 模 块	分　　布	功　　能
SYS0/SYS	4000H—4CFFH (常驻)	系统核心，处理DOS初始化，磁盘I/O，时钟中断及装入其它模块等功能
SYS1/SYS	4D00H—51FFH (覆盖)	查询DOS 命令，键盘扫描
SYS2/SYS	4D00H—51FFH (覆盖)	建立文件，打开 FCB 及处理 RENAME 和 LOAD 命令
SYS3/SYS	4D00H—51FFH (覆盖)	关闭 FCB，删除文件，处理 BREAK, BLINK, CLOCK 等命令
SYS4/SYS	4D00H—51FFH (覆盖)	显示 DOS 系统的错误信息
SYS5/SYS	4D00H—51FFH (覆盖)	DEBUG 功能
SYS6/SYS	4D00H—6FFFH	处理 FORMAT, COPY 和 APPEND 命令
SYS7/SYS	4D00H—51FFH (覆盖)	处理 TIME, DATE, AUTO 等命令
SYS8/SYS	4D00H—51FFH (覆盖)	处理 DIR, FREE 等命令
SYS9/SYS	4D00H—51FFH (覆盖)	处理 BOOT, CHAIN 等命令
SYS14/SYS	4D00H—51FFH (覆盖)	处理 CLEAR, LIST 等命令
SYS15/SYS	4D00H—51FFH (覆盖)	处理 FORMS 和 SETCOM 命令
SYS16/SYS	4D00H—51FFH (覆盖)	处理 PDRIVE 命令
SYS17/SYS	4D00H—51FFH (覆盖)	处理 WRDIRP 及 SYSTEM 命令

各系统模块的详细说明：

* SYS0/SYS　占据 3 个 gr.。DOS 的这个常驻模块是由系统引导程序 (BOOT/SYS) 从盘上装入内存的。永久驻留在内存中，成为常驻模块，它不包括覆盖区中的 DOS 初始化子程序，当不再需要这个初始化子程序时它是要被覆盖的。SYS0/SYS 要处理 DOS 的初始化，磁盘 I/O，时钟中断，装入其它的系统模块和管理键盘等。它是整个系统的核心部分。

* SYS1/SYS　占据 1 个 gr.。它查询 DOS 命令，完成键盘命令的扫描工作。

* SYS2/SYS　占据 1 个 gr.。它的功能是建立文件，打开 FCB，分配文件空间，分配 FDE、对通行字编码和装入要执行的用户程序。还包括处理库命令 RENAME 和 LOAD 的程序。

* SYS3/SYS　占据 1 个 gr.。它的功能是关闭 FCB，删除文件，插入或删除25毫秒链中的定时子程序。处理库命令 BREAK, BLINK, CLOCK, DEBUG, JKL, LCDVR, LC, VERIFY 和 PURGE（部分功能）的子程序。

* SYS4/SYS　占据 1 个 gr.。显示 DOS 系统的错误信息。

* SYS5/SYS　占据 1 个 gr.。DEBUG 功能。

* SYS6/SYS　占据 7 个 gr.。在 4D00H—6FFFH 中执行。处理库命令 FORMAT, COPY, APPEND 的程序。

* SYS7/SYS　占据 1 个 gr.。处理库命令 TIME, DATE, AUTO, ATTRIB, PROT, DUMP, HIMEM 和 PURGE, SYSTEM 及 PDRIVE 命令的开始部分的程序。

* SYS8/SYS　占据 1 个 gr.。处理库命令 DIR 和 FREE 的程序。

* SYS9/SYS　占据 1 个 gr.。处理库命令 BASIC2, BOOT, CHAIN, CHNON, MDCOPY, PAUSE 和 STMT 的程序。把用户的逻辑子程序放入队列或者从队列中撤除（请看第三章中的 DOS 子程序 4461H 和 4464H)。

*SYS14/SYS　占据 1 个 gr.。处理库命令 CLEAR, CREATE, ERROR, LIST, PRINT和

ROUTE 的程序。

* SYS15/SYS 占据 1 个 gr.。处理库命令 FORMS 和 SETCOM 的程序。

* SYS16/SYS 占据 1 个 gr.。完成 PDRIVE 命令的大部分功能。

* SYS17/SYS 占据 1 个 gr.。处理命令 WRDIRP 的程序以及完成 SYSTEM 命令的大部分功能。

5.3 NEWDOS/80 的 BASIC 模块

NEWDOS/80 的磁盘 BASIC 与 TRS-80 的 ROM BASIC 相比增添了一个主常驻模块和 8 个覆盖模块。模块 SYS10/SYS—SYS13/SYS 和 SYS21/SYS 从 DOS 覆盖区 (4D00H—51FFH) 执行。模块 SYS18/SYS—SYS20/SYS 从 BASIC 覆盖区 (5200H—56FFH) 执行。所有的 BASIC 模块 (除 BASIC/CMD 以外) 是根据需要临时装入的，它们必须在系统盘片上，才能完成 BASIC 应有的全部功能。这里列出 BASIC 的各个模块、分布及其简要功能说明：

BASIC 模块	分　布	功　能
SYS10/SYS	4D00H—51FFH	执行 BASIC 语句 GET和PUT
SYS11/SYS	4D00H—51FFH	执行 BASIC 直接语句 RENUM
SYS12/SYS	4D00H—51FFH	执行 BASIC 直接语句 REF
SYS13/SYS	4D00H—51FFH	显示 BASIC 的错误信息
SYS18/SYS	5200H—56FFH	BASIC 接直语句的处理程序
SYS19/SYS	5200H—56FFH	执行 BASIC 语句 LOAD, RUN 等
SYS20/SYS	5200H—56FFH	处理许多磁盘 BASIC 语句
SYS21/SYS	4D00H—51FFH	处理 CMD "O"的程序

详细说明如下：

* BASIC/CMD 占据4个 gr.。这个主常驻模块放在 5700H 以上的区域。它完成磁盘 BASIC 的功能。这个模块不必放在系统盘片上，因为可以从数据盘片上调取它 (就像调用其它程序一样)，这个模块一旦被调用以后，就不再需要了，一直到下次再想进入 BASIC 时才去调用它。

* SYS13/SYS 占据 1 个 gr.。这个模块的功能是显示 BASIC 的错误信息和执行 RENUM 的第一部分程序。BASIC 工作的时候，这个模块就必须在系统盘片上。

* SYS12/SYS 占据 1 个 gr.。执行 BASIC 的直接命令 REF。如果要执行 REF，系统盘片上必须要有这个模块。

* SYS11/SYS 占据 1 个 gr.。执行 BASIC 的直接命令 RENUM。如果要执行 RENUM，系统盘片上必须要有这个模块。

* SYS10/SYS 占据 1 个 gr.。执行 BASIC 语句 GET 和 PUT。如果要执行这两个语句中的任意一个，这个模块就必须在系统盘片上。

* SYS18/SYS 占据 1 个 gr.。这是 BASIC 直接语句的处理程序，BASIC 工作期间，这个模块必须在系统盘片上。

* SYS19/SYS 占据 1 个 gr.。处理 BASIC 语句 LOAD, RUN, MERGE, SAVE和CMD

〃F〃 DELETE 的程序。BASIC 工作期间，在系统盘片上必须要有这个模块。

＊ SYS20/SYS　占据 1 个 gr.。处理许多磁盘 BASIC 语句的程序，当 BASIC 正在执行一个程序时，这个模块通常是驻留在主存中的。BASIC 工作期间，这个模块必须在系统盘片上。

＊ SYS21/SYS　占据 1 个 gr.。处理 CMD 〃O〃 的程序，如果要执行 CMD 〃O〃，这个模块就必须在系统盘片上。

5.4 NEWDOS/80 盘片上的其它一些模块

在 NEWDOS/80 系统盘片上还有一些附加的模块，它们是一些独立的实用程序模块，对于要进行软件开发和分析操作系统的用户是很有用的。它们是：

＊ DIRCHECK/CMD　这是一个检查目录错误，在荧光屏上列出或在打印机上打印目录清单的程序。详细情况请看 6.4 节。

＊ EDTASM/CMD　这是一个用于编辑和汇编 Z80 源程序，以及把目的程序存入磁盘或磁带（或从磁盘、磁带上装入内存）的程序。详细情况请看 6.5 节。

＊ DISASSEM/CMD　这是一个对 Z80 目的代码进行反汇编的程序。详细情况请看 6.2 节。

＊ LMOFFSET/CMD　这个程序可用于从磁盘或磁带上读取装入模块，以及把这些模块写入磁盘或磁带。这个程序可以随意地

（1）赋予新的装入地址；

（2）附加一段预先执行的辅助程序。把程序从装入区移动到执行区；

（3）编制没有 DOS 系统也可以执行的程序。详细情况请看 6.3 节。

＊ SUPERZAP/CMD　这个程序可以检查和修改磁盘或内存储器内容。对磁盘的操作可以是针对盘片的，或者是针对文件的。请看 6.1 节。

＊ CHAINTST/JCL　这是一个用 CHAINBLD/BAS 程序建立的示范性链文件。

＊ CHAINBLD/BAS　这是一个 BASIC 程序，用来建立和编辑简单的、按记录定位的、经 DOS 命令 CHAIN 或 DO 使用的链文件。

＊ ASPOOL/MAS　这是 Apparat 公司修改 H. S. Gentry 自动假脱机 (SPOOL) 程序后，供 NEWDOS/80 使用的一个程序。详细情况请看 6.7 节。

5.5 减小操作系统的规模

只要通行字不起作用，就可以建立一个缩小规模的系统，具体的做法是：把一个完整的 NEWDOS/80 系统盘片复制到一个新的盘片上，然后删除那些不需要的文件。一个具有打开、关闭文件能力的最小系统将包括10个 gr. (BOOT, DIR, SYS0—SYS4)。如果要使用 DEBUG 程序（包括 BASIC 的 CMD 〃D〃），则要加上 SYS5。在 5.2 节中给出了需要增加 DOS 库命令时必须增加的附加模块。如果要用 BASIC 语言，5.3 节给出了必须加入的 BASIC 模块，而 5.2 节给出了如果要经 BASIC 的 CMD 〃xx〃 语句执行 DOS 库命令必须加入的 DOS 模块。如果系统模块装入程序发现模块的目录项无效或者在装入过程中遇到错误，就会发生下列两种情况中的一种：

（1）如果 SYS4 是系统中的一个有效模块，就会通过转移到 4409H 子程序而显示 SY-

STEM PROGRAM NOT FOUND（系统程序没有找到）错误信息。

（2）如果经 4409H 子程序要转入 SYS4，但发现 SYS4 不在系统中，则要执行 Z-80的 HALT 指令，在 TRS-80 Ⅰ型中就要引起计算机的重新启动，而在 TRS-80 Ⅲ型中就会使计算机停止运行（这时用户必须手动复位计算机）。包括在系统模块范畴内的模块是 SYS1/SYS—SYS21/SYS。如果 BASIC 覆盖模块中有一个没有装入，用户就必须小心地执行 BASIC＊，以便取回这个 BASIC 程序的文本（关于 BASIC＊ 的说明，请看 7.3 节）。

注意！一旦系统文件已经从一个系统盘上删除，就不能用从另一个系统盘上复制过来的简单方法加以恢复。DOS 系统装入程序要求系统文件的主目录登记项 (FPDE) 是在目录中特定的 FDE 区间中，而且还要求所有系统文件的空间是用第一个范围元 (extent element—这是两字节的控制数据，规定了分配给文件的连续空间）来登记的。SYS0/SYS 必须占有它被删除以前所在的同一个 gr.，为了使系统能有效地工作，建议所有的其它系统文件也放在与原来相同的 gr. 内。在重新把 FPDE 安排合适以后，就可以用 DOS 命令 COPY 去复制这些系统文件的内容了。

5.6 磁盘的目录结构

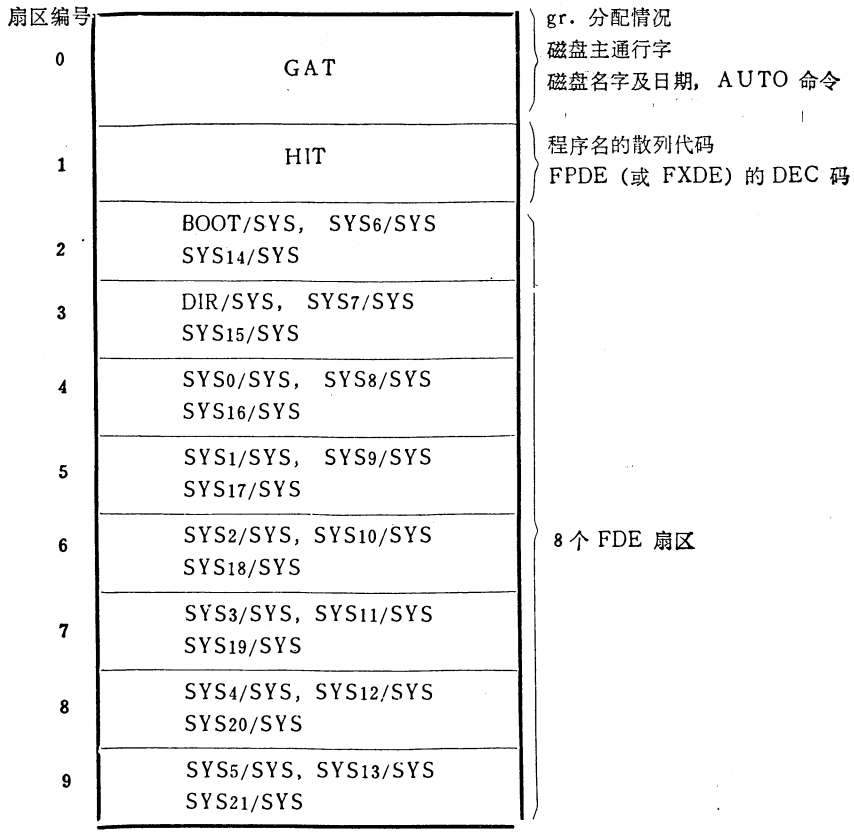
在 TRS-80 Ⅰ型中，若 NEWDOS/80 磁盘的目录文件 DIR/SYS 由 2 个 gr. 组成（请看 2.22 节 FORMAT 和 2.14 节 COPY 命令的 DDGA 参数），并设定每个磁道有 10 个扇区，每个 gr. 组有 2 个 gr. 和每个 gr. 有 5 个扇区（NEWDOS/80 的标准就是每个 gr. 包括 5 个扇区），则 NEWDOS/80 盘片和 TRSDOS 盘片是可以互换的。在这两种盘片上的某些文件可能是没有互换性的，而系统模块，BASIC, ELECTRIC PENCIL, SCRIPSIT 等是肯定不可互换的，但由上述程序管理的文件是有互换性的。

在 TRS-80 Ⅲ型中，NEWDOS/80 盘片和 TRSDOS 盘片的目录是不兼容的。TRSDOS Ⅲ型盘片不能直接供 NEWDOS/80 使用，同样，NEWDOS/80 盘片也不能直接供 Ⅲ 型的 TRSDOS 使用。如果 NEWDOS/80 单密度盘片的目录同 I 型标准的目录有同样的位置和容量，Ⅲ型的 TRSDOS 有一个变换程序，可以把数据复制到一个Ⅲ型的盘片上。NEWDOS 80 2.0 版本的 COPY 命令也有办法把Ⅲ型的 TRSDOS 1.3 版本（或以后更新的版本）盘/片上的一个、多个甚至于全部文件复制到一个 NEWDOS/80 的盘片上，同样，也能把 NEWDOS/80 盘片上的文件复制到 TRSDOS 的盘片上（请看附录 B.2 和 2.14 节）。

NEWDOS/80 除BOOT/SYS 和 DIR/SYS 占用的目录项以外，所有磁盘目录项 (FDE) 都可以供用户使用，所以新格式化后的数据盘上的 2 个 gr. 可以有 62 个文件目录项供使用。这是因为 2 个 gr. 等于 10 个扇区，GAT 和 HIT 各占一个扇区，剩余 8 个扇区，每个扇区可放 8 个 FDE，总共 64 个 FDE，扣除 BOOT/SYS 和 DIR/SYS 所占的 2 个 FDE，故尚还有 62 个FDE 可以使用。NEWDOS/80 在磁盘格式化时（请看 PDRIVE, FORMAT和COPY 命令的 DDGA 参数）允许目录有多达 6 个 gr.，所以最多可以提供 222 个可用的文件目录项。

盘目录文件总是起始于 gr. 组的分界处，并包含有 GAT 扇区（gr. 分配表），接着是 HIT 扇区（散列代码索引表），再后面是 8，13，18，23 或 28 个 FDE 扇区（请注意！不要混淆 FDE 扇区和 FDE，每个 FDE 扇区内可以容纳 8 个 FDE），FDE 扇区的数目取决于分配给目录的 gr. 数。例如，当设定 DDGA=4 时，就允许有 4 个 gr.（即 20 个扇区）

用于安放目录项，除去 GAT 扇区和 HIT 扇区以外，还有 18 个 FDE 扇区。用户可以使用 SUPERZAP 程序（请看 6.1 节）来研究目录的结构。每个盘片的第一扇区的第三个字节总是存放目录的起始 gr. 组的编号（以 16 进制表示），这个值是供 DOS 寻找目录用的。下面列出了目录结构简图。



目录结构及系统文件分布（目录文件占 2 个 gr.）

1. GAT（gr. 分配表）扇区

GAT 扇区（共 256 个字节）是目录中的第一个扇区，它记载着本盘片目前使用的情况，并包含如下内容：

（1）gr. 空闲/已分配表。GAT 扇区的相对字节 00H—5FH中每一个字节对应一个 gr.组（总共 96 个），每个相对字节内的每一位表示该 gr. 组内各 gr. 的空闲/已分配的状态。每个 gr. 组包含的 gr. 数目是由 PDRIVE 命令的 GPL 参数确定的，其数值处于 2—8 之间。表示 gr. 组第一个 gr. 的位是位 0（从右算起），表示第二个 gr. 的位是位 1，依此类推直到第八个 gr.。如果某状态位是 0，则对应的 gr. 是空闲的。如果状态位等于 1，表示这个 gr. 已被分配或者根本不存在。

（2）gr. 存在表。GAT 扇区的相对字节 60H—BFH 对应于相对字节 00H—5FH。如果字节中的某一位等于 0，那个 gr. 组的相应 gr. 是存在的，可以使用。如果等于 1，则相应的 gr. 不存在，不能使用，同时在 00H—5FH 中的对应位也必须等于 1。虽然 NEWDOS/80 在

格式化时建立了这些表征 gr. 是否存在的字节，但这样做的目的仅仅是为了能与老格式的 TRSDOS 磁盘兼容（在 TRSDOS 中这些字节称为封锁字节）。实际上，NEWDOS/80 决不会用 gr. 存在表去设置表示 gr. 不存在的状态。如有需要，可以废除 gr. 存在表，把这些空出的 GAT 扇区字节用作为 gr. 空闲/已分配表。

（3）为了使 GAT 扇区管理的磁盘空间总数为最大，NEWDOS/80 2.0 版本允许 GAT 扇区的 gr. 空闲/已分配区扩展到 GAT 的存在表区。这时，空闲/已分配状态字节就占据了 GAT 扇区内的相对字节 00H—BFH（以代替上述的 00H—5FH 区）。如果磁盘的 gr. 组数目超过了 60H（即96），则格式化时会自动完成这种扩展。

（4）磁盘的已编码通行字（2字节散列代码形式）在相对字节 CEH—CFH。

（5）磁盘的名字（最多允许有 8 个字符）在相对字节 D0H—D7H。

（6）磁盘的日期在相对字节 D8H—DFH。

（7）如果是一个系统盘片，复位时使用的 AUTO 命令放在相对字节 E0H—FFH。如果这个区内的第一个字节是 0DH (EOL)，说明这个系统盘片上没有 AUTO 命令。

下面列出一种80磁道 NEWDOS/80 2.0 系统盘片的 GAT 扇区排列图（设 1 gr. 组＝2gr.，故 80 磁道就对应于 80 个 gr. 组）。

相对字节	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	gr. 分配表（共96个gr. 组）
10	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
20	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	
30	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	
40	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	
50	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	
60																	gr. 存在表 可以当 gr. 分配表用
70																	
80																	
90																	
A0																	
B0																	
C0																	磁盘主通行字的散列代码
D0																xx	磁盘名字和日期
E0																	AUTO 命令
F0																	

GAT 扇区排列图

下面还将列出一个 80 磁道系统盘片的 GAT 扇区实例，这是在 DPS-85 数据分析系统（中国科学院声学研究所数字系统开发部生产）上，用 SUPERZAP 程序得到的，供对NEWDOS/80 数据结构感兴趣的读者参考，以后所举实例均是在 DPS-85 系统上进行的（1gr.组＝2 个gr.）。

相对字节

```
DRV 00  FFFF FFFF FCFF FFFF FFFF FFFF FFFF FFFF    ................
0   10  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF    ................
0H  20  FFFF FDFC FCFC FCFC FCFC FCFC FCFC FCFC    ................
```

```
    30  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC    ................
DRS 40  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC    ................
170 50  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF    ................
AAH 60  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC    ................
    70  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC    ................
TRK 80  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC    ................
17  90  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC    ................
11H A0  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC    ................
    B0  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF    ................
TRS C0  FFFF FFFF FFFF FFFF FFFF FF82 0000 E042    ..............B
0   D0  4E45 5744 4F53 3830 3030 2F30 302F 3030    NEWDOS8000/00/00
0H  E0  0D41 5349 430D 454C 4956 4552 492C 312C    .ASIC.ELIVERY,1,
    F0  320D 0DFF FFFF FFFF FFFF FFFF FFFF FFFF    2...............
```

图中左边各项含义：

DRV	驱动器编号	DRS	磁盘相对扇区编号
0	十进制值	170	十进制值
0H	十六进制值	AAH	十六进制值
TRK	磁道编号	TRS	磁道内相对扇区编号
17	十进制值	0	十进制值
11H	十六进制值	0H	十六进制值

其中 gr. 分配表区的值有如下含义：

FFH(11111111)——第一和第二 gr. 已分配（扇区 0—9）

FEH(11111110)——第二 gr. 已分配（扇区 5—9）

FDH(11111101)——第一 gr. 已分配（扇区 0—4）

FCH(11111100)——第一和第二 gr. 均空闲（扇区 0—9）

从上面的实例中可以看到：

（1）GAT 扇区的相对字节 CEH 和 CFH 放有通行字编码 E042，它是根据 PASSWORD 这个字算出的两字节散列代码。使用 SUPERZAP 程序 DPWE 功能可以计算通行字的散列代码。

（2）从相对字节 D0H 开始放有磁盘名字和日期，上例右边是各字节相应的 ASCII 码，这个盘片的名字为 NEWDOS80，日期设定为 00/00/00。

（3）在相对字节 E0H—FFH 放有 AUTO 命令。相对字节 E0H 是 AUTO 命令是否存在的标记字节，现在是 0DH（行结束字符），说明目前没有设置 AUTO 命令。

2. HIT（散列代码索引表）扇区

HIT 扇区是目录中的第二个扇区。这个扇区放的是散列代码 (HASH CODE)，它是与每个存放在盘上的文件名有关的。散列代码在 HIT 扇区中所处的位置还能告诉 NEWDOS/80 在目录中的什么地方有该文件的信息。每个存放在磁盘上的程序或文件都有一个 1 字节的散列代码。所以散列代码及其在 HIT 扇区中的位置都是极为重要的。它用作磁盘文件的 FPDE 的索引，还可表示哪些文件目录项(FDE)是空闲的，哪些是在使用。如果一个 HIT 扇

区字节等于0，则相应的FDE或者是不存在的，或者是空闲的。如果一个HIT扇区字节不为0，则相应的FDE已经在使用，另外，如果一个目录项作为一个FPDE使用，则HIT扇区字节的值是由FPDE的第六到第十六字节（文件名和文件名扩展符）的内容算出的散列代码。这样，当必须在目录中查找一个文件时，就要先计算散列代码，然后在HIT扇区内寻找与之一致的散列代码。如果找到一个与它相同的散列代码，就要根据这个代码在HIT扇区的位置去读一个相应的FDE扇区，并检查相应的FPDE，看一下文件名和文件名扩展符是否相同，因为不同的文件名和文件名扩展有可能算出的散列代码是相同的。如果不一致，就要继续搜索HIT扇区。

散列代码是用“异或”布尔表达式算出的，具体算法是：

$$HASH\ (x)=(...(((x_1*2)\oplus x_2)*2)\oplus.....x_n)*2$$

其中：$n=11$（因为文件名最多是8个字符，文件名扩展符最多是3个字符）

计算文件名和扩展符的1字节散列代码时要注意两点：

（1）算式中 $*2$ 表示 x_n 字符的相应ASCII码循环左移一位，即最高位（位7）要移到最低位（位0）。$\oplus$ 表示异或运算。

（2）当文件名不足8个字符时要在文件名后面填入空格（20H）凑满8个，扩展符不满3个字符也要填入20H，例如DIR/SYS，

$x_1=44H$，$x_2=49H$，$x_3=52H$，$x_4=20H$，$x_5=20H$，$x_6=20H$，$x_7=20H$，$x_8=20H$，

$x_9=53H$，$x_{10}=59H$，$x_{11}=53H$

注意！(/)符号的ASCII代码不参与计算。根据HASH (x)算式可以算出DIR/SYS的散列代码是C4H。

在SUPERZAP程序中的DNTH功能也可以计算这个散列代码，用户只要按照SUPERZAP程序的要求分别打入文件名和文件名扩展符，计算机就可以为你算出散列代码。

在HIT扇区中，HIT字节（即散列代码所在的字节）的相对位置精确地等于相应的FDE的目录项代码(DEC)，例如上述DIR/SYS的散列代码C4H在HIT扇区中的相对字节01处，于是这个DIR/SYS文件的FDE的DEC码就是01H。DEC码是寻找FDE的依据，1字节的DEC码格式是：

R	R	R	S	S	S	S	S

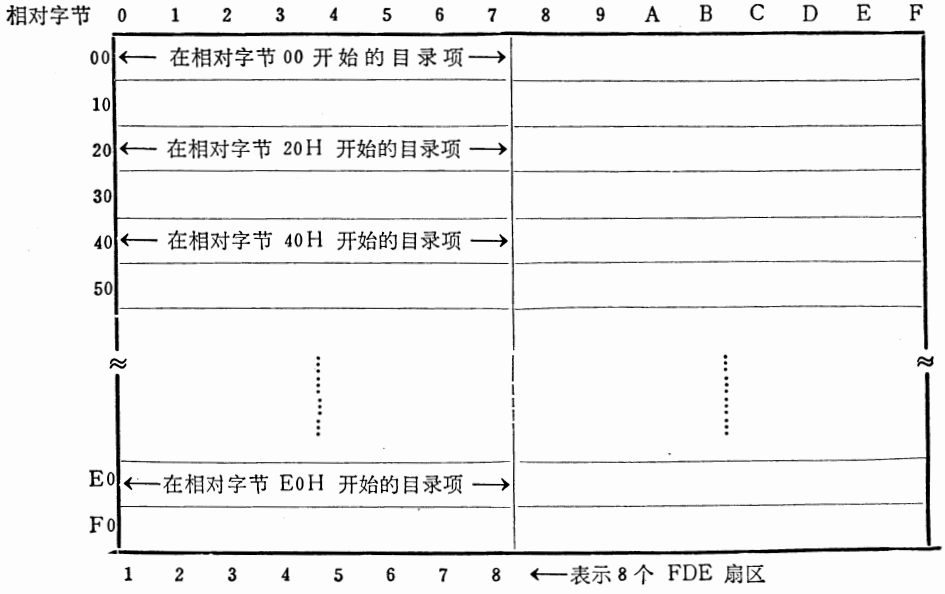
其中：SSSSS+2=目录中FDE扇区的相对编号

RRR×32=该扇区中FDE的相对字节地址

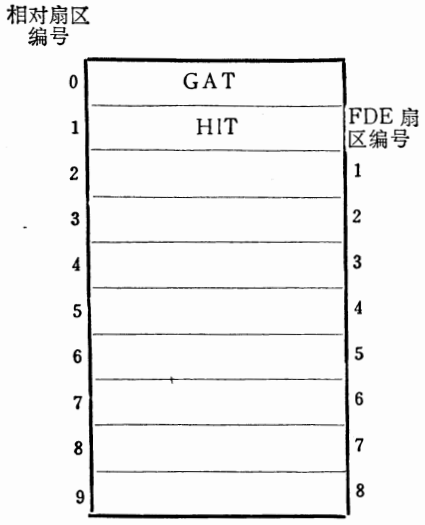
上面两个表达式中，SSSSS和RRR均要由二进制代码换算成十进制数后再运算。例如上述DIR/SYS文件FDE的DEC码是01H，所以它的FDE位于目录扇区的相对扇区1+2=3，相对字节 0×32=0，这可以从后面给出的FDE扇区实例的相对扇区3中得到证实。

因为DEC码可用来检索HIT扇区，所以当一个FDE被分配时，系统知道哪一个HIT字节要置成非零值，而FDE空闲时，系统知道把哪一个HIT字节置成0。

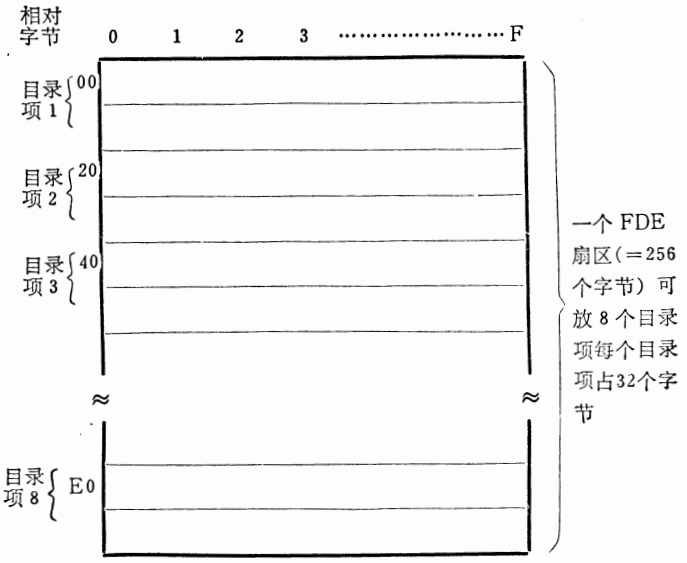
下面列出HIT扇区分布图和FDE分布图：



HIT 扇区分布图



FDE扇区分布图



目录项在FDE扇区内的分布

系统先根据文件名和扩展符算出它的散列代码，然后搜索 HIT 扇区的 HIT 字节，找到与它一致的 HIT 字节，即可算出待查找目录项的精确位置。

下面给出一个 HIT 扇区的实例：

```
DRV 00   A2C4 2E2F 2C2D 2A2B 0000 0000 0000 0000    .../,-*+........
0   10   0000 0000 0000 0000 0000 0000 0000 0000    ................
0H  20   2829 2627 27A7 26A6 0000 0000 0000 0000    ()&'.&..........
     30  0000 0000 0000 0000 0000 0000 0000 0000    ................
```

```
DRS 40   25A5 24A4 23A3 24A4 0000 0000 0000 0000    %.$.#.$.........
171 50   0000 0000 0000 0000 0000 0000 0000 0000    ................
ABH 60   F000 2AD3 0000 0000 0000 0000 0000 0000    ..*.............
    70   0000 0000 0000 0000 0000 0000 0000 0000    ................
TRK 80   0000 323B 0000 461F 0000 0000 0000 0000    ..2;..F.........
17  90   0000 0000 0000 0000 0000 0000 0000 0000    ................
11H A0   0000 0000 0000 EE00 0000 0000 0000 0000    ................
    B0   0000 0000 0000 0000 0000 0000 0000 0000    ................
TRS C0   0000 0000 0000 0000 0000 0000 0000 0000    ................
1   D0   0000 0000 0000 0000 0000 0000 0000 0000    ................
1H  E0   0089 0000 0000 0000 0000 0000 0000 0000    ................
    F0   0000 0000 0000 0000 0000 0000 0000 0000    ................
```

1 2 3 4 5 6 7 8 ←— 表示8个FDE扇区编号

例如要查出 BOOT/SYS 的目录项，先算出它的散列代码是 A2H，从 HIT 扇区中可以找到在FDE 扇区编号 1 处（即目录文件的相对扇区 2，请看上面给出的 FDE 扇区分布图），并可看到目录项在该扇区的相对字节 00H 处开始。接着可在目录相对扇区 2 的 00 字节开始的目录项中看到文件名和扩展符字节区（第 6—13 字节）确是 BOOT/SYS（文件目录项内各字节的意义请看 5.7 节）。这样，该目录项所列的信息就是代表文件 BOOT/SYS 的。下面把这个实例中的目录文件的各 FDE 扇区的内容列出来，供读者参考和练习。

```
DRV 00   5E00 0000 0042 4F4F 5420 2020 2053 5953    ^....BOOT....SYS
0   10   607F 1FB2 0500 0000 FFFF FFFF FFFF FFFF    ................
0H  20   5F20 0000 0053 5953 3620 2020 2053 5953    _....SYS6....SYS
    30   5678 1234 2300 1426 FFFF FFFF FFFF FFFF    VX.4#..&........
DRS 40   5F20 0000 0053 5953 3134 2020 2053 5953    _....SYS14...SYS
172 50   5678 1234 0500 0C20 FFFF FFFF FFFF FFFF    VX.4............
ACH 60   1820 0000 0042 4153 4943 2020 2043 4D44    .....BASIC...CMD
    70   782F 782F 1200 0203 FFFF FFFF FFFF FFFF    x/x/............
TRK 80   0000 0000 0000 0000 0000 0000 0000 0000    ................
17  90   0000 0000 0000 0000 0000 0000 0000 0000    ................
11H A0   0000 0000 0000 0000 0000 0000 0000 0000    ................
    B0   0000 0000 0000 0000 0000 0000 0000 0000    ................
TRS C0   0000 0000 0000 0000 0000 0000 0000 0000    ................
2   D0   0000 0000 0000 0000 0000 0000 0000 0000    ................
2H  E0   0000 0000 0000 0000 0000 0000 0000 0000    ................
    F0   0000 0000 0000 0000 0000 0000 0000 0000    ................
```

目录文件的相对扇区 2

```
DRV 00  5D00 0000 0044 4952 2020 2020 2053 5953    ]....DIR.....SYS
0   10  A71D F9E5 0A00 1101 FFFF FFFF FFFF FFFF    ................
0H  20  5F20 0000 0053 5953 3720 2020 2053 5953    _....SYS7....SYS
    30  5678 1234 0500 0D00 FFFF FFFF FFFF FFFF    VX.4............
DRS 40  5F20 0000 0053 5953 3135 2020 2025 5953    _....SYS15...SYS
173 50  5678 1234 0500 0C00 FFFF FFFF FFFF FFFF    VX.4............
ADH 60  0000 0000 0000 0000 0000 0000 0000 0000    ................
    70  0000 0000 0000 0000 0000 0000 0000 0000    ................
TRK 80  0000 0000 0000 0000 0000 0000 0000 0000    ................
17  90  0000 0000 0000 0000 0000 0000 0000 0000    ................
11H A0  0000 0000 0000 0000 0000 0000 0000 0000    ................
    B0  0000 0000 0000 0000 0000 0000 0000 0000    ................
TRS C0  0000 0000 0000 0000 0000 0000 0000 0000    ................
3   D0  0000 0000 0000 0000 0000 0000 0000 0000    ................
3H  E0  1020 0000 0045 4454 4153 4D20 2043 4D44    .....EDTASM..CMD
    F0  9642 9642 2300 0506 FFFF FFFF FFFF FFFF    .B.B#...........
```

目录文件的相对扇区 3

```
DRV 00  5F20 0000 0053 5953 3020 2020 2053 5953    _....SYS0.. .SYS
0   10  5678 1234 0F00 0022 FFFF FFFF FFFF FFFF    VX.4...".. ....
0H  20  5F20 0000 0053 5953 3820 2020 2053 5953    _....SYS8....SYS
    30  5678 1234 0500 1300 FFFF FFFF FFFF FFFF    VX.4............
DRS 40  5F20 0000 0053 5953 3136 2020 2053 5953    _....SYS16.. SYS
174 50  5678 1234 0500 0B00 FFFF FFFF FFFF FFFF    VX.4............
AEH 60  1020 0032 0043 4841 494E 5453 544A 434C    ...2.CHAINTSTJCL
    70  9642 9642 0200 0820 FFFF FFFF FFFF FFFF    .B.B............
TRK 80  1020 0000 004C 4D4F 4646 5345 5443 4D44    .....LMOFFSETCMD
17  90  9642 9642 0A00 0901 FFFF FFFF FFFF FFFF    .B.B............
11H A0  0000 0000 0000 0000 0000 0000 0000 0000    ................
    B0  0000 0000 0000 0000 0000 0000 0000 0000    ................
TRS C0  0000 0000 0000 0000 0000 0000 0000 0000    ................
4   D0  0000 0000 0000 0000 0000 0000 0000 0000    ................
4H  E0  0000 0000 0000 0000 0000 0000 0000 0000    ................
    F0  0000 0000 0000 0000 0000 0000 0000 0000    ................
```

目录文件的相对扇区 4

```
DRV 00  5F20 0000 0053 5953 3120 2020 2053 5953    _....SYS1....SYS
0   10  5678 1234 0500 1000 FFFF FFFF FFFF FFFF    VX.4............
0H  20  5F20 0000 0053 5953 3920 2020 2053 5953    _....SYS9....SYS
    30  5678 1234 0500 1320 FFFF FFFF FFFF FFFF    VX.4............
```

```
DRS 40  5F20 0000 0053 5953 3137 2020 2053 5953   _.....SYS17...SYS
175 50  5678 1234 0500 0A20 FFFF FFFF FFFF FFFF   VX.4............
AFH 60  1020 0000 0041 5350 4F4F 4C20 204D 4153   .....ASPOOL..MAS
    70  9642 9642 0A00 0A00 1900 FFFF FFFF FFFF   .B.B............
TRK 80  1020 003A 0043 4841 494E 424C 4442 4153   ...:.CHAINBLDBAS
17  90  9642 9642 1400 1923 FFFF FFFF FFFF FFFF   .B.B...#........
11H A0  0000 0000 0000 0000 0000 0000 0000 0000   ................
    B0  0000 0000 0000 0000 0000 0000 0000 0000   ................
TRS C0  0000 0000 0000 0000 0000 0000 0000 0000   ................
5   D0  0000 0000 0000 0000 0000 0000 0000 0000   ................
5H  E0  0000 0000 0000 0000 0000 0000 0000 0000   ................
    F0  0000 0000 0000 0000 0000 0000 0000 0000   ................
```

目录文件的相对扇区 5

```
DRV 00  5F20 0000 0053 5953 3220 2020 2053 5953   _.....SYS2....SYS
0   10  5678 1234 0500 1020 FFFF FFFF FFFF FFFF   VX.4............
0H  20  5F20 0000 0053 5953 3130 2020 2053 5953   _.....SYS10...SYS
    30  5678 1234 0500 0F00 FFFF FFFF FFFF FFFF   VX.4............
DRS 40  5F20 0000 0053 5953 3138 2020 2053 5953   _.....SYS18...SYS
176 50  5678 1234 0500 0E00 FFFF FFFF FFFF FFFF   VX.4............
B0H 60  0000 0000 0000 0000 0000 0000 0000 0000   ................
    70  0000 0000 0000 0000 0000 0000 0000 0000   ................
TRK 80  0000 0000 0000 0000 0000 0000 0000 0000   ................
17  90  0000 0000 0000 0000 0000 0000 0000 0000   ................
11H A0  0000 0000 0000 0000 0000 0000 0000 0000   ................
    B0  0000 0000 0000 0000 0000 0000 0000 0000   ................
TRS C0  0000 0000 0000 0000 0000 0000 0000 0000   ................
6   D0  0000 0000 0000 0000 0000 0000 0000 0000   ................
6H  E0  0000 0000 0000 0000 0000 0000 0000 0000   ................
    F0  0000 0000 0000 0000 0000 0000 0000 0000   ................
```

目录文件的相对扇区 6

```
DRV 00  5F20 0000 0053 5953 3320 2020 2053 5953   _.....SYS3....SYS
0   10  5678 1234 0500 1200 FFFF FFFF FFFF FFFF   VX.4............
0H  20  5F20 0000 0053 5953 3131 2020 2053 5953   _.....SYS11...SYS
    30  5678 1234 0500 0B20 FFFF FFFF FFFF FFFF   VX.4............
DRS 40  5F20 0000 0053 5953 3139 2020 2053 5953   _.....SYS19...SYS
177 50  5678 1234 0500 0D20 FFFF FFFF FFFF FFFF   VX.4............
B1H 60  0000 0000 0000 0000 0000 0000 0000 0000   ................
```

```
    70  0000 0000 0000 0000 0000 0000 0000 0000    ................
TRK 80  0000 0000 0000 0000 0000 0000 0000 0000    ................
17  90  0000 0000 0000 0000 0000 0000 0000 0000    ................
11H A0  0000 0000 0000 0000 0000 0000 0000 0000    ................
    B0  0000 0000 0000 0000 0000 0000 0000 0000    ................
TRS C0  0000 0000 0000 0000 0000 0000 0000 0000    ................
7   D0  0000 0000 0000 0000 0000 0000 0000 0000    ................
7H  E0  0000 0000 0000 0000 0000 0000 0000 0000    ................
    F0  0000 0000 0000 0000 0000 0000 0000 0000    ................
```

目录文件的相对扇区 7

```
DRV 00  5F20 0000 0053 5953 3420 2020 2053 5953    _....SYS4....SYS
0   10  5678 1234 0500 1220 FFFF FFFF FFFF FFFF    VX.4............
0H  20  5F20 0000 0053 5953 3132 2020 2053 5953    _....SYS12...SYS
    30  5678 1234 0500 1800 FFFF FFFF FFFF FFFF    VX.4............
DRS 40  5F20 0000 0053 5953 3230 2020 2053 5953    _....SYS20...SYS
178 50  5678 1234 0500 0F20 FFFF FFFF FFFF FFFF    VX.4............
B2H 60  0000 0000 0000 0000 0000 0000 0000 0000    ................
    70  0000 0000 0000 0000 0000 0000 0000 0000    ................
TRK 80  1020 0000 0044 4953 4153 5345 4D43 4D44    .....DISASSEMCMD
17  90  9642 9642 1900 1B24 FFFF FFFF FFFF FFFF    .B.B...$........
11H A0  1020 0000 0044 4952 4348 4543 4B43 4D44    .....DIRCHECKCMD
    B0  9642 9642 0F00 1E02 FFFF FFFF FFFF FFFF    .B.B............
TRS C0  0000 0000 0000 0000 0000 0000 0000 0000    ................
8   D0  0000 0000 0000 0000 0000 0000 0000 0000    ................
8H  E0  0000 0000 0000 0000 0000 0000 0000 0000    ................
    F0  0000 0000 0000 0000 0000 0000 0000 0000    ................
```

目录文件的相对扇区 8

```
DRV 00  5F20 0000 0053 5953 3520 2020 2053 5953    _....SYS5....SYS
0   10  5678 1234 0500 1400 FFFF FFFF FFFF FFFF    VX.4............
0H  20  5F20 0000 0053 5953 3133 2020 2053 5953    _....SYS13...SYS
    30  5678 1234 0500 0E20 FFFF FFFF FFFF FFFF    VX.4............
DRS 40  5F20 0000 0053 5953 3231 2020 2053 5953    _....SYS21...SYS
179 50  5678 1234 0500 1820 FFFF FFFF FFFF FFFF    VX.4............
B3H 60  0000 0000 0000 0000 0000 0000 0000 0000    ................
    70  0000 0000 0000 0000 0000 0000 0000 0000    ................
TRK 80  1020 0000 0053 5550 4552 5A41 5043 4D44    .....SUPERZAPCMD
17  90  9642 9642 1E00 1F25 FFFF FFFF FFFF FFFF    .B.B...%........
```

```
11H A0  0000 0000 0000 0000 0000 0000 0000 0000   ................
    B0  0000 0000 0000 0000 0000 0000 0000 0000   ................
TRS C0  0000 0000 0000 0000 0000 0000 0000 0000   ................
9   D0  0000 0000 0000 0000 0000 0000 0000 0000   ................
9H  E0  0000 0000 0000 0000 0000 0000 0000 0000   ................
    F0  0000 0000 0000 0000 0000 0000 0000 0000   ................
```

目录文件的相对扇区 9

在 NEWDOS/80 中，HIT 扇区的第 32 字节的使用不同于所有其它的 HIT 扇区字节。这个字节包含有分配给目录的，额外附加的 FDE 扇区数，合法的数值是 0，5，15 和 20。当磁盘格式化时就设定了这个值。上例中的 HIT 扇区的第 32 字节都是 00H，这是一个合法的值。

在老的 TRS-80 I 型磁盘中，DIR/SYS (HIT 扇区的第二个字节) 的 HIT 扇区字节的值是 2CH，这是一个错误的数值（正确值应是 C4H)。当存取这个目录文件本身时，这个不正确的值就会使荧光屏显示出 FILE NOT IN DIRECTORY（目录中没有这个文件）错误信息。对于这种盘片，可以利用 SUPERZAP 程序把正确的值 C4H 放入 HIT扇区的第二字节（前面给出的实例中的 HIT 扇区第二字节已改正）。

3. FDE（文件目录项）扇区

目录扇区的剩余部分是 FDE 扇区，每个由 256 字节构成的扇区包含有八个 32 字节组成的 FDE。如果 FDE 的第一字节的位 4 等于 0，则这个 FDE 是空闲的；如果该位等于 1，说明这个 FDE 在使用；如果第一字节的位 7 等于 0，表示使用中的 FDE 是一个 FPDE，如果该位等于 1，这个 FDE 就是 FXDE。当空出一个 FDE 时，只有该 FDE 第一个字节的位 4 要变成 0，同时相应的 HIT 扇区字节也要被置成 0，其它位均不改变。但是，用户可以使用 DIRCHECK 的 C 功能使得每个不使用的 FDE 的全部 32 个字节统统置成 0，这样就得到了一个已经清零的目录。

5.7 FPDE　文件主目录登记项

当建立每个文件的时候，就要为文件在 FDE 扇区内分配一个目录项。其中放有程序的名字，属性，通行字，程序的长度，EOF 以及程序在磁盘上的位置等信息。这个目录项包括 32 个字节，如果 32 个字节存放不下 DOS 需要的关于该文件的信息，就要为 FPDE 建立一个 32 字节的扩充项，称为 FXDE，有关 FXDE 的内容下节讨论，这里只讨论 FPDE。FPDE 内各字节的详细内容如下：

第 1 字节

位7＝0　表示是 FPDE，反之为 FXDE。这一位表示了目录项的属性。

位6＝1　表示是一个系统文件。这一位表示文件的属性。

位5＝0　没有用。

位4＝1　表示这个 FDE 已分配给某一文件了。这一位是分配标志位。

位3＝1　表示文件具有隐性属性，不在荧光屏上显示。

位2—0　表示存取级别。

位2	位1	位0	存取级别
1	1	1	不能存取
1	1	0	只能执行
1	0	1	可读入/执行
0	1	1	（没有使用）
0	1	0	改名/写/读/执行
0	0	1	删除/改名/写/读/执行
0	0	0	存取不受限制

第2字节

位7＝0　必要时可以为文件分配更多的空间。

位7＝1　禁止为文件分配更多的空间。DIR, ATTRIB, CREATE 和 DOS 文件空间分配子程要用到这一位。

位6＝0　DOS 的文件关闭功能可以重新分配超过 EOF 规定的 gr.（即明显不能由文件使用的部分）。

位6＝1　禁止上述功能。DIR, ATTRIB, CREATE 和 DOS 关闭文件时要用到这一位。

位5＝1　自上一次把这一位置0以来，至少已写了文件的一个扇区（或是写新的数据，或是写修改的数据），DIR, ATTRIB, CREATE,PROT, COPY 和 DOS 扇区写子程序要用到这一位。

位4—0　现在没有用。

第3字节　没有用，留待以后使用。

第4字节　这是文件 EOF 的低位字节。这个字节的值表示文件结束的那个扇区中 EOF 的位置。请看后面 FCB 的第14字节的说明。

第5字节　以字节计算的逻辑记录的长度(0表示256)。当用 4420H 向量调用建立一个文件时，从寄存器B中取来的值就放在这里。当打开一个已有文件时（即使打开一个新的输出文件），也不修改这个值。在 NEWDOS/80 中，从来不用这个值。打开文件时，保存在 FCB+9 处的值是来自于寄存器 B，而不是 FPDE。

第6—13字节　文件名，如果文件名不足8个字符就要在右边填满空格（即 20H)。(/) 不存入盘内。

第14—16字节　文件名扩展符，如果文件名扩展符不足3个字符就要在右边填满空格(20H)。

第17—18字节　修改通行字的编码。这是2字节的散列代码。

第19—20字节　存取通行字的编码。这也是2字节的散列代码。

第21字节　EOF 字段的中间字节。

第22字节　EOF 字段的高位字节。第4，第21和第22字节是3字节的 EOF 值。这个 EOF 值不是像 FCB 的 EOF 和 NEXT 字段所用的 RBA 格式（关于RBA的含义请看附录 D），而是保持老的 TRSDOS 的格式，这个格式符合如下规则：

* 如果 EOF 的低位字节等于0，则 EOF 要采用 RBA 格式。
* 如果 EOF 的低位字节不等于0，则 FPDE 中的 EOF 值等于实际的 RBA值

加上 256（即 EOF 值的高位 2 个字节加 1）。

NEWDOS/80 目录的 FPDE 的 EOF 字段符合上述规则，其目的就是要与老的 TRS-80 I 型的 TRSDOS 2.3 版本盘片（请看附录 B.2）保持兼容。只有在文件建立过程中写 EOF 值时和 DOS 关闭文件时才把文件的新的 EOF 值放进 FPDE 中。这样，如果工作中因某种原因导致系统失败（这时并未关闭打开的文件）而需要复位以便重新启动系统，则用户能够期望获得如下结果：在系统失败时为了输出而打开的文件，将会包含有新的数据，但是通常 EOF 是老的。

关于 EOF 和 NEXT 与其它 DOS 不兼容的情况请看附录 B.2 节。

通过上面介绍的内容，我们可以找到文件的名字，文件的长度等信息，但是文件在磁盘上的精确位置还是不知道，下面介绍的范围元就记录了关于文件精确位置的信息。

第23—30字节　这是四个 2 字节对（每个字节对就是一个范围元），每一个范围元确定了分配给这个文件的一个连续磁盘区。范围元的格式是：

第一字节——根据其内容的不同，有下列三种情况：

* 255（或 FFH）表示该文件范围元的结束。
* 254（或 FEH）表示下一个字节含有分配给该文件的 EXDE 的 DEC码。
* 0—253（或 0—FDH）取的值等于文件区开始的磁盘 gr. 组编号。考虑到 GAT 扇区能够管理的 gr. 组数是 96（扩充后为 192），因而限定该值在 0—191 的范围之内。这个值表示在 GAT 扇区中与此 gr. 组关联的字节的相对位置。第二字节（当第一字节之值小于 254 时）：
* 左边 3 位等于从 gr. 组的开头到文件区开始的 gr. 相对偏移量（0—7）。
* 右边 5 位的值加 1 以后得到的数等于分配给文件区的连续 gr. 数。

举例 1　假定 1gr. 组＝2gr.，而且范围元内容是0718，请指出该范围元表示的磁盘位置。

第一字节为 07H，表示文件开始于第 7gr. 组（用的是 16 进制表示格式）。

第二字节为18H，这个字节采用位记录形式。分解如下：

000 11000 (＝18H)

└———18H 表示该文件区有 25 个连续的 gr.（即18H＋1＝19H)。

└———从 gr. 组的开头到文件开始的相对偏移量。0表示 gr. 偏移量为 0，即从 0 扇区开始。

举例 2　假定 1gr. 组＝2gr.，范围元内容是 1A29。

第一字节为 1AH 表示文件开始于第 26gr. 组。

第二字节为 29H

001 01001 (＝29H)

└———09H 表示该文件区有 10 个连续的 gr.。

└———从该 gr. 组的开头到文件区开始的相对偏移量为 1 个 gr.。因为我们假定 1gr.组＝2gr.，即从第 5 扇区开始。

从上面的两个例子中可以看到：

（1）通过范围元很容易确定该文件处于磁盘上的哪一个 gr. 组，如例 1 中的第7gr. 组，

例 2 中的第 26gr. 组。

（2）当 1gr.组＝2gr. 时（通常情况），如果范围元 1 的第 2 个字节是19H或小于19H，则文件起始于 0 扇区。如果范围元 1 的第 2 个字节是 20H 或大于 20H，则文件起始于第 5 扇区。

第31—32字节　这个范围元的第一个字节不是 255 就一定是 254。如果第一字节是 255，表示范围元结束。如果是 254，则后面的一个字节就是 FXDE 的 DEC 码。根据这个 DEC 码可以找到 FXDE。

举例 3　下面是一个 32 字节组成的 FPDE，它的相对扇区编号为9。并假定 1gr.组＝2gr.

相对字节
（16进制）

C0	1000	002B	0044	4953	4B4F	5247	2050	434C
D0	9642	9642	4200	2023	0124	0500	0701	FE40

2023 — 范围元 1　0124 — 范围元 2　0500 — 范围元 3　0701 — 范围元 4

FE40 — 范围元 5

第一字节 FEH　表示下一个字节是定位 FXDE 的 DEC 码。

第二字节40H　FXDE 的 DEC 码（DEC 码定义参看附录D）。

010 00000（＝40H）

00000 ——0＋2＝2，这是 FXDE 的目录相对扇区编号。本例为 HIT 扇区后面的一个扇区。

010 ——2×32＝64，这是目录项在扇区中的相对字节编号。本例为相对字节 40H（即64）。

所以这个 FXDE 是相对扇区2，相对字节 40H 开始的目录项。

5.8 FXDE 文件扩展目录项

当分配给一个文件的连续区域超过 4 个范围元的时候，就要为该文件分配附加的范围元，这些附加的范围元放在 FXDE 中。FXDE 的格式如下：

第 1 字节　位 7 和位 4 两者都是 1，表示是 FXDE，其余各位均为0。

第 2 字节　该文件前一个 FXDE 或 FPDE 的 DEC 码。这是一个反向链指针（请看例4）。前一个目录项的第31字节一定是254，而第 32 字节应该包含有 FXDE 的 DEC 代码。

第 3—22 字节　没有使用，应该都是0。

第23—32字节　与 FPDE 中规定的含义是相同的。

举例 4　这里列出了例 3 中用到的 FXDE。

这是相对扇区 2

相对字节
（16进制）

00	5E00	0000	0042	4F4F	5420	2020	2053	5953
10	EB29	210E	0500	0000	FFFF	FFFF	FFFF	FFFF
20	0000	0000	0000	0000	0000	0000	0000	0000

30	0000	0000	0000	0000	0000	0000	0000	0000
40	90C7	0000	0000	0000	0000	0000	0000	0000
50	0000	0000	0000	0821	FFFF	FFFF	FFFF	FFFF

例3中的范围元5给出 FXDE 在相对扇区2中的相对字节 40H 开始。我们可以从上面列出的相对扇区2中的相对字节 40H 处找出 FXDE 的前二个字节为 90C7。本节开始处已说明 FXDE 的第1字节各位的含义与 FPDE 第1字节中各位的含义完全一样，所以：

1 0 0 1 0 0 0 0 (=90H)

位4=1，表示这个目录项已经分配给某一文件了。

位7=1，表示是 FXDE，不是 FPDE。

FXDE 的第2字节是一个反向链指针，它是指向勾链到 FXDE 的原来的 FPDE（或 FXDE）的 DEC 码。

1 1 0 0 0 1 1 1 (=C7H)

7，7+2=9 表示反向链接到相对扇区9。

6，6×32=196（C0H）表示目录项在扇区中的相对字节号。

所以，本例中的 FXDE 在相对扇区2，反向链接到相对扇区9。根据算出的目录项在扇区中的相对字节编号为 C0H，可以得到链接本例 FXDE 的前一个 FPDE（或 FXDE）是从相对扇区9的相对字节 C0H 开始。读者只要看一下例3就可以得到证实。

FXDE 中从第3—22字节均为0，第23—32字节为范围元。本例中仅用了第1个范围元（0821）。其中第1字节为 08H，说明文件扩充到第 8gr. 组，第2字节为 21H，它是位定义的，分解如下：

0 0 1 0 0 0 0 1 (=21H)

1，1+1=2 表示有2个连续的 gr. 区。

1，表示从第 8gr. 组开始到文件开始的相对扇区偏移量为1个gr.，即从第5扇区开始。

第2范围元为 FFFF，表示文件已经结束。

5.9 FCB 文件控制块

FCB 也称为 DCB（数据控制块）或 IOB（输入/输出块）。它是文件管理的依据，FCB 为查找文件提供了必要的信息。

为了从盘上读取文件信息（或者把文件信息写到盘上），就必须在文件和用户程序之间建立联系。DOS 打开文件的功能（见 3.15 节）就可以建立这种联系，而 DOS 关闭文件的功能（见 3.15 节）可以解除这种联系。存在这种联系的时候，有关连接的控制信息保留在主存中的32个字节组成的一个区内，这个区称为文件的控制块 (FCB)。文件打开的时候，用户要规定这个 FCB 放在用户存储器的哪一个区域中。当存在这种联系的时候，内存的 FCB 区不能再作其它的用途。DOS 不去记住 FCB 在什么地方。用户要把执行与 FCB 有关的功能所需要的 FCB 告诉 DOS。要解除这种联接可以简单地不再在功能调用中使用这

个 FCB，或者在打开一个新的文件时使用这个 FCB 区。可是要记住：写一个文件时，其中的 EOF 是要改变的，为确保把 EOF 正确地放置在FPDE中，必须执行 DOS 的 CLOSE 命令，关闭这个文件或者经 DOS 去写 EOF（见 3.28 节）。

在打开一个文件时（DOS 调用 4420H 或 4424H），调用的程序要在寄存器 DE 中放入 FCB 的 32 个字节存储区的首地址，以便系统在打开文件时使用 FCB。在把文件打开以前，用户必须把所要文件的文件标识符（用 0DH 或 03H 结尾）放在 FCB 的开始几个字节内（是按左齐形式放置的），这样做了以后，DOS 关闭文件时，就将会把它放回去。在文件打开以后，FCB 的各个字节的每一位的内容和含义就不再是所需处理文件的文件标识符了，详细情况请看后面的说明。NEWDOS/80 可以接受 TRS-80 Ⅲ型 TRSDOS 的 50 字节 FCB 区中的前 32 个字节。在 FCB 打开时，FCB 的 32 个字节的格式如下：

第 1 字节

位7＝1　文件和用户程序之间处于连接状态（即已完成打开操作）。这一位是文件打开标志位。

位7＝0　不存在连接（或者文件尚未打开，或者文件刚刚关闭）。

位6—位2＝0　不用。

位1＝1　在磁盘上（而不是文件中）FCB 的 NEXT 字段的值和 EOF 字段的值采用 RBA 格式。这就使用户能够直接对磁盘扇区输入/输出，完全不顾及文件的概念。在用 0013H 或 001BH 调用作字节的输入/输出时，这一位决不应该是 1。

位0＝1　用 DOS 写目录扇区同样的方法，为写到文件内的扇区写读保护。在用0013H 或 001BH 调用作字节的输入/输出时，这一位决不应该是 1。

第 2 字节

位7＝1　表示正在经 FCB 作单字节操作或逻辑记录操作（记录长度在 FCB 的第 10 个字节）。NEXT 值保持指向要读或写的下一个字节。在文件打开时如果寄存器 B 不为 0，这一位就要置成 1。每当经 0013H 或 001BH ROM 调用作字节输入/输出时，这一位也要置成 1。这一位表示操作类型。

位7＝0　读和写操作是以整个256字节的扇区来进行的，每完成一次输入/输出操作，FCB 的 NEXT 值就要递增 256 字节。

位6＝0　FCB 的 EOF 值要设定为每次完成写操作时 FCB 得到的 NEXT 值。

位6＝1　只有在完成这次写操作得到的 NEXT 值超过当前的 EOF 值，才把 FCB的 EOF 值设定为 FCB 得到的 NEXT 值。

位5＝0　FCB 的缓冲区包含有当前文件扇区的数据。如果位 5＝1，缓冲区就不包含有当前文件扇区的数据，需要时，应该把扇区数据读进这个缓冲区。

位4＝0　FCB 的缓冲区不包含有尚未发送到文件的更新过的数据。如果位 4＝1，缓冲区内放有尚未发送到文件的更新过的数据。在 DOS 关闭文件期间，如果这位是 1，缓冲区中的数据就会自动写到盘上。更新过的数据还能在每次 443FH 和 4451H 调用时写到盘上，并且每次调用 4442H, 4445H, 4448H 和 444EH 把文件定位于不同的扇区内时，更新过的数据也能写到盘上。

位3＝1　表示 FCB 是按 NEWDOS/80 2.0 版本的要求安排第 17—32 字节的。这一

位在 DOS 打开文件时置成 1。如果位 3=0，表示 FCB 是老的格式，在 NEWDOS/80 2.0 版本中，位 3=0 是非法的。

位2—位0　表示文件的存取级别（请看 2.3 节 ATTRIB 命令的 PROT 参数）。

第 3 字节

位7—位5　这几位的含义与 FPDE 第 2 字节中的相应位相同（见 5.7 节）。如果位5=0，DOS 扇区写子程序在把当前扇区写到盘上去以前，要把 FCB 和 FPDE 中的这一位置成 1。

位4—位0　现在没有用，留待以后使用。

第 4，5 字节　这两个字节放有 FCB 缓冲区的内存储器地址。用户决定缓冲区设在什么地方，在调用 DOS 打开文件子程序之前，要把这个地址放在 HL 寄存器中。扇区从磁盘上读入这个缓冲区，或者从这个缓冲区把扇区写到磁盘上。

第 6 字节　FCB 的 NEXT 字段的低位字节。这是扇区中的相对位置。请看后面 FCB 第12字节的讨论。

第 7 字节　放有该文件的盘片所在的相对驱动器编号。

第 8 字节　文件 FPDE 的 DEC 码。打开 FCB 以后，这个 DEC 码是打开的 FCB 和文件目录信息之间的联系纽带，因为 FCB 本身不再包含有文件标识符。

第 9 字节　EOF 的低位字节。这是在文件结束扇区中的相对位置。请看后面的 FCB 的第14字节的讨论。

第10字节　这个字节放的是文件逻辑记录的长度 (LRECL)，0 表示 256。文件打开时由调用的程序在寄存器 B 中给出这个值。如果在打开文件时这个字节不是 0，就把 FCB 的第 2 字节的位 7 置成 1，接着的 DOS 扇区读或写操作的调用程序必须包含（在寄存器 HL 中）要移入 FCB 缓冲区（写操作）的逻辑记录的地址，或者包含把 FCB 缓冲区（读操作）的逻辑记录填充进去的地址（也在寄存器 HL 中）。

第11字节　NEXT 字段的中间字节。

第12字节　NEXT 字段的高位字节。第 12，11 和 6 字节构成了该文件中的一个 3 字节的相对字节地址 (RBA)，它指出了要处理（输入或输出）的下一个字节的位置。

对于单字节和逻辑记录 I/O，DOS 严格按照 RBA 格式保持 FCB 的 NEXT 字段。

对于整扇区的 I/O，DOS 也严格按照 RBA 的格式保持着 NEXT 字段。但是 DOS 的工作极其微妙，有点捉摸不定，如果用户对它们的工作了解不够，就很容易出问题。在整扇区 I/O 期间，DOS 不改变 NEXT 字段的低位字节。通常，这个字节是 0，而且这样最好。然而，用户能够把这个字节置成非 0 的值，如果前次按单字节或逻辑记录方式进行 I/O 操作，则低位字节或许将会是非 0 值。用户必须知道下面的一些规则：

* 在整扇区读操作期间，就像单字节和逻辑记录读操作一样，NEXT 字段的所有三个字节都参与 EOF 检查。
* 在整扇区写操作期间，当 NEXT 字段的低位字节不为 0 时，完成写操作并不使NEXT 字段递增 256 字节，而且 EOF（如果要更新它）采取不递增的 NEXT 值。理由是：如果 NEXT 值的低位字节是 0，写操作完成以后的 NEXT 值就成为下一扇区的第一字节，而 NEXT 字段的低位字节不为 0，则写操作完成以后的 NEXT 值还留在刚写过的扇区内。

NEXT 和 EOF 字段与其它操作系统的不兼容性在附录 B.2 中讨论。

第13字节　EOF 字段的中间字节。

第14字节　第14，13 和 9 字节构成了文件中的一个 3 字节的 RBA，它指出了文件的结束（也就是文件最后一个数据字节后面紧跟着的一个字节）。打开文件时，从 FPDE 中取得这个预置值，而在扇区、逻辑记录或字节写操作时在 FCB 第 2 字节位 6 的控制下修改这个值。

第15—22字节　与 FPDE 的第 23—30 字节的含义相同。

第23—24字节　如果 FCB 的第 25—32 字节是当前 FXDE 的 4 个范围元，则这两个字节的数表示了 FXDE 的第 1 范围元的第 1gr. 的相对 gr. 号。如果这两个字节的值等于 FFFFH，则表示第25—32字节不是 FXDE 的范围元。

第25—32字节　与当前 EXDE（如果有的话）的第23—30 字节的内容相同。

FCB 的第 17—32 字节的讨论：

NEWDOS/80 2.0 版本已经改变了 FCB 的第 17—32 字节的定义，所以现在的定义与 NEWDOS/80 1.0 版本和 TRS-80 I 型的 TRSDOS 中的定义是不同的。一般来说，极少数用户程序才去访问这些字节，因为这些字节只是用于减少常驻 DOS 存取目录的次数。然而，有些用户程序（像 BASIC 中原有的 SUPERZAP程序）已经使用了这些字节的老定义，以避免磁盘（而不是文件）I/O 时必须去打开一个文件。NEWDOS/80 的 1.0 和 2.0 版本已经提供了一种磁盘（不同于文件）I/O 方法（请看 FCB 第 1 字节的位 1 定义），所以你要使用 NEWDOS/80 2.0 版本现在提供的方法就必须放弃以前那种老的 FCB 定义方法。为了适应这种变动，在 NEWDOS/80 2.0 版本中已经启用了原来保留备用的 FCB 第 2 字节的位 3，以便区分新、老 FCB 格式。防止发生不可挽救的事故。

改变 FCB 的第 17—32 字节的定义以后，使得 FCB 可以容纳任何有 8 个及 8 个以下范围元的文件范围信息（带有 A 选用参数的 DIR 命令将显示一个文件有多少范围元）。如果文件占有邻接的磁盘空间，则 8 个范围元足以存放大约 300000 个字节的数据，如果目录占用空间，则约为 270000 个字节。

如果文件具有的范围元超过 8 个（这就是说，要为文件分配多于一个的 FXDE），则 FCB 内包含有文件最初的 4 个范围元和最后一个有扇区读或写的 FXDE 的 1 到 4 个范围元的空间信息。对于大的随机存取文件，很可能比 NEWDOS/80 1.0 版本要求更多的目录存取操作。

5.10 数据的恢复

当我们对 NEWDOS/80 2.0 的磁盘结构，数据分布及磁盘目录有一定的了解以后，就有可能把某些因目录被破坏而影响使用的有用程序给予复原。另外，对于不小心删除的有用文件，立即就可以给予恢复，避免了不必要的损失。这一节要介绍如何恢复被删除的文件，修补有问题的 BASIC 程序和汇编语言程序。在这一节中还要用到两个很有用的独立于 NEWDOS/80 的附加程序，它们是检查和修改磁盘或存储器内容的 SUPERZAP 程序，以及检查目录错误的 DIRCHECK 程序，有关这两个程序的详细内容请看第六章的有关章节。

1. 恢复用 KILL 命令删除的文件

当我们用 KILL 命令从磁盘上删去一个文件的时候，用户就不能够再存取这个文件了。

在删除一个文件时会发生下列三件事：

（1）从目录的 HIT 扇区中勾消该文件的散列代码。

（2）修正 GAT 扇区，使得它能反映当前 gr. 分配情况。

（3）把该文件的 FPDE（或 FXDE）中的第 1 字节的位 4 置成 0。

但是，被删除文件的内容还留在磁盘上，而且，FPDE（或 FXDE）除了第 1 字节的位 4 被修改成0以外，其它字节均保持不变。只有在 DOS 要重新使用这个 FPDE（或 FXDE）及磁盘空间时才会真正破坏掉被删除文件的内容。这就使得有可能恢复被删除的文件。

要想恢复被删除的文件，要经过下列几个步骤：

（1）先要计算被删除文件的文件名和扩展符的1字节散列代码。前面已经介绍了有关文件名散列代码的概念及其计算方法，可以用第 88 页介绍的公式来计算文件名的散列代码，也可以用 SUPERZAP 程序的 DNTH 功能来获得文件名的散列代码，后者只要按照 SUPERZAP 程序的要求逐项回答询问，即可得到文件名的散列代码，极其方便。

（2）得到被删除文件的散列代码以后，就要设法把它放到 HIT 扇区内的一个合适的位置上，这个位置就是用 KILL 命令删除的文件名散列代码原来存放的位置。为此，首先要从磁盘上找到 HIT 扇区的位置，然后再找出被删除文件的文件名散列代码在 HIT 扇区中所处的位置，最后用 SUPERZAP 程序的修改工作方式的 MOD 功能去修改散列代码。

（3）根据该文件 FPDE（或 FXDE）的范围元内容，修改 GAT 扇区，恢复被删除文件的 gr. 分配情况。

（4）最后一项修改就是用 SUPERZAP 程序把 FPDE（或 FXDE）的第 1 字节的位 4 置成 1。

完成上面的这些修改以后，被删除的文件就得到了恢复，这时用 DIR 命令就可以看到该文件的名字又出现在荧光屏上，于是可对该文件进行正常的存取操作。

举例：假设刚才不小心删除了 0 号驱动器内盘片上的 MARJONG/BAS 程序，用系统命令 DIR 已经无法在目录上查到这个文件的文件名。请设法恢复这个文件。

根据上面介绍的复原步骤，首先要找出 MARJONG/BAS 的散列代码。计算散列代码最简便的方法是使用 SUPERZAP 程序的 DNTH 功能。在系统命令级（即处于 NEWDOS/80 READY 状态）打入 SUPERZAP ENTER。

这时荧光屏上会显示出 SUPERZAP 程序的功能清单（请看第六章的 6.1 节）。接着打入 DNTH，于是 SUPERZAP 程序要询问文件名及文件名扩展符，这时按照 SUPERZAP 程序的要求分别打入 MARJONG 和 BAS，于是荧光屏上会显示出“EE”，这就是 MARJONG/BAS 的散列代码。

算出 MARJONG/BAS 的散列代码以后，就要想法把它放入 HIT 扇区内它原来所处的位置。放在我们面前的问题是：HIT 扇区在什么地方？MARJONG/BAS 的散列代码又该放在 HIT 扇区的哪一个位置？要解决这两个问题，首先要知道目录文件放在磁盘的哪一个gr. 组。根据 5.6 节的说明，可以知道：

（1）每个盘片的第一个扇区的第 3 个字节总是存放目录起始 gr. 组的编号。

（2）磁盘目录文件总是起始于 gr. 组的分界处。

（3）目录文件最前面放有 GAT 扇区，接着是 HIT 扇区，再后面是文件目录项扇区。

查找盘片的第一个扇区第 3 字节内容的方法是使用 SUPERZAP 程序的 DD 功能，显

示 0 号驱动器的相对扇区 0 的内容，即可看到第 3 字节的内容为 11H （说明目录起始于第 17gr. 组或相对扇区 170)。HIT 扇区是目录文件中的第 2 个扇区，即相对扇区 171。知道了 HIT 扇区在磁盘上的相对扇区编号以后，只要使用 SUPERZAP 程序就可以很容易找到HIT 扇区，并显示其内容。本例中的 HIT 扇区内容如下：

```
DRV  00   A2C4 2E2F 2C2D 2A2B 0000 0000 0000 0000    .../,-*+........
0    10   0000 0000 0000 0000 0000 0000 0000 0000    ................
0H   20   2829 2627 27A7 26A6 0000 0000 0000 0000    ()&'.&..........
     30   0000 0000 0000 0000 0000 0000 0000 0000    ................
DRS  40   25A5 24A4 23A3 24A4 0000 0000 0000 0000    %.$.#.$.........
171  50   0000 0000 0000 0000 0000 0000 0000 0000    ................
ABH  60   F000 2AD3 0000 2E00 0000 0000 0000 0000    ..*.............
     70   0000 0000 0000 0000 0000 0000 0000 0000    ................
TRK  80   6879 323B 0000 461F 0000 0000 0000 0000    hy2;..F.........
17   90   0000 0000 0000 0000 0000 0000 0000 0000    ................
11H  A0   7029 0200 0000 EE27 0000 0000 0000 0000    p).....'........
     B0   0000 0000 0000 0000 0000 0000 0000 0000    ................
TRS  C0   F0A1 6A00 0000 764F 0000 0000 0000 0000    ..j...vO........
1    D0   0000 0000 0000 0000 0000 0000 0000 0000    ................
1H   E0   0089 0000 0000 4600 0000 0000 0000 0000    ......F.........
     F0   0000 0000 0000 0000 0000 0000 0000 0000    ................
```

接着要确定 MARJONG/BAS 散列代码在 HIT 扇区中的具体位置。根据本节前面的内容可以知道，当删去 MARJONG/BAS 程序时，已从 HIT 扇区中勾消了它的散列代码，因而无法从这张 HIT 扇区分布图中找出它来。但是可以通过其它途径来查找。我们知道当删除一个文件时，表示该文件的 FPDE 中，只有第 1 字节的位 4 被置成 0，其它各字节均未改变。因为由 32 个字节构成的 FPDE 中第6—16 字节就放有文件名及文件名扩展符，所以我们可以从相对扇区 172 开始的 FDE 中去寻找 MARJONG/BAS 的 FPDE。使用 SUPER-ZAP 程序的 DD 功能，可以找到在相对扇区编号 179 的扇区内的相对字节 60H 起的32个字节就是MARJONG的 FPDE。从下面给出的 179 扇区内容分布图中可以看到右半部的 ASCII 码区就清楚的显示了 MARJONG/BAS 文件名（标有下划线的部分)。

```
DRV  00   5F20 0000 0053 5953 3520 2020 2053 5953    _.....SYS5....SYS
0    10   5678 1234 0500 1400 FFFF FFFF FFFF FFFF    VX.4............
0H   20   5F20 0000 0053 5953 3133 2020 2053 5953    _.....SYS13...SYS
     30   5678 1234 0500 0E20 FFFF FFFF FFFF FFFF    VX.4............
DRS  40   5F20 0000 0053 5953 3231 2020 2053 5953    _.....SYS21...SYS
179  50   5678 1234 0500 1820 FFFF FFFF FFFF FFFF    VX.4............
B3H  60   0020 0066 004D 4152 4A4F 4E47 2042 4153    ...F.MARJONG.BAS
     70   9642 9642 2700 3807 FFFF FFFF FFFF FFFF    .B.B'.8.........
TRK  80   1020 0000 0053 5550 4552 5A41 5043 4D44    .....SUPERZAPCMD
17   90   9642 9642 1E00 1F25 FFFF FFFF FFFF FFFF    .B.B...%........
11H  A0   1020 00B7 0046 4958 5345 4320 2043 4D44    .....FIXSEC..CMD
     B0   9642 9642 0200 3C00 FFFF FFFF FFFF FFFF    .B.B..<.........
TRS  C0   1020 0090 0048 414E 3330 2020 2042 4153    .....HAN30...BAS
9    D0   9642 9642 2800 3D07 FFFF FFFF FFFF FFFF    .B.B(.=.........
9H   E0   0000 0000 0000 0000 0000 0000 0000 0000    ................
     F0   0000 0000 0000 0000 0000 0000 0000 0000    ................
```

所以文件 MARJONG/BAS 的 FPDE 在 0 号驱动器盘片的第 17gr. 组中的相对扇区 9

内，它位于相对字节 60H 处。众所周知，一个文件的 FPDE 的位置是可以从 HIT 扇区中该文件的散列代码所处的位置（DEC 码）得到（请看 5.6 节的1.）。反之，知道了一个文件的 FPDE 的位置，也可以算出 DEC 码，从而得到该文件的散列代码在 HIT 扇区中的位置，这样就可以完成恢复 MARJONG/BAS 的散列代码的工作。现在 MARJONG/BAS 的FPDE 在第 17gr. 组中的相对扇区 9，相对字节 60H (96) 处，由此可求得 DEC 码为：

9－2＝7 （DEC 码的位4—位 0）

96÷32＝3 （DEC 码的位7—位 5）

所以 DEC 码是 01100111 即 67H。

我们可以用 SUPERZAP 程序的修改功能 MOD 把“EE”代码放入 HIT 扇区的相对字节位置 67H 处（即前面 HIT 扇区图中有下划线处）。

下一步的工作是要修改 GAT 扇区，因为删除一个文件后，操作系统要修改 GAT 扇区中表示该文件分配情况的位，使得 GAT 扇区能反映当前磁盘 gr. 的分配情况。

要找出表示 MARJONG/BAS 程序在 GAT 扇区中所对应的位，需要依靠这个文件的 FPDE 中范围元所表征的信息，从相对扇区 179 中可以看到 MARJONG/BAS 的 FPDE 中范围元 1 （即 FPDE 的第 23—24 字节）包含有“3807”，其中“38”表示该文件从 56（即 38H）gr. 组开始，“07”表示从该 gr. 组的 0 扇区开始，连续占有 8 个 gr.（可参考 5.7 节例 1 和例 2 的计算方法）。根据 5.6 节中1. 的 GAT 扇区排列图可在本例的 GAT 扇区中找出 MARJONG/BAS 的位置，它应从 56gr. 组开始，占有连续的 8 个 gr.，下面列出的就是删除 MARJONG/BAS 以后的 GAT 扇区内容。下划线表示的就是相应的 gr. 分配情况。

```
DRV 00  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF   ................
0   10  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF   ................
OH  20  FFFF FFFF FFFF FFFF FFFF FCFF FFFF FFFF   ................
    30  FFFF FFFC FCFC FCFC FCFC FCFC FFFF FFFF   ................
DRS 40  FFFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC   ................
170 50  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF   ................
AAH 60  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC   ................
    70  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC   ................
TRK 80  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC   ................
17  90  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC   ................
11H A0  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC   ................
    B0  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF   ................
TRS C0  FFFF FFFF FFFF FFFF FFFF FF82 0000 E042   ..............B
0   D0  4E45 5744 4F53 3830 3030 2F30 302F 3030   NEWDOS8000/00/00
OH  E0  0D41 5349 430D 4540 4956 4552 592C 312C   ,ASIC.ELIVERY,1,
    F0  320D 0DFF FFFF FFFF FFFF FFFF FFFF FFFF   2...............
```

因为 MARJONG/BAS 的 FPDE 中第 2 范围元的内容是“FFFF”，说明该程序只占有 8 个 gr.，不再占有其它的磁盘空间。从上面的 GAT 扇区中可以看到 gr. 分配情况，即相对字节位置 38H—3BH 的内容均为“FC”，FC 表示表示第 1 和第 2 gr. 均是空闲的，尚未分配掉（请看 87 gr. 分配表区值的定义），这是删除 MARJONG/BAS 程序时由操作系统修改成这样的，现在我们可以用 SUPERZAP 程序的 MOD 功能，把这些字节均改成“FF”，表示这 8 个 gr. 已经分配掉了，使这些 gr. 今后不会再被分配给其它文件，也就使 MARJONG/BAS 的内容得到了保护。

最后，用 SUPERZAP 程序把 MARJONG/BAS 的 FPDE 的第 1 字节的位 4 置成 1（即相对扇区 179 的相对字节 60H 的内容由“00”改成“10”），表示这个文件目录项已经分配给 MARJONG/BAS 了。

到目前为止，我们已经完成了所有的修补工作。这时使用系统命令 DIR，就可以看到在文件目录中重新出现了 MARJONG/BAS。于是可以像对待其它文件一样的对该文件进行存取和处理。

在这个实例中所用的磁盘是 80 磁道，每个 gr. 组等于 2 个 gr.，每个 gr. 等于 5 个扇区，每个扇区包含 256 个字节。这是 NEWDOS/80 2.0 常用的一种标准格式。

2. 恢复一个 BASIC 程序（二进制或 ASCII 格式）

如果一个 BASIC 程序能用 DIR 命令在目录清单上看到它，但是无法装入和运行，这时很可能是该文件的散列代码被破坏了，为了要运行这个程序，就要用上面介绍的方法进行复原处理。

但是如果是 GAT 扇区有问题，则可以用下面介绍的简单办法进行修补。

（1）先打入命令 BASIC，进入 BASIC 管理状态，然后把怀疑 GAT 分配有问题的BASIC 程序用 LOAD 命令装入计算机存储器。

（2）换一个文件名把该程序用 SAVE 命令存入磁盘，这时得到的新文件有正确的 GAT 分配。带有 GAT 错误的文件并不影响装入和执行，但是在进行 SAVE 操作时，由于不正确的 GAT，就有可能破坏 gr. 分配有问题的 BASIC 程序的内容，引起出错。

（3）最后用 DIRCHECK 程序检查是否还有 GAT 或 HIT 错误存在。如有错误，要继续进行修正，直到所有的错误全部排除为止。

3. 修复一个 ASCII、二进制数据文件或者是一个汇编语言程序

可以采用如下的方法：

（1）首先使用 DIRCHECK 程序检查是否存在 GAT 和（或）HIT 错误。

（2）根据 GAT 扇区及 HIT 扇区分布图，用 SUPERZAP 程序修正这些错误。

（3）重复第（1）步，直到所有的 GAT 和 HIT 错误都得到纠正为止。

关于 DIRCHECK 程序的使用及详细介绍请看 6.4 节，这里只介绍使用 DIRCHECK 程序检查 HIT 和 GAT 错误时可能出现的一些情况。

（1）xx BAD ″HIT″ SECTOR BYTE（有问题的 HIT 扇区字节）

〔说明〕在 HIT 扇区中这一个相对字节位置本来不应该存在散列代码，现在有了一个散列代码。最左边 xx 数字表示了在 HIT 扇区内有问题的散列代码的相对字节位置。这时应该用“00”去替换这个有问题的散列代码。

（2）文件名 xx FPDE HAS BAD CODE IN ″HIT″ SECTOR（在 HIT 扇区内有一个不正确的散列代码）

〔说明〕所列出文件名的文件的 FPDE 相应的散列代码不正确。xx 代表了不正确散列代码在 HIT 扇区内的相对字节位置。因此，要纠正这类错误，必须先用 SUPERZAP 程序计算出文件名的散列代码，然后用获得的这个正确的散列代码去更换由 xx 数字给出的 HIT 扇区内相对字节位置的不正确的散列代码。

（3）bb, x ＊＊＊＊＊ GRANULE FREE, BUT ASSIGNED TO FILE (s)

yy 文件名

（这个 gr. 实际上已经分配给显示出文件名的文件了，但是在 GAT 扇区内，表示出这个gr. 是空闲的）

〔说明〕这种错误使得已经由一个文件在使用的 gr. 有可能再次被分配给另一个文件，造成有用信息的损失。其中 bb 表示了 GAT 扇区排列图（见 5.6 节的 1.）中的 gr. 组编号（也是 GAT 扇区中该 gr. 组的相对字节地址），而 x 表示了该 gr. 组中的相对 gr. 编号，这个 gr. 的分配有问题。为了纠正这种错误，必须把 GAT 扇区中表示这个 gr. 的位，由 0 改成1。第二行中的 yy 文件名，表示了占有这个 gr. 的文件的文件名，yy 是相应散列代码在 HIT 扇区中的位置（即 DEC 码）。

（4）xx BAD "GAT" SECTOR BYTE

（错误的GAT扇区字节）

〔说明〕在用户特定的使用条件下，有些 GAT 扇区字节是不能用的，例如你的 NEWDOS/80 是工作在80磁道盘片的情况下，则 GAT 扇区内相对字节位置在 50H 开始，到 5FH 为止（即相应 81—96 磁道）是不能表示实际的盘片磁道的。如果因某种原因改变了这些 GAT 扇区字节的话（正常情况，这些字节内容均为FF），则将显示上面所列出的信息，其中 xx 表示这个错误的 GAT 扇区字节在 GAT 扇区内的相对字节位置。

（5）bb, x ＊＊＊＊＊ GRANULE LOCKED OUT, BUT FREE

（这是 gr. 存在表区发生了错误，而且对应于 gr. 分配表的位置是空闲的）

〔说明〕这是表示 gr. 存在表区（即 60H—BFH）的错误，当采用 80 磁道的盘片时，GAT 扇区内（60H—AFH）的相对字节内容均为 FC，如果由于某种原因使其值不等于 FC，则会显示这种错误信息。其中 bb，x 表示出错 GAT 扇区字节相对应的 gr. 分配表区（00H—4FH）的 gr. 相对位置（请看 5.6 节的 1.），而且这个 gr. 分配表（注意！不是 gr. 存在表）相对字节所表示的 gr. 没有分配给任何文件，是一个空闲的 gr.。

（6）bb, x ＊＊＊＊＊ GRANULE LOCKED OUT, BUT TO FILE (s)

yy 文件名

（这是 gr. 存在表区发生了错误，而且对应于 gr. 分配表的位置表示的 gr. 已分配给文件了）

〔说明〕这个错误信息的含义与(5)相同，只是 bb，x，所表示的gr.组及gr. 相对编号所指的磁盘扇区已经分配给了显示出文件名的文件，该文件的 FPDE 在目录文件中的 DEC 码是yy。

（7）bb, x ＊＊＊＊＊ GRANULE ALLOCATED, BUT NOT ASSIGNED TO ANY FILE

（在 GAT 扇区中，表示这个 gr. 已经分配掉了，但是实际上没有分配给文件）

〔说明〕这种错误使得有些空闲的 gr. 不能有效地分配给文件使用，因而降低了磁盘的利用率。其中 bb 表示 GAT 扇区排列图中的 gr. 组编号（也是 GAT 扇区中该 gr. 组的相对字节位置），而 x 表示该 gr. 组中的相对 gr. 编号，这个 gr. 的分配有问题。为了纠正这种错误，必须把 GAT 扇区中表示这个 gr. 的位由 1 改成0。

（8）bb, x ＊＊＊＊＊ GRANULE ALLOCATED, BUT ASSIGNED TO MULTIPLE FILES

yy 文件名 1

nn 文件名 2

（这个 gr. 已经分配了，而且同时分配给了多个文件）

〔说明〕发生 GAT 扇区错误后没有及时修复，就有可能发生同一个 gr. 分配给多个文件的错误，这时最后一个占用该 gr. 的文件会覆盖以前的内容，而以前的内容就会被破坏。在显示的错误信息中，bb, x 表示发生错误的 gr. 所在的 gr. 组编号 (bb) 及相对 gr. 的编号 (x)。yy 是先占有这个 gr. 的文件 1 在目录扇区中的 FPDE 的 DEC 码。nn 的含义与 yy 相同，只是它是属于文件 2 的，而且，文件 2 覆盖了文件 1。

遇到这种错误时，处理方法是：先把最后占用这个 gr. 的文件（如文件 2）用 COPY 命令复制到另一个盘片上，然后用 KILL 命令把原来盘片上的这个文件（文件 2）删去。接着把被覆盖的文件（如文件 1）用 LOAD 命令装入内存储器，修正覆盖时被冲掉的内容（如果不知道原来的内容，就无法恢复文件 1），然后用 SAVE 命令把文件 1 存回这个盘片或另一个盘片。最后修正 GAT 错误，这样就完成了错误的修正工作。但是一般情况下，被覆盖文件（如文件 1）是不容易复原的，所以应该经常用 DIRCHECK 程序检查使用盘片的目录状态，发现错误及时改正。

5.11 通行字

通行字 (PASSWORD) 是在建立文件的时候赋予文件的。可以使用 ATTRIB 和 PROT 命令更改通行字（请看第二章的 2.3 及 2.39 节）。具有通行字的文件只有在指明通行字时才能存取。所以，用户应该记住文件的通行字。

首先看一下磁盘的主通行字。这是赋予该盘片（注意，不是文件）的通行字，只有指明这个磁盘的主通行字才能用PROT命令更改该盘片的控制信息。磁盘的主通行字的散列字码（2 字节）保存在 GAT 扇区的相对字节位置 "CD" 和 "CE" 处。在 5.6 节 1. 给出的例子中，磁盘主通行字的散列代码是 "E042"（相应的通行字是 PASSWORD），这是一个通行字的 2 字节散列代码，前面第二章已经介绍过，通行字由 1—8 个字母数字组成，第一个字符必须是英文字母，后面可以是英文字母，也可以是数字。要计算通行字的散列代码，可以使用 SUPERZAP 程序的 DPWE 功能。当你按照 SUPERZAP 程序的询问，打入通行字作为响应，即可得到相应的 2 字节的散列代码。磁盘主通行字是供 DOS 命令 PROT 使用的。

当打入 PROT 命令，就会把磁盘的主通行字传送给所有的用户文件，作为这些文件的通行字。系统文件和隐性文件还是保留它们原来的通行字不变。如果在 PROT 命令中用了 UNLOCK 参数的话，将会把所有的用户文件的通行字全部修改成 8 个空格（相应的 2 字节散列代码为 "9642"）。这时系统就会不管通行字，换句话说，8 个空格所表示的通行字就说明文件标识符中可以不指明通行字。从上面的说明很容易想到：设立磁盘主通行字及 PROT 命令的目的之一是为遗忘文件通行字的用户提供一种恢复文件存取的手段。

设立通行字可以限制某些文件的存取范围。一个文件可以设立两种通行字，一种是修改通行字，另一种是存取通行字，这两个通行字分别放在该文件的 FPDE 内的第 17和18 字节及第 19 和 20 字节（请注意!! 磁盘的主通行字是放在 GAT 扇区内的）。当 SYSTEM 的参数 AA=Y 时（即系统内通行字有效），这些通行字就可以起作用。使用文件通行字时要注意，在打开一个文件时，修改通行字是优先于存取通行字的，也就是说，如果某个文件的修改通行字设定为 8 个空格（即空缺修改通行字），这时文件即使有一个合法的存取通行字，系统也得先去检查空缺的修改通行字，所以你在文件标识符中不指明任何通行字，该文件也

能作 FULL 级的存取（即允许进行任何操作）。当你为文件分别设定了修改通行字和存取通行字以后，系统先去检查修改通行字，然后检查存取通行字。设定了存取通行字以后，就可以使用文件保护措施，限定某些文件输入/输出时的存取级别（有关存取级别的详细情况，请看 2.3 节）。如果系统中通行字有效，某一个文件又具有通行字，则该文件的文件标识符中的通行字部分就不能省略，并且一般使用的是存取通行字，文件的存取范围由保护级决定。

如果使用修改通行字去打开一个文件，就可以对这个文件作 FULL 级存取，而不理会设定的保护级别，所以希望用户使用存取通行字，使得某些文件得到一定的保护。

第六章　NEWDOS/80磁盘上的其它附加程序

6.1 SUPERZAP　查看和修改磁盘或内存储器的内容

SUPERZAP程序为用户提供了读、写256字节的标准磁盘扇区或内存储器中任一部分的手段，但是不能往ROM内写数据。我们极力向用户推荐学习使用SUPERZAP程序。在学习使用SUPERZAP程序的时候，要在那些不带重要数据的盘片上练习，因为弄不好就会破坏盘上的有用数据。

某些盘片可能是以非标准扇区格式来写的，这时SUPERZAP程序就无能为力了。有另外的一些程序可以读任意格式的盘片，但是它们往往不具备SUPERZAP程序的某些功能。需要时，用户可以从其它软件公司买到这种程序。

SUPERZAP程序可以使用大写和小写两种型式的字符。

用到数值的地方，除非另有说明，SUPERZAP程序都假定它们是十进制的，加后缀H的则表示是十六进制的。

1. 工作模式

用户打入SUPERZAP命令以后，荧光屏上会显示出清单，它给出了SUPERZAP程序现有的各种功能。功能清单如下：

```
APPARAT'S SUPERZAP/80. INPUT ONE OF THE FOLLOWING FUNCTIONS:
'DD'    — OR NULL — DISPLAY DISK SECTOR
'DM'    — DISPLAY MAIN MEMORY
'DFS'   — DISPLAY FILE'S SECTOR
'DTS'   — DISPLAY TRACK'S SECTOR
'DMDB'  — DISPLAY MEMORY DUMP BLOCK
'VDS'   — VERIFY DISK SECTORS
'ZDS'   — ZERO DISK SECTORS
'CDS'   — COPY DISK SECTORS
'CDD'   — COPY DISK DATA
'DPWE'  — DISPLAY PASSWORD ENCODE
'DNTH'  — DISPLAY NAME/TYPE HASH CODE
'EXIT'  — END SUPERZAP, EXIT TO DOS
PRINTER OUTPUT. APPEND, P TO DD, DM, DFS, DTS OR DMDB
```

用户只要通过键盘打入所选中的代表相应功能的字符，然后按下ENTER键就可以了。SUPERZAP程序的功能如下：

＊ DD　显示磁盘扇区的内容。SUPERZAP程序将询问驱动器编号和盘片中的相对扇区编号，然后读这个扇区并把它的内容显示出来。

＊ DM　显示内存中256字节的一页。SUPERZAP程序将询问要显示的存储器地址，并由此开始截取256字节，把这一页显示出来。

＊ DFS　显示文件扇区内容。SUPERZAP程序将询问文件名和文件中的相对扇区编号。然后读取这个扇区，并把它们的内容显示出来。

* DTS　显示某一磁道的扇区内容。SUPERZAP 程序将询问驱动的编号。然后，SUPERZAP 程序将读取被选中扇区的内容，并显示出来。

* DMDB　显示存储器转储块。SUPERZAP 程序将询问存储器转储文件的文件标识符（转储文件是由 DUMP 命令建立的，请看 2.20 节）。SUPERZAP 程序将给出转储的基地址。接着要询问转储范围内的内存地址，并截取 256 字节，最后显示内存的这一页。

* VDS　复核磁盘扇区。SUPERZAP 程序将询问：遇到读保护扇区，用户是否想暂停一下。接着，SUPERZAP 程序还将询问驱动器编号和首先要复核的扇区在盘片上的相对扇区编号。最后要询问需要复核的扇区数目。然后，SUPERZAP 程序着手复核工作，复核就是读指定范围内的每一个扇区。当遇到一个保护的扇区，如果需要暂停，SUPERZAP 程序将显示出该扇区的位置，并等待用户按下 ENTER 键，以便继续进行复核工作。对于怀疑有损坏的盘片，VDS 是一种寻找坏扇区的快速方法。正在复核的时候，只要按下⏎键就可以取消 VDS 功能。

* ZDS　磁盘扇区清零。SUPERZAP 程序将询问驱动器编号和第一个待清零扇区的盘片上的相对扇区编号。接着还将询问待清零的扇区数目。然后进行清零。这个功能不改变每个扇区的读保护状态。

* CDS　复制磁盘扇区。SUPERZAP 程序将询问驱动器编号和源区（取数据的地方）第一个扇区所在磁盘上的相对扇区编号。接着对目的区要提出同样的一些问题。最后SUPERZAP 再询问要复制的扇区数目。然后复制。每个目的扇区要赋予与源扇区相应的读保护状态。

* CDD　复制磁盘数据。这个功能与CDS不同之处在于可以复制盘片上的任意字符串。SUPERZAP 程序将询问驱动器编号和包含源区第一个字节的扇区所在盘片上的相对扇区编号，然后询问该扇区中的字节偏移量。SUPERZAP 程序还将对目的区的第一个字节作同样的询问。最后询问要复制的字节数（最大值为 65535）。然后进行复制。不改变目的扇区的谈保护状态。

* DPWE　显示通行字的编码。SUPERZAP 程序将询问通行字，并对通行字编码，然后按目录 FPDE 中的 16 进制格式显示这个编码。

* DNTH　显示名字和类型的散列代码。SUPERZAP 程序首先询问文件名，接着询问类型（即文件名扩展符）。然后计算它们的散列代码，最后按目录 HIT 扇区中的十六进制格式显示算出的散列代码。

* EXIT　结束 SUPERZAP 程序，并返回到 440DH（即 DOS　READY）。

因为 ZDS, CDS 和 CDD 要改变盘上的数据，所以打入这些命令后，SUPERZAP 程序首先要询问用户是否确实要执行这些功能（即确认一下），得到答复后再去执行，这样可以防止键入错误命令。使用 CDS 和 CDD功能时要注意，复制通常是按字节序号递增的次序来进行的，对于源和目的两者都是如此。但是，如果最高的源字节在目的区中间，为避免那些破坏性的覆盖，复制就要以字节顺序递减的次序去进行。

所有的磁盘输入/输出都是通过通常的 DOS 扇区 I/O 子程序去完成的。如果发生错误，SYSTEM 的选用参数 AM 和 AW（重新试一下输入/输出）就能起作用。

使用 VDS, ZDS, CDS 和 CDD 时，如果发生磁盘输入/输出错误，SUPERZAP 程序可为用户提供选择的余地：例如可以打入 R，重新再试一下；打入 S，跳过有问题的扇区；或者

打入 X, 结束工作。在大多数情况下，重复的试几次，最终是会成功的。如果错误扇区是一个源扇区，则跳过这一扇区将会使接收到的那些相应的目的字节是源缓冲区中的随机数据，但是这不应该成为问题，因为用户可以想法去恢复它们。

当 SUPERZAP 程序正在等待输入数值时，这时若把 X 作为数值打入时，将会使 SUPERZAP 程序终止当前的工作，而返回到显示功能清单的状态。如果 SUPERZAP 程序正在等待输入文件标识符，用户不给回答，直接打入 ENTER 键将会终止当前的功能。

DD, DM, DFS, DTS 或者 DMDB 带有",P"后缀时，将会使扇区或存储器的内容在荧光屏上显示的同时，由打印机打印出来。使用 DD, P, DFS, P 或 DTS, P 时，SUPERZAP 程序将会询问用户要打印的扇区数目。使用 DM, P 或 DMDB, P 时将询问用户要打印的字节数目。如果打印机没有准备就绪，SUPERZAP 程序将进入循环等待，而不给用户任何提示信息。按下 P 键将会使正在打印的打印机暂停，按下 ENTER 键可以恢复打印。按下 H 键将会使正在打印的打印机停止工作。

. 显示模式

使用 DD, DM, DFS, DTS 和 DMDB 功能显示一个扇区或存储器页面时，SUPERZAP 程序处于显示工作方式，并等待用户发出显示模式命令。除了 F 和 L命令以外，打入的命令字符不会显示出来，也不需要用 ENTER 作结束。当一个显示模式命令的所有字符都打入以后，就立即执行这个命令。显示模式命令是：

* X 终止当前的功能，并使 SUPERZAP 程序返回到显示功能清单。
* R 重复显示同一个扇区或存储器页面 (256 个字节)。
* + (或;)显示下一个较高扇区或存储器页面。
* − 显示下一个较低扇区或存储器页面。
* J 重新开始相同的功能。
* K 重新开始相同的功能，并保持第一个参数不改变。例如 DD 命令，原来要询问驱动器编号和相对扇区编号这两个参数，现在保持第一个参数不变，只询问相对扇区编号。
* SCOPY 这个命令只适用于 DD和 DTS 功能。当前的扇区要复制到一个指定的扇区。SUPERZAP 程序将询问目的扇区的驱动器编号和相对扇区编号。目的扇区可以与源扇区相同。SUPERZAP 程序将读目的扇区，并报告目的扇区的状态。然后把源扇区的内容写入目的扇区。在发现扇区有奇偶错，但是除少数字节外，其余原封未动，这时就要用到 SCOPY 命令，用 SCOPY 复制扇区自身，将会产生新的奇偶校验值，于是该扇区就得到了修补。SCOPY 还可用来更改扇区的读保护状态。

当 SUPERZAP 程序处于显示工作方式时，它有搜索磁盘、文件、内存储器和转储文件的能力。可以搜索用 aa, bb, cc, dd 表示的 1 至 4 个16进制字节(不带 H 后缀)。当找到与之相同的字节串后，显示包含有找到的第一字节的扇区或存储区，并以淡色垂直闪烁光标来指出它的位置。按任意键都会使这个光标立即消失。但是，在需要继续进行搜索时，所找到的这个位置是不会丢失的。当 SUPERZAP 程序处于显示工作方式时，为了进行搜索，可以打入下列命令，但是要用 ENTER 作结束。

* F, aa, bb, cc, dd 把要找的 1 到 4 个十六进制字节保存起来，并从磁盘(如果处于 DD 或 DTS 方式)，文件 (如果处于 DFS 或 DMDS 方式) 或内存储器 (如果是 DM 方式) 的第一个字节开始搜索。

* F, 同上述命令的功能，差别仅在于不设定新的查找字节，还是利用上一次设定的查找字节。

* F xx, aa, bb, cc, dd 把要查找的1到4个16进制字节保存起来，从当前扇区或存储器中的第xx相对字节开始搜索，其中xx是2个十六进制数，不带H后缀。

* F xx 或 Fxx, 与上述命令的功能相同，差别在于不设定新的查找字节，还是利用上一次设定的查找字节。

* F 紧接着上次找到的第1字节的位置继续搜索，这次搜索使用上次设定的查找字节。

* L, aa, bb, cc, dd 在DFS方式中，当正在被搜索的文件是一个标准的装入模块（如SUPERZAP/CMD，LMOFFSET/CMD等），而且用户希望SUPERZAP程序在搜索中清除所有除实际的目的代码字节以外的东西时，就要用这个命令来替代F, aa, bb, cc, dd命令。这样就可以在查找过程中免除嵌入的装入控制信息的影响。最终显示将仍旧包含有装入控制信息，以备用户需要时去查看嵌入在待查找字节中的控制信息。通常(不总是这样)，控制字节是4个字节长，并以01（16进制）字节开始。除了能从待查找的字节中清除这种控制信息以外，L, aa, bb, cc, dd的工作是与F, aa, bb, cc, dd一样的。在L类命令的搜索以后，可以使用F命令继续搜索。

* L, 与上述命令功能相同，差别在于不设定新的查找字节，仍然利用上一次设定的查找字节。

* MOD xx 只适用于DD, DM DFS或DTS。执行这个命令，SUPERZAP程序进入修改工作方式，并把光标定位到当前存储器页面（或扇区）的相对字节xx(其值为00H—FFH)的第一个16进制数字处。

* EXIT 结束SUPERZAP程序，并返回到402DH(即返回DOS READY状态)。

用户在打入上述的命令(F, L和MOD)时，各打入的字符从上到下的垂直分布在荧光屏上的第7列。

如果在打入显示方式命令时发生错误，SUPERZAP程序就不理睬这个命令，并重新显示这个扇区或存储器页面的内容。

3. 修改模式

执行显示模式的命令MOD xx就可使SUPERZAP程序进入修改模式。修改模式可以使我们改变当前磁盘扇区或存储器页面中的各个字节。在修改模式中有如下功能：

* 打入0—9或A—F 在当前光标位置的一个16进制数字可以用打入一个新的16进制数字去更换，然后光标向前移动一个位置。如果光标移到相对字节FFH以后，再继续前移，光标就会绕回到相对字节00H，接着打入的字符如果是一个16进制数字，就会发生错误。在存储器页面内更改的内容是实实在在的改变了存储器的内容，而扇区中更改的只是缓冲区的内容，一直要到写扇区时才真正写入扇区，也可以用Q命令取消尚未写入磁盘的更改操作。

* 按下空格键或⇨键 光标前进一个位置。

* 按下⇦键 光标退回一个位置。

* 同时按SHIFT键和⇨键 光标前进四个位置。

* 同时按下SHIFT键和⇦键 光标退回四个位置。

* 按下⇩键 光标下移一个显示行。

＊ 按下↑键　光标上移一个显示行。

＊ ZTxx　在荧光屏上列7垂直显示这个命令字符串，这个命令必须用ENTER作结束。从光标位置（包括光标）到相对字节 xx 的第二个 16 进制数字（包括这个数字）的全部 16 进制数字都被清零。光标留在相对字节 xx 后面的第一个 16 进制数字处，如果光标移过相对字节 FFH 以后，继续前移，绕回到相对字节 00H，则下一个输入字符不可以是 16 进制数字。

＊ RTxx,jk　这个命令的功能与 Zxx 相似，但不是清零，只是每个字节的第一位 16 进制数字用 j（用户给定的以 16 进制表示的数字）来代替，而每个字节的第二个 16 进制数字用 k（用户给出的以 16 进制表示的数字）来代替。

＊ Q　只适用于扇区操作。终止修改模式，把缓冲区内修改的内容统统作废，SUPERZAP 程序返回到显示模式。

＊ ENTER　ENTER 对于存储器页面操作的作用是终止修改模式，SUPERZAP 程序返回到显示模式。ENTER 对于扇区操作来说，SUPERZAP 程序要询问用户是否现在真要更新这个扇区。如果不要更改，SUPERZAP 程序继续留在修改模式。如果要更新，就把更改过的扇区写回到磁盘上，并终止修改模式，然后 SUPERZAP 程序返回到显示模式。

在修改模式中遇到一个错误时，将显示 INVALID MODIFICATION MODE CHAR. REPLY '*' TO CONTINUE(打入的是非法的修改命令。打入"*"可以继续留在修改模式)。SUPERZAP 程序收到用户打入的＊后，就会返回修改模式。

4. SUPERZAP 程序应用举例

请用 SUPERZAP 程序显示或打印磁盘的目录文件。

要得到磁盘的目录文件，首先要知道目录放在磁盘上的确切位置，然后，选用 SUPERZAP 程序的某一功能去显示或打印目录文件的内容。

根据 5.6 节，可以知道每个盘片的第一扇区的第三字节总是存放目录的起始 gr. 组的编号，而且磁盘目录总是起始于 gr. 组的分界处。于是，我们先用 SUPERZAP程序去查看该字节的内容，具体操作是：在 DOS READY 状态（即处于系统命令级）打入命令：

SUPERZAP [ENTER]

这时荧光屏上将显示 SUPERZAP 程序的功能清单（请看本书第109页）。

接着打入：DD [ENTER]

荧光屏上将显示：

DRIVE AND RELATIVE—SECTOR—WITHIN—DRIVE #'S?

这是 SUPERZAP 程序询问需要显示哪一个驱动器和驱动器内盘片上的哪一个相对扇区。我们现在要看的是系统盘片，所以要看 0 号驱动器内盘片上的第一个扇区（即相对扇区编号 0）。所以应该回答：0,0 [ENTER]

这时荧光屏上显示出：

DRV	00	00FE	11F3	21EC	3736	FE36	D023	3600	2336
0	10	0011	0500	D931	E041	21FF	51CD	5242	FE20
0H	20	4730	2957	CD52	424F	CD52	425F	1012	CD52
	30	4257	0D0D	2CCC	5542	7E12	130D	20F6	18DB
DRS	40	10F9	CD52	4257	1AFE	A513	D5C8	21E5	42C3
0	50	C342	2C7E	C0D9	060A	21E1	3736	01D5	C57B
0H	60	D60A	3803	5F36	0921	EC37	CDCE	42ED	53EE
	70	3736	1BCD	CE42	3688	11EF	3701	0051	CDD7

TRK	80	427E	E683	E281	421A	0203	CB4E	C287	42CB
0	90	4EC2	8742	CB4E	20EF	CB46	2808	CB4E	20E7
0H	A0	CB7E	28E6	7E36	D0C1	D1E6	FC20	0C1C	7BD6
	B0	0A20	0314	1E00	D97E	C9CD	D742	360B	1098
TRS	C0	21DD	427E	FE03	28FB	23CD	3300	18F5	CDD7
0	D0	42CB	4620	FC7E	C93E	063D	20FD	C91C	1F45
0H	E0	5252	4F52	031C	1F4E	4F20	5359	5303	0000
	F0	FFFF	0000	FFFF	0000	FFFF	0000	FF00	0700

其中：

左边的几项标志含义是：DRV——驱动器编号，下面紧接着的是十进制和十六进制表示的被显示驱动器编号；DRS——该磁盘上的相对扇区编号，下面紧接着的是十进制和十六进制表示的被显示扇区的相对扇区编号；TRK——磁道编号，下面是用十进制和十六进制表示的被显示扇区所在的磁道编号；TRS——该磁道内的相对扇区编号，下面是用十进制和十六进制表示的被显示扇区在该磁道内的相对扇区编号。在上例显示的扇区实例中，各项数字表明被显示扇区是0号驱动器内盘片上的0号相对扇区。如果用磁道及磁道内扇区相对编号来表示，则被显示的扇区是0磁道内的相对扇区0。

从被显示扇区中可以看到第三字节的内容是11H(即17gr.组)。

假定每个gr.组等于2个gr.,每个gr.,由5个扇区组成。可以算出目录文件所在的相对扇区编号：17gr.组$=17\times2\times5=170$

为了从显示模式返回到SUPERZAP程序的功能清单状态，只要按下X键。返回功能清单状态后，再打入

DD, P [ENTER] （结果要送到打印机去打印）

接着荧光屏上将显示：

DRIVE AND RELATIVE-SECTOR-WITHIN-DRIVE #'S? （询问驱动器编号和驱动器内相对扇区的编号）

为了得到目录文件的扇区，可以回答：

0, 170 [ENTER]

接着SUPERZAP程序还将询问：

SECTOR COUNT? （要打印几个扇区）

这可以根据用户的具体要求打入数值。在第五章（第90页）给出了用SUPERZAP程序打印的NEWDOS/80 2.0系统盘片上的目录文件内容，总共是10个扇区，其中相对扇区0是GAT扇区，相对扇区1是HIT扇区，接着8个是FDE扇区。

6.2 DISASSEM Z80反汇编程序

在工作中经常要分析、了解别人编制的程序，但是这种程序往往是以可执行的目的码的形式提供的，并不给出源文本，因而无法阅读。这时就要借助于一种称为反汇编程序的专门程序来产生出相应的源文本。在原来的TRSDOS中并不包括这种程序，在NEWDOS中提供了这种反汇编程序，但是早期的NEWDOS提供的反汇编程序的功能不够完善，例如不能把反汇编得到的源文本（具有EDTASM的源文本格式）存入磁盘，也不能控制打印机的打印格式（如每页的行数，页间保留的空距等等），使用起来不太方便。而NEWDOS/80 2.0版本中附带的反汇编程序（3.0版本）功能比较完善。这个反汇编程序的文件名是DISASSEM/

CMD，它是一个目的码文件，能够对标准的TRS-80装入模块或者主机存储器内的Z80目的码进行反汇编。反汇编得到的结果可以送到荧光屏上显示，或者送到打印机去打印。产生的源文本可以存入磁盘，同时还将产生单元交叉访问表(Location cross reference table)。

因为反汇编程序是一个目的码形式的文件，所以在NEWDOS/80命令状态打入DISASSEM和ENTER键，就能调用反汇编程序，这时荧光屏上就会显示出：

APPARAT DISASSEMBLER 3.0

OBJECT FROM MAIN MEMORY OR DISK? (M OR D) (询问用户待反汇编的目的码取自存储器还是取自磁盘)

1. 如果你要反汇编的目的码是磁盘装入模块，可以回答D，接着打入ENTER键，也可以只打入ENTER键。

DISASSEM程序的反应就是继续提出要从磁盘提取装入模块所需的各种询问。

(1)荧光屏显示出：FILESPEC? (询问装入模块的文件标识符)

这时用户必须把要反汇编的磁盘装入模块的文件标识符打入作回答。

(2)荧光屏接着显示：OFFSET OBJECT VIRTUAL ADDRESSER BY (HEX)? (询问偏移目的码地址的值)

可以空缺（即直接用ENTER回答，这表示偏移值是0)或者用1到4个十六进制数(不带后缀H）作为回答。把这个偏移值加到装入模块中的装入地址上，将会得到一个合适的地址，所谓合适的地址，就是正要进行反汇编的目的码就在它实际执行时应该存放的地址。当目的模块根据装入地址装入存储器中的一个地方，而该目的码实际执行的地址却是另外一个地方，这时就要使用这个参数。加偏移地址值时，允许其结果超过64K字节，这时我们可以把它看作是拆转返回到的地址（即去掉64K字节后剩下的地址值)。例如： 如果磁盘目的模块装入地址是C000H—FFFFH，而目的码实际是在7000H—AFFFH地址执行，加上偏移量B000H以后，就可以使反汇编程序好象实际上是对已装入7000H—AFFFH的目的模块作反汇编。

(3)VIRTUAL RESTART LOCATION (HEX)? (询问实际重新开始反汇编的地址)

可以直接打入ENTER键(表示在文件开头的地方开始)或者是打入1到4个十六进制数字（不带后缀H)，这个数字表示了重新开始列出的反汇编指令的地址。这样可以使一个大的反汇编分段列出清单，这个重新开始的地址一般就选在一页内被中断打印处，把它作为重新开始的第一条指令。例如：

```
地址  目的码                    源程序
8000  210000          LD    HL, 0000H
8003  010000          LD    BC, 0100H
8006  325050    LOOP: LD    (5050H), A
8009  0B              DEC   BC
800A  79              LD    A, C
800B  B7              OR    A
800C  C20680          JP    NZ, LOOP
```

当我们对于VIRTUAL RESTART LOCATION (HEX)?回答8006,则可以显示或打印：

```
8006  325050    LOOP: LD    (5050H), A
```

```
8009   0B                    DEC  BC
800A   79                    LD   A,C
800B   B7                    OR   A
800C   C20680                JP   NZ, LOOP
```

2. 如果要反汇编的目的码已经在存储器内了，则在回答 DISASSEM 程序的询问：

OBJECT FROM MAIN MEMORY OR DISK? (M OR D) 时，要回答 M。

接着 DISASSEM 程序就要提出另外一些问题请用户回答。

(1)OBJECT VIRTUAL BASE ADDRESS? (HEX) (询问目的码的虚拟基地址)

用 1 到 4 个十六进制数字（不带后缀 H)来回答，可以认为目的码是从这里开始执行的，而不管实际上是否从这里开始。在得到的反汇编清单上，这个值将是第一条指令的地址值。假如存储器 8000H 单元开始，放有下列目的码（不包括右半部分的源程序)：

```
地址   目的码                        源程序
8000   210000                LD   HL, 0000H
8003   010001                LD   BC, 0100H
8006   325050        LOOP:   LD   (5050H), A
8009   0B                    DEC  BC
800A   79                    LD   A, C
800B   B7                    OR   A
800C   C20680                JP   NZ,  LOOP
```

对于 OBJECT VIRTUAL BASE ADDRESS?, 如果回答 0000, 则可以得到：

```
0000   210000                LD   HL, 0000H
0003   010001                LD   BC, 0100H
0006   325050        LOOP:   LD   (5050H), A
0009   0B                    DEC  BC
000A   79                    LD   A, C
000B   B7                    OR   A
000C   C20680                JP   NZ, LOOP
```

(2)第二个问题 OBJECT REAL BASE ADDRESS (HEX)? (询问目的码的真实基地址)

可以用空缺（直接打入 ENTER 键）作回答，表示真实基地址和虚拟基地址是相同的，或回答 1 到 4 个 16 进制数字（不带后缀 H）的存储器地址，反汇编程序将在这里找到目的码。

3. 下面关于选用参数的询问对于反汇编磁盘装入模块或存在于存储器内的目的码都是一样的，其中有些参数只供磁盘装入模块用，具体介绍各个参数时将会说明。

ANY OPTION? (询问是否使用选用参数)

可供选用的参数有：

(1)空缺（即直接打入 ENTER） 表示不再指定选用参数。

(2)PTR 把输出送到打印机去打印，代替显示器的显示。

(3)BFSP 避免满屏时暂停。在正常工作状态，反汇编程序在显示内容充满了整个屏幕

或者在反汇编文件中的某个位置上发生一个断点时，均会暂时停止工作。这时反汇编程序等待用户回答：

* 按下 ENTER 键，继续进行反汇编。

* 按下 X 键，终止反汇编。

* 按下 V 键（仅适用于取自存储器的目的码），在一个新的位置重新开始反汇编。

BFSP 参数可以避免发生这种暂停，使得反汇编连续进行，不断的给出显示。如果指定了 PTR 参数，就自动的使 BFSP 参数有效。

下面的这些参数只适用于目的码是取自磁盘的情况。

（4）NCR　不建立单元访问表，也不显示和列出它完成的工作。单元访问表是反汇编程序中对一些存储器地址或立即数进行访问的那些单元构成的表格，它按被访问数（包括数值和地址值）的大小以递增的顺序排列，每个数一行，右边是对它进行访问的单元值及访问类型后缀。例如前面举的例子中，8000H 开始的 3 个字节中，第一个字节为 21H，CPU取入这个操作码后，经指令译码，知道是立即数传送指令，同样，CPU 也能识别出 8003H 单元的立即数指令，8006H 处的访问 5050H 单元地址及 800CH 处的转移到 8006H 单元地址。对于反汇编程序来说，它也能识别出指令类型，并构成单元访问表，上例中的单元访问表为：

```
0000    8000U
0100    8003U
5050    8006X
8006    800CP
```

其中 U，X 和 P 等后缀表示了访问类型（各种访问类型及其后缀表示符的含义，下面有介绍）。

（5）NLP　不打印也不显示反汇编的指令。

（6）STD　把源文本送到磁盘上。反汇编的结果将以 EDTASM 的源文本格式送到磁盘上。详细内容，请看下面的讨论。

（7）FGN=xxx　这是反汇编目的码模块（磁盘）后，存入磁盘时（使用 STD 参数）为被访问数字设定的名字。xxx 是三个字母字符组成的名字，它是 STD 参数有效时为访问数指定的第一个名字。缺项名字是 AAA 。例如：

```
地址      目的码
8000      210000
8003      010001
8006      23
8007      0B
8008      79
8009      B7
800A      C20680
```

经反汇编，并取 STD 参数以后，如使用缺项 FGN（即用 AAA作为第一个名字），可得EDT-ASM 格式的源文本(存盘文件)：

```
行编号          源程序
00001           . ORG  . 8000H
```

```
00002          . LD    . HL, AAA
00003          . LD    . BC, AAB
00004    AAC   . INC    . HL
00005          . DEC   . BC
00006          . LD    . A, C
00007          . OR    . A
00008          . JP    . NZ, AAC
00009    AAA   . EQU    . 0000H
00010    AAB   . EQU    . 0100H
00011          . END   . AAA
```

(8)RTD　把单元访问表存到磁盘上。如指定了 STD 参数，在访问表建立以后，DISASSEM 程序将会询问 REFERENCE TABLE FILESPEC?　(询问访问表的文件标识符)。这时可以按文件标识符的格式为访问表文件规定一个文件标识符。访问表文件可以由一个用户建立的程序使用，也能够把两个或多个程序的访问表合并起来。

(9)REA　在列出访问表时，使所有的访问类型都有效，这是缺项参数。

(10)RE&　在列出访问表时，只使指定的访问类型有效，其中“&”表示 L,P,R,S,T, U, V,W 或 X 中的一个。访问类型在每个单元访问表清单开始的地方有定义，现列出如下：

L=CALL　(调用类访问)

P=JP　(转移类访问)

R=JR　(相对转移类访问)

S=LD DR, (NN)　(传送类访问)

T=LD A, (NN); IN A, (N)　(传送类访问)

U=LD DR, NN　(传送类访问——立即数)

V=LD SR, N; OP N　(传送类访问——立即数)

W=LD (NN), DR (传送类访问)

X=LD (NN), A; OUT (N), A　(传送类访问)

其中：DR 表示寄存器对，SR 表示寄存器，NN 表示双字节值，N 表示单字节值，OP 表示对立即数 N 进行操作（如 CP, OR 等操作)。

(11)RIA　在列出访问表时，所有的访问类型均不列出。

(12)RI&　在列出访问表时，指定的访问类型不列出，其中“&”表示 L,P,R,S,T, U, V, W 或 X 中的任意一个。

反汇编程序的工作过程大致可分为 4 个阶段：

(1)如果要求反汇编的目的码来自磁盘，并且不指定 NCR 选用参数，DISASSEM 程序就会在荧光屏上给出 BULIDING CROSS REFERENCE TABLE(建立交叉访问表)信息，同时开始扫描要建立访问表的目的码。对于一个比较大的程序，要求反汇编建立交叉访问表是要费些时间的。如果计算机内的存储器容纳不下这个访问表，将会终止反汇编。

(2)如果指定了 RTD 参数，在建立访问表以后，在这个阶段就要把单元访问表写到磁盘上。

(3)接着，反汇编得到的源程序就会列出在荧光屏上或者在打印机上打印出来。其形式

如下：

	地址	目的码	源程序		目的码相应的ASCII字符
1	8000	210000	LD	HL, 0000H	! . .
	8003	010001	LD	BC, 0100H	...
2C	8006	325050	LD	(5050H), A	2PP
	8009	0B	DEC	BC	.
	800A	320780	LD	(8007H), A	2..
	800D	79	LD	A, C	Y
	800E	B7	OR	A	.
	800F	C20680	JP	NZ, 8006H	. . .
	8012	C30080	JP	8000H	. . .

其中，左半部分是从8000H单元开始要反汇编的目码，右半部分就是反汇编得到的指令，最右边的几列是被反汇编的目的码所对应的ASCII代码，其中·代表不可打印字符（即ASCII代码表中20H以下的控制码，以及80H以上的未定义码）。

在上述清单中，最左边第一列表示指令字节访问过的次数，其值是十六进制表示格式，空位表示没有访问过，F表示访问过15次及15次以上。上例中8000H单元开始的一条指令（LD HL, 0000H）被访问过一次，8006H单元开始的一条指令（LD (5050H), A）被访问过2次，其余各指令均未被访问过。第二列指出该指令中那几个字节是被访问过的，列2的值也是一个16进制数，但它被看作为4位二进制的屏蔽位，每位代表一个字节，屏蔽位从左开始为第一字节，第二字节……，若屏蔽位为"1"，则相应的那一个字节被访问过，"0"表示没有访问过。例如上例中，8000H单元开始的那一条指令，其列1=1，列2=0，表示只有该指令的第一个字节被访问过一次。而8006H单元开始的指令，其列1=2，列2=C，表示该指令被访问过两次，而列2所含信息指出了被访问的具体字节，这里列2=C，其相应屏蔽码为1100，因为屏蔽码是从左算起的，所以可知该指令中的第一字节及第二字节是被访问过的，这个结果从反汇编得到的指令清单上是显而易见的，即800FH单元的指令(JP NZ, 8006H)要访问它的第一个字节，而800AH单元的指令（LD (8007H), A）要访问8007H单元，也就是它的第二个字节。由于DISASSEM程序能够提供这种访问信息，所以为分析反汇编得到的源程序提供了方便。

如果选用了参数STD，这个阶段还将把反汇编得到的源文本写到磁盘上保存起来，供进一步的处理和应用。

（4）如果要反汇编的目的码取自磁盘，并且不指定NCR参数，就要显示或打印单元交叉访问表。首先在访问表的开始处列出访问类型代码的定义。然后，按数字递增的顺序，根据每条访问过的指令的位置列出每个访问单元。每个访问单元地址的后缀是被访问的Z80指令的访问类型代码。上例的单元访问表显示格式如下：

L=CALL

P=JP

R=JR

S=LD DR, (NN)

T=LD A, (NN), IN A, (N)

U=LD DR, NN

V=LD SR, N, OP N

W=LD (NN), DR

X=LD (NN), A, OUT (N), A

0000 8000U

0100 8003U

5050 8006X

8000 8012P

8006 800FP

8007 800AX

END OF LOCATION REFERENCE TABLE

ANOTHER DISASSEMBLY? (Y OR N)

（单元访问表结束，是否要对另一个目的码程序进行反汇编）

如果反汇编程序在目的模块中发现有问题，则会显示 DISK OBJECT FILE FORMAT NOT AS EXPECTED（磁盘目的文件格式不符）或 PAST END OF FILE（超过了文件的结束）信息，并终止反汇编。

在显示和打印反汇编指令的过程中，若按下 P 键，将会暂停工作；按下 ENTER 键，可以继续刚才停下的工作；按下 X 键，将终止反汇编的进行。DISASSEM 程序正在等待用户回答的时候，同时按下↑键和 ENTER 键，可以终止反汇编程序的工作。

如果要反汇编的目的码已经在计算机的存储器内了，用户可以随意地移动反汇编的起点。当反汇编暂停时，只要按下 V 键，就会在荧光屏上显示出：VIRTUAL RESTART LOCATION (HEX)?（询问重新开始反汇编的地址）信息。用户可以用 1 到 4 个十六进制数作回答，这个值就是反汇编重新开始的存储器的地址。这个功能是很有用的，有时发现反汇编得到的源文本有问题，这可能是目的码中夹杂有数据，当把数据找出来以后，就需要重新从数据区终点开始反汇编，以便得到正确的源程序，这时就要用到这个功能。在目的码取自磁盘的情况，也有同样的功能，只是不必按 V 键，在装入磁盘目的模块以后，DISASSEM 程序会自动提出询问，详细内容请看前面 1. 的部分。

如果指定了 PTR 参数，并且所有要选用的参数均已设定完毕以后，DISASSEM 程序将会向用户提出有关打印格式方面的一些询问，用户可以按自己的要求，一一给予答复。所提问题有下列这些：

(1) # LINES PER PAGE, EXCLUDING TOP AND BOTTOM MARGING? (1—255)

（询问每页要有多少行，不包括页顶，页底的空白页边）

可以根据用户的要求来回答每页所需的行数，取值范围为 1—255。

(2) # LINES EACH FOR TOP AND BOTTOM MARGING? (0—10)（询问在每页的顶部和底部要空出的页边所占行数）

这个问题是为用户提供了编排打印机输出格式的功能，用户可以根据自己的要求编排整齐的打印格式，可以设定页顶和页底的空白页边宽度，取值范围是 0—10。如果回答 0，则反汇编程序使打印输出不分页，连续打印各行。这种留出页顶和页底空白页边的能力，是包含在 DISASSEM 程序内的功能，打印机驱动程序本身并不具备这种能力。

(3)REPLY "ENTER" WHEN PRINTER AT TOP OF PAGE（告诉用户当打

印机打印头处于纸的页顶时再按 ENTER 键)

当你回答了上面的问题(2)以后，荧光屏上将会显示出上述信息。当打印机处于工作状态，而且打印头位置已经调整在打印纸的页顶时，这时按 ENTER 键才会获得整齐的打印清单。因为打印机本身并不能够识别和自动调整到页顶，因而 DISASSEM 程序给出上述提示信息，请用户调整好打印机的走纸位置，然后再按 ENTER 键。

(4)HIGH ASCII CODE FOR PRINTER? (5A—FF HEX) (询问用户的打印机可打印的最高 ASCII 代码)

因为目前市场上打印机的种类和型号很多，而且功能上有很大差别，有的打印字符种类很多，有的只有标准的字符可打印，因而 DISASSEM 程序设置了这个问题，它要用户根据自己所用打印机的可打印字符来回答。如果打印机对于英文字母只能打印大写字母，则可打印的最高 ASCII 代码就是 5AH (相当于英文字母 Z)，这时就要回答 5AH。回答的数要用 2 位十六进制数字来表示 (不带后缀 H)。打印机可打印字符在打印机说明书中可以查到。

下面介绍一些与 STD 参数有关的情况。STD 参数使要反汇编的目的代码变换成 EDTASM 格式的源文本代码。这个源文本 (如果不太长的话) 可以被装入存储器，并可用 EDTASM 程序进行汇编。使用 STD 参数以后，输出源文本的工作过程如下：

在经过建立交叉访问表阶段和访问表存盘 (RTD 参数) 阶段以后，DISASSEM 程序将会询问：ASSEMBLY SOURCE TEXT OUTPUT FILESPEC? (询问输出到盘上的汇编语言源文本的文件标识符)。必须要为经过反汇编得到的这个源文件规定一个文件标识符，规定文件标识符以后，这个文件将被打开，并在反汇编阶段把产生的源文本发送到该文件。

在要反汇编的目的码中，所有代表数值的代码都要用表征此值的唯一的字母名字 (由 3 个字符组成) 来代替。这些数值的字母名字是按字母顺序来分配的，例如第一个数值赋予 AAA 的字母名字，接着的一个就赋予 AAB, 再后面就顺序分配 AAC, AAD...,而第一个字母名字是由选用参数 FGN 来指定的，如果不指定具体的字母名字，就采用缺项值 AAA。如果有下列的目的码程序：

地址	目的码
8000	210000
8003	010001
8006	23
8007	0B
8008	79
8009	B7
800A	C20680

经反汇编，并选用 STD 参数，以及给定了存盘的源文本的文件标识符以后，就可以把反汇编得到的源程序存入磁盘。这个源程序可以打印出清单，具体清单如下：

行编号		源程序
00001	. ORG	. 8000H
00002	. LD	. HL, AAA
00003	. LD	. BC, AAB
00004	AAC . INC	. HL

```
00005          . DEC   . BC
00006          . LD    . A, C
00007          . OR    A
00008          . JP    . NZ, AAC
00009    AAA   . EQU   . 0000H
00010    AAB   . EQU   . 0100H
00011          . END   . AAA
```

其中 800AH 单元开始的一条指令 (JP NZ, 8006H) 的第二、三字节是数值代码，它实际上对应着一个反汇编单元 8006H, 因而存盘时就把它的字母名字 AAC 放在这条指令的单元名字区（请看行编号 00004 处的那一行）。

如果一个数值代码不对应一个反汇编单元，则为了使这个字母名字与数值相等，就要在源文本的结束处产生一个 EQU 语句。例如上例中8000H单元的指令的第二、三字节(0000H), 8003H 单元的指令的第二、三字节(0001H), 分别在行编号 00009 及00010 处放有EQU 语句。

根据需要还将产生 ORG 语句（如行编号 00001 处的那一行），并把 END 语句作为源文本的最后一行（如行编号为 00011 处的那一行)。这些内容在 8000H 单元开始的目的码中是看不到的，是由 DISASSEM 程序根据要反汇编的目的码模块产生的。

6.3 LMOFFSET 把模块移到新的装入位置

LMOFFSET 程序可以装入和执行一些通常驻留在 RAM 中的、DOS 所不能装入的程序。它可以告诉你被装入程序的地址及其入口，这也是反汇编程序必须知道的参数。使用 LMOFFSET 程序不需要知道被装入的程序在磁盘上的分布情况，只要打入文件名和文件名扩展符就可以了。

LMOFFSET/CMD 程序读取磁带或磁盘的装入模块，显示它的装入信息，随意改变程序的装入区，随意附加一个辅助程序，这个辅助程序可以使装入模块在执行时从它的装入区移动到执行区，随意编制在非磁盘 BASIC 管理下（即 DOS 不起作用）经 SYSTEM 运行的模块，并且能把模块以新的名字存到盘上或带上。

LMOFFSET 程序有如下功能：

1. 从磁带上读取磁带格式的汇编装入模块，或者从磁盘上读取磁盘格式的汇编装入程序。当打入 LMOFFSET ENTER 以后，荧光屏上将会显示

APPARAT LOAD MODULE OFFSET PROGRAM, VERSION 2.0 SOURCE FROM DISK OR TAPE? (D OR T)

如果是从磁盘上读取装入模块，LMOFFSET 程序要询问用户源文件的文件标识符。

从磁带上读取装入模块，当 LMOFFSET 程序已准备好接收磁带数据时将显示单个的 * 记号。用户要快速倒带（如果需要的话），倒好带后按下磁带机的 PLAY 键。当已经完成了读带同步时会出现 * * *。当遇到一个有问题的检查和 (checksum) 时，将显示字符 C。如果遇到了规定以外的引导数据字节，将会显示字符 P。如果遇到了规定以外的嵌入字节，将显示字符 I。

2. 模块装入后，LMOFFSET 程序将显示

MODULE LOADS TO 4D00—6FF9

MODULE LOAD OVERLAPS DOS (4000—51FF)

ENTRY POINT=4D00

带下划线的部分随装入模块的不同而改变。以上显示告诉用户：

（1）模块将要装入的区域，

（2）将要与系统存储区发生的抵触，

（3）模块的入口地址。如果打算用辅助程序，则入口地址就会被设置在这个辅助程序内。

3. 接着显示

NEW LOAD BASE ADDRESS (HEX)?

这是询问新的装入地址。用户可以用一个新的装入地址作答复。如果还是用现有的装入地址，只要按一下ENTER键就可以了。如果用户只是传送装入模块，不需要改变它，那么在第一次询问新的装入地址时就可以用 ENTER 作回答，这样，LMOFFSET 程序将直接进入下面的第7步。

4. 如果指定了新的装入地址，这时将显示

SHALL APPENDAGE BE SUPPRESSED? (Y OR N)

LMOFFSET程序将要询问是否要禁止辅助程序的工作。

（1）如果禁止了辅助程序的工作，因为没有辅助程序把这个装入模块移到它的执行区，因此得到的模块只能经DOS库命令LOAD使用，通过LOAD命令能使用得到的装入模块，把两个或多个装入模块装入主存储器，然后作为一个装入模块由DOS库命令DUMP贮存。

（2）有两种方式的辅助程序，一种是在DOS正常工作情况下的辅助程序，还有一种是DOS不起作用时也能工作的辅助程序。如果没有禁止辅助程序，则 LMOFFSET 程序将把DOS有效的辅助程序或DOS不起作用时用的辅助程序附加到用户程序上，到底用哪一种，主要取决于执行这个模块时是否要禁止DOS的正常工作。

5. 如果已经指定了一个新的装入地址，LMOFFSET返回到上面的3处，显示得到的装入信息并询问新的装入地址。如果给出了另一个装入地址，它将勾销前一个设定的装入地址，包括它的预先设定的辅助程序。

6. 最后，当对上述3的回答是空缺（即只打入ENTER键）时，如果已经指定了一个新的入口地址，而且不禁止辅助程序工作，LMOFFSET就要询问DOS是否要被禁止，荧光屏显示：DISABLE DOS? (Y OR N)，如果回答Y,就选用DOS被禁止的辅助程序，如果回答N,就要选用DOS有效的辅助程序。

7. LMOFFSET程序接着要问是写入磁盘还是磁带。如果是磁盘，LMOFFSET程序就要询问被建立的装入模块的文件标识符。

如果是写入磁带，LMOFFSET程序就要询问磁带模块的名字,以及用什么带速(L或H)。接着，要求用户在装好磁带并按下录音键以后再按ENTER键。

8. 然后，得到的装入模块要被写入磁盘或磁带。如果指定了一个新的装入地址，则：

（1）要更改每个目的码记录的装入地址。

（2）如果要用辅助程序，就要把额外的目的代码（辅助程序）插在入口地址的记录之前，并把入口地址设定为辅助程序的第一个字节。

（3）如果指定了新的装入地址和禁止附加辅助程序，这时就会根据新的装入地址自动修

改入口地址，如：

```
MODULE LOADS TO 7000—94FF
ENTRY POINT=8D00
NEW LOAD BASE ADDRESS (HEX)? 8000
SHALL APPENDAGE BE SUPPRESSED? (Y OR N) Y
MODULE LOADS TO 8000—A4FF
ENTRY POINT=9D00
```

（有下划线的部分是用户打入的）

9. 当完成了目标文件的写入操作，或者在上面第7,8阶段内发生一个错误或者写入成功，LMOFFSET程序就要询问是否相同的模块要写到另一个文件内(也可以是相同的文件)。如果是，就要重复上面第7和第8的操作。

10. 当一切工作均已完成，或者在第7,8步中没有发生错误或其它形式的终止，LMOFFSET程序就要询问是否要处理另一个源装入模块。如果要处理，就要回到上述的第1步；如果不要处理，就要退出LMOFFSET程序返回到DOS。

随时都可用↑键终止当前的LMOFFSET功能。如果LMOFFSET程序正在等待用户回答，为了终止LMOFFSET程序的工作，可以在保持按下↑键的同时按下ENTER键。

如果从一个LMOFFSET程序执行后得到的输出要作为另一个LMOFFSET程序的输入，于是，一个模块就可能以多个辅助程序作结束，但是极力劝阻用户这样做，在这种情况下，若有一个辅助程序是DOS不工作的辅助程序，它必定不能被执行。所以LMOFFSET程序一点儿也不知道前一次执行LMOFFSET程序传送给现在执行的LMOFFSET辅助程序的信息，使得工作无法进行下去。

LMOFFSET程序不能完成任何目的码的浮动定位!!!它只能把目的码分配到新的装入地址，因此它能在不破坏DOS的情况下从磁盘装入模块。

如果源模块装入到显示区(3C00H—3FFFH)，并且没有超出这个区，将不修改这些目的码记录的装入地址。

由LMOFFSET程序加到模块上的辅助程序用64个字节的0作为开始。这个区可给用户放修补用的代码。如果只有一个辅助程序，修补区的装入地址与模块得到的入口地址是相同的。当该程序执行时，放在这个区的Z80修补码将首先执行。这个工作应在把该程序移到执行地址和禁止DOS（如果DOS要禁止）之前去完成。

当一个程序要在DOS区的任意部分运行时，必须指定采用DOS被禁止的那一种辅助程序。DOS被禁止的辅助程序可以执行用户程序，就像在非磁盘BASIC的SYSTEM功能管理下从磁带装入这个程序，然后再执行一样。

下面介绍执行得到的用户模块的工作过程。对于DOS有效的辅助程序：

（1）执行用户提供的在64个字节修补区内的代码。

（2）把主程序移到它的执行地址。

（3）开始执行主程序。

对于DOS不工作的辅助程序：

（1）执行用户提供的在64个字节修补区内的代码。

（2）把显示荧光屏上的内容移到存储器的高区。

(3)显示下列信息:

RECORD AND THEN PERFORM THE FOLLOWING INSTRUCTIONS

(记录和执行下面的命令)

*HOLD DOWN BREAK KEY AND PRESS RESET TO ACTIVATE NON-DISK BASIC.

(按住 BREAK 键,并按下 RESET 按钮,使得非磁盘 BASIC 工作)

*RELEASE BREAK KEY AND ENTER BASIC INITIALIZATION RESPONSES.

(放开 BREAK 键,同时进行 BASIC 初始化对话)

*ENTER "SYSTEM".

(打入"SYSTEM")

*ENTER ".".

(打入".")

当操作者完成上述操作后,辅助程序继续执行。

(4)把存储器高区的内容恢复到荧光屏上。

(5)把主程序移到它的执行地址。

(6)开始执行主程序。

6.4 DIRCHECK 检查和列出磁盘目录

DIRCHECK 程序可以测试和列出你要查看的磁盘目录。如果在检查目录过程中发现有错误,在目录清单之前会列出这些错误(有关目录错误信息的内容,请看 5.10 节)。DIRCHECK 程序也可以有选择地清除(不是修改)目录。DIRCHECK 程序还可以作为单密度磁盘在 NEWDOS/80 管理下的 TRS-80 I 型和 III 型之间来回转换的辅助手段,即有选择地去写有保护的目录。

对于问题 OUTPUT TO PRINTER (是否要输出到打印机),如果要输出到打印机,则回答 Y; 如回答 N, 表示只要输出到显示器。

对于问题 WHICH DRIVE CONTAINS TARGET DISKETTE (哪个驱动器内放有要查看目录的盘片),就要回答放有待查看目录盘片的驱动器编号(用十进制数表示)。

DIRCHECK 程序读 BOOT 扇区(该磁盘的第一个扇区),并测试开头的两个字节是否分别为 00H 和 FEH。如果是,DIRCHECK 程序就用第 3 个字节作为目录开始的 gr.组编号,该 gr.组的第 1 扇区就是目录开始的地方。如果不是,DIRCHECK 程序就显示 ***** DISKETTE 1ST SECTOR NOT "BOOT". ASSUMING DIRCTORY START ON LUMP 17 DECIMAL. (磁盘的第 1 扇区不是"引导"扇区。假定目录是从第 17gr.组开始的)。

DIRCHECK 程序继续读这个目录。在以前几个版本的 NEWDOS 中,DIRCHECK 程序拒绝处理不是写保护的目录。因为现在要在 NEWDOS/80 的 I 型和 III 型之间互换单密度盘片,所以可以处理不带写保护的目录,但是,同时荧光屏上要显示两种信息,这时先显示一种信息,列出文件后还将显示另一种信息。先显示的信息是***** AT LEAST ONE DIRECTORY SECTOR UNPROTECTED. (至少有一个目录扇区没有保护)如果这个信

息和另外的一些错误信息一起出现，用户就可以认为DIRCHECK程序没有找到目录，不应该执行W功能（下面要讨论）。

DIRCHECK程序用驱动器的PDRIVE（见2.37节）数据去测定由目录登记的gr.组数和gr.数。如果PDRIVE的数据不符合这个盘片的情况，很可能DIRCHECK程序列出的错误并不是实际存在的情况。

如果目录有问题，接着就会被列出来。如果列出一个数字，它就是供SUPERZAP程序用于修正目录的16进制数。除非你十分了解目录结构、对所做的工作确有把握，否则请不要随意修改有问题的目录！比较好的办法是用COPY命令把有价值的文件摘取出来，然后把目录有问题的盘片重新格式化。

如果问题出在文件的目录项上（主目录项或扩充目录项），则给出的十六进制代码就是文件主目录项的DEC代码。若问题涉及到文件的扩充目录项，但不指定文件名和扩展符，则给出的16进制代码是文件扩充目录项本身的DEC码。若问题涉及到HIT扇区字节，则十六进制代码就是HIT扇区内有问题的字节的相对位置。若问题涉及到GAT扇区字节，则十六进制代码就是该字节在GAT扇区的相对位置。若问题涉及到gr.，则这个十六进制值就表示成bb,x格式，其中bb表示gr.组编号，也就是GAT扇区中表示这个gr.组的相对字节位置，而x表示gr.组中的相对gr.编号，也就是GAT扇区中对应这个gr.组的字节的位的编号，该字节中的各位从右边开始计算，最右边为位0。

接着列出的是磁盘的名字和日期。

再下面就列出文件目录，并带有十进制的数值和字符（字符含义说明如下）：

* S　　系统文件。
* I　　隐性文件。
* P=nnn　文件的存取级别，修改和存取通行字两者都不空缺。
* EOF=sss/bbb　文件结束值。
  sss=文件中的相对扇区。
  bbb=扇区中的相对字节。
* nnn EXTS　nnn是范围元的数目，每个FDE最多有4个范围元，用于表明文件所占的磁盘空间。
* nnnn SECTOR　分配给这个文件的扇区数。

最后，显示这个盘片上空闲的gr.数目和封锁的gr.数目。如果这个盘片的容量超过60H（十进制为96）个gr.组或者GAT的相对字节60H等于FFH，则DIRCHECK程序认为没有封锁表（即存在表）。注意，NEWDOS/80不把gr.标记为封锁状态，保留封锁表仅仅是为了能与TRSDOS兼容。

只要有一个目录扇区是不被保护的，那就要在荧光屏上显示另一种信息 FUNCTION COMPLETED（已完成DIRCHECK），接着提出如下问题：

REPLY

N　TO EXIT PROGRAM（退出DIRCHECK程序）

Y　IF ANOTHER DISKETTE FOR SAME PRINTER SPECS（是否另有一个磁盘要作同样的打印输出处理）

I　FOR PROGRAM RE-INITIALIZATION（重新开始执行DIRCHECK程序）

W TO WRITE DIRECTORY SECTORS PROTECTED (写保护目录扇区)

C TO CLEAN UP (NOT REPAIR) THE DIRECTORY (清目录)

挑选下列字符中的一个回答上述问题：

N 退出 DIRCHECK 程序，回到 DOS (在 402DH)。

Y 另外的一个磁盘要作目录检查，但是对打印处理的要求与上一个磁盘相同。

I 另外的一个磁盘要作目录检查，但是要改变对打印处理的要求。

W 读目录扇区，并重写保护状态。有关内容可参考 DOS 命令 WRDIRP (见 2.49节) 和 SYSTEM 的选用参数 BN (见 2.46 节)。这个功能只对单密度盘片有意义，可用来完成从 I 型转换到Ⅲ型，或相反，或相互传送。

C 把目录中的所有不用的 FDE 清零。这只是一种修饰性的功能，它从不再用的 FDE 中清除剩余的信息。通常，当 DOS 经 KILL 命令或自动空间释放功能去解除 FDE 时，仅使 FDE 的第 1 个字节的位 4 清 0，保留其余的一些信息，使得有经验的用户要想为自己恢复文件或扇区的时候还有一点可能性。

在显示或打印期间可以按下列键：

BREAK——可以暂停显示或打印，停留在当前的行或行组的结束处。

ENTER——继续显示或打印。

↑ ——终止正在进行的显示或打印。

下面给出一个使用 DIRCHECK 程序打印出的目录清单：

```
NEWDOS80        00/00/00
BOOT/SYS        SIP=6   EOF =  5/0    1  EXTS   5  SECTORS
SYS6/SYS        SIP=7   EOF = 35/0    1  EXTS  35  SECTORS
SYS14/SYS       SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
BASIC/CMD       IP=0    EOF = 18/0    1  EXTS  20  SECTORS
DIR/SYS         SIP=5   EOF = 10/0    1  EXTS  10  SECTORS
SYS7/SYS        SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
SYS15/SYS       SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
EDTASM/CMD              EOF = 35/0    1  EXTS  35  SECTORS
SYS0/SYS        SIP=7   EOF = 15/0    1  EXTS  15  SECTORS
SYS8/SYS        SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
SYS16/SYS       SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
CHAINTST/JCL            EOF =  1/50   1  EXTS   5  SECTORS
LMOFFSET/CMD            EOF = 10/0    1  EXTS  10  SECTORS
SYS1/SYS        SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
SYS9/SYS        SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
SYS17/SYS       SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
ASPOOL/MAS              EOF = 10/0    2  EXTS  10  SECTORS
CHAINBLD/BAS            EOF = 19/58   1  EXTS  20  SECTORS
SYS2/SYS        SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
SYS10/SYS       SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
SYS18/SYS       SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
SYS3/SYS        SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
SYS11/SYS       SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
SYS19/SYS       SIP=7   EOF =  5/0    1  EXTS   5  SECTURS
SYS4/SYS        SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
SYS12/SYS       SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
SYS20/SYS       SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
DISASSEM/CMD            EOF = 25/0    1  EXTS  25  SECTORS
DIRCHECK/CMD            EOF = 15/0    1  EXTS  15  SECTORS
SYS5/SYS        SIP=7   EOF =  5/0    1  EXTS   5  SECTORS
```

```
SYS13/SYS          SIP=7     EOF = 5/0        1    EXTS     5 SECTORS
SYS21/SYS          SIP=7     EOF = 5/0        1    EXTS     5 SECTORS
SUPERZAP/CMD                 EOF = 30/0       1    EXTS    30 SECTORS
93 FREE GRANULES.    0 LOCKED-OUT GRANULES.
```

6.5 EDTASM 磁盘编辑/汇编程序

这个磁盘编辑/汇编程序是 Apparat 公司从 TRS-80 的磁带编辑/汇编程序改编而成的，并有如下特点：

（1）可以从磁盘或磁带读取文本。

（2）可以把文本和目的码写入磁盘或磁带。在所有的扇区写完以后，就可以读取磁盘文件了。

（3）按⊡键可以使显示的文本上移一帧(15行)。

（4）防止混淆与 DEFM（定义字符串）有关的打印机输出。DEFM 只列出目的码的第一个字节。

（5）用访问表列出按字母次序排列的符号。

（6）能识别小写字母，并把它变换成大写字母。

这个 EDTASM 程序实际上与 NEWDOS/21 和 NEWDOS/80 1.0 版本提供的 EDTASM是相同的，其差别仅在于：

（1）现在作为 A 命令（汇编命令）的一部分，在显示 TOTAL ERROR 信息以后，EDTASM 程序还将显示文本区剩下的字节数，这就使用户能够知道离符号表溢出或文本缓冲区溢出还差多少字节。

（2）对于 TRS-80 Ⅲ型来说，目的码不能输出到磁带。用户必须先把目的码送到磁盘，然后再用 LMOFFSET 程序把它复制到磁带上。

由于 Apparat 公司提供的磁盘编辑/汇编程序是根据 TRS-80 的磁带编辑/汇编程序改进而成的，因而原来对磁带适用的命令，这里也能使用，因此这里不再列出，读者可参考《微计算机实用手册》299—322页（海洋出版社出版)。这里只给出补充的几个新命令。

1. 为了把一个文本模块装入文本缓冲区，可以打入下面的命令：

（1）LD=filespec 1　这适用于从磁盘上读入文本的情况。

（2）LT=nnnnnn　这适用于从盒式磁带上读入文本的情况。

其中 filespec 1 是指要从磁盘上装入到文本缓冲区的汇编语言文本模块的文件标识符，而 nnnnnn 是指要从磁带上装入到文本缓冲区的汇编语言文本模块的名字。

举例：

（1）LD=OLDTEXT/SRC:1

表示从目前装在 1 号驱动器内的盘片上把汇编语言文本文件 OLDTEXT/SRC 装入文本缓冲区。

（2）LT=OLDTXT

表示从磁带上把汇编语言文本文件 OLDTXT 装入文本缓冲区。

如果文本缓冲区内已经放有文本，显示器荧光屏上就会出现 TEXT IN BUFFER ARE YOU CONCATENATING??? (缓冲区已有文本，新文本是否要接在后面)。如果不打算把新装入的文本连接在原有文本的后面，可以打入 N 作为回答，这样在装入指定的文本模块之前，

就会清除缓冲区，使缓冲区空出来。如果要把新文本接在原有文本的后面，可以回答Y,这样新装入的文本就会附加在原有文本的后面。但是装入新文本时，并不管这些文本有没有重迭的行编号，所以装入工作完成以后用户应该使用 EDTASM 程序中的 N 命令，重新排列行编号，保证缓冲区内的行编号按递增的次序排列。

2. 存储文本模块的命令：

(1)WD=filespec 2 文本存入磁盘。

(2)WT=nnnnnn 文本存入磁带。

其中 filespec 2 是要从缓冲区写入磁盘的磁盘文件的文件标识符，而 nnnnnn 是要写到磁带上的文本文件名，它是由 1—6 个字符组成的名字。

举例：

(1)WD=NEWTEXT/SRC:1

把当前文本缓冲区内的汇编语言源文本（不是目的码）写到装配在 1 号驱动器内的磁盘上的文件 NEWTEXT/SRC 内。

(2)WT=NEWTXT

把当前文本缓冲区内的汇编语言源文本写到磁带上，并定名为 NEWTXT。

3. 对于不指定 NO 参数的 A 命令，EDTASM 程序会提出询问 OBJECT FILE TO DISK OR TAPE? REPLY D OR T?（目的文本存到磁盘还是磁带，用 D 或 T 来回答）回答是：

(1)T（只适用于 TRS-80 I 型）

目的码将存到磁带上。程序名是在 A 命令中给出的。

(2)D

目的码将存放到磁盘上。EDTASM 程序接着将询问 OBJECT FILESPEC?（请给出目的码文件的文件标识符),可以用目的模块的文件标识符（格式是：nnnnnnnn/ttt.ppppppppp:d）来回答。这时将立即打开这个文件，但是要到汇编清单完成以后才写入这个文件。A 命令中给出的名字，在这种情况下是没有用的。

4. 当打开输出文本文件或目的磁盘文件时，会显示出下列两种信息中的一种：

(1)FILE ALREADY EXISTS. USE IT??? （文件已经存在，是否用它），如果这正是需要用的文件，就用打入 Y 来回答。否则按 BREAK 键以终止 W 或 A 命令。

(2)＊＊＊＊＊FILE NON-EXISTENT. REPLY 'C' TO CREATE IT（文件不存在，是否要建立它)。如果这个文件就是你想建立的，则打入 C 作为回答。否则按 BREAK 键以终止 W 或 A 命令。

5. 由于在原来的 DOS 中有一个错误，为了使 BREAK 起正确的作用，所以 EDTASM 程序在中断关闭（不包括由键盘输入/输出重新启动的时候）状态工作。

6. 这个 EDTASM 程序可以在一个标准的 TRS-80 I 型的 TRSDOS 中执行。

7. 这个 EDTASM 程序使用标准的键盘，显示和打印子程序以及相应的控制块。用户在改动系统时要小心！

8. 用 EDTASM 程序得到的目的码可以用下面的几种方法来执行：

(1)在 DOS READY 状态（即处于系统命令级),可以直接打入目的码文件的文件标识符和 ENTER 键，NEWDOS/80 就会把目的码装入存储器，并以汇编源程序中伪指令 END 中

给出的入口地址（请参阅《微计算机实用手册》317页——海洋出版社出版）作为程序入口而加以执行。

（2）在DEBUG程序中，打入G命令就可以从汇编源程序中伪指令END后面给出的入口地址处执行这个目的码程序。或者由I和C命令单步执行这个目的码程序。

（3）可以在DOS READY状态用LOAD命令把目的码模块装入存储器，然后进入BASIC，使用USR (x) 功能调用这个目的码模块（请参阅《微计算机实用手册》中BASIC的有关章节）。

6.6 CHAINBLD 建立和修改链文件

CHAINBLD/BAS是一个简单的BASIC程序，用户可以用它来建立和修改链文件（有关链接的讨论请看4.3节）。

CHAINBLD程序按记录方式来工作（要求在文件中至少每240个字节中出现一个EOL字符——ENTER），而且CHAINBLD程序把每个遇到的EOL字符作为BASIC输入行的结束和链文件中记录的结束来处理。所有的插入、删除、更改、传送和复制均用记录来处理。

此外，CHAINBLD程序不能建立包含有特殊的不可打印的字符的文件（不包括老的1.0版本的16进制代码80H—83H）。对于字符的使用符合如下规则：如果由BASIC语句LINE-INPUT C$形成的字符串不能包括的字符，同样不能成为链文件的组成部分。例外情况是CHAINBLD程序自动提供的EOL字符。如果用户需要在他的链文件中包含有特殊字符，就必须使用其它的程序来建立链文件。作为一种最后的手段，就是SUPERZAP程序。

CHAINBLD程序开始有16秒钟的初始化时间，这时最大限度地把空间分配为字符串区。所以必须提醒用户，如果要用BREAK键来中断或终止CHAINBLD程序，必须牢记此时所有可用空间均已分配给了字符串区，同时还要知道，由于缺少空间，有些命令就无法正常工作。如果使用CLEAR命令去腾出一些空间，就一定要规定字符串区的大小。

CHAINBLD程序完成初始化以后，就会显示出CHAINBLD功能的主清单（请不要与编辑功能清单混淆）：

```
CHAIN FILE BUILD/EDIT PROGRAM
INPUT NUMBER FOR DESIRED FUNCTION
1. DELETE ALL TEXT LINES
2. LOAD EXISTING TEXT FROM DISK
3. SAVE TEXT TO DISK
4. EDIT TEXT
5. EXIT PROGRAM
?
```

操作者只要在显示的'？'号后面打入选中项对应的数字就可以了。

用户可以选用的功能项的含义如下：

1. DELETE ALL TEXT LINES （删除全部文本行）

删除字符串区中所有的文本行，并显示出编辑功能清单。在CHAINBLD程序开始执行时，字符串区内是没有文本行的。

2. LOAD EXISTING TEXT FROM DISK （把已有的链文件从磁盘装入字符

串区)。

用这个选用项就可以编辑已有的链文件。如果这时字符串区已经放有文本行，CHAINBLD 程序将会询问用户是否要删除这些文本行。如果不删除，CHAINBLD 程序认为用户还想对字符串区内的文本作进一步处理，于是 CHAINBLD 程序就返回到它的主清单。否则就把老的文本删除掉。然后 CHAINBLD 程序将询问已有的链文件的文件标识符。如果文件标识符不包括文件名扩展，就假定文件名扩展是 JCL。接着把这个文件装进字符串区。文件长度不能超出字符串区的容量，即不能多于 1000 行。链文件必须符合以前介绍过的格式，分段成记录。文件装入字符串区以后，CHAINBLD 程序就显示出编辑功能清单。

3. SAVE TEXT TO DISK （把处理完的链文件存入磁盘)。

用户完成了链文件文本的建立和（或）编辑，现在就可以把它写入磁盘了。如果链文件中没有文本行，CHAINBLD 程序将询问是否要写这个内容空缺的文件。如果不写，CHAINBLD 就要返回到它的主清单。

接着 CHAINBLD 程序要询问是否把这个文件写到盘上，写到盘上以后就可以用 NEWDOS/80 来处理了。如果要把文件写到盘上，就会把/·/0 到/·/3 的键控制记录用它们的相应单字节控制代码(80H—83H)来替换。而字符串区内的文本是不会改变的。

最后，CHAINBLD 程序要询问输出文件的文件标识符。如果文件标识符中没有给出文件名扩展，就用 JCL 作文件名扩展。接着把这个文件写到磁盘上。当完成写盘操作以后，CHAINBLD 程序就返回到它的主清单。

4. EDIT TEXT （编辑文本）

这个选用项只是显示编辑功能清单，不作其它的操作。显示的编辑功能清单如下：

```
EDIT COMMANDS. MM, NN AND TT ARE LINE NUMBERS. ;=1ST LINE.
/ =LAST LINE.      0 = BEFORE 1ST LINE, IF ANY.
L  MM             LIST LINES STARTING A LINE MM
;                 LIST NEXT DISPLAY PAGE
R  MM             REPLACE LINE MM
D  MM  NN         DELETE LINES MM THRU NN
I  MM             INSERT LINES AFTER LINE MM
X  MM             ADD TO LINE MM
C  MM  NN  TT     INSERT COPY OF LINES MM THRU NN AFTER LINE TT
M  MM  NN  TT     DELETE LINES MM THRU NN  AND  INSERT AFTER LINE TT
Q                 RETURN TO MAIN MEMU
U                 RETURN TO EDIT MENU
COMMAND?.
```

5. EXIT PROGRAM （退出 CHAINBLD 程序）

如果字符串区还有没有写到盘上去的文本，CHAINBLD 程序就要询问用户是否真想退出，如果不退出，CHAINBLD 程序就返回到它的主清单。否则，CHAINBLD 程序就要删掉所有的文本行，并释放所有的字符串区，只保留 50 字节的空间。然后按通常的方式结束这个程序。

当显示编辑功能清单时，用户可以选用的命令有：

1. L 和:命令

文本行按顺序隐含编号，与文本中发生的改变无关。行编号并不属于某一特定的文本行。所以插入一个行编号就表示当时文件中这一行的位置。这就是说插入、删除、复制和移

动等操作均要改变文本行的部分或全部的行编号。L 和：编辑命令可以显示文本行。L：显示第一行。L/ 显示最后一行。L52 显示第 52 行。在上述各种情况下，只要选中的显示行后面还有文本行，这些行同时也会显示出来。编辑命令？可以使文本向前换一页。

2. I 命令

这个编辑命令可以把一行或多行插入文本中指定行的后面。例如，I0 表示插入文本的开头。I/ 表示插入文本的末尾。I23 表示插入到第 23 行后面。把一些行插入文本中时，直至遇到包含有 /·/1 字符序列的行（但这一行不包括在插入行中）时为止。上述字符序列终止了行插入工作方式。

3. R 命令

这个编辑命令用于把一个新行替换原来的那一行。例如，R43 表示要用一个新行来替换文本的第 43 行，CHAINBLD 程序将会询问用户要放入什么样的新行。

4. D 命令

这个编辑命令的功能是删除一个文本行或多个文本行。例如 D43 表示要删除文本的第 43 行。D20 40 表示要删除文本的第 20 到 40 行。

5. X 命令

这个编辑命令的功能是把指定的文本行加到文本内。注意：CHAINBLD 程序实际上不能对一行进行编辑。编辑方式真正编辑的是整个文本。

6. C 命令

这个编辑命令可以把一些指定的行复制到文本的另一部分去。例如，C20 30 5 表示把第 20 行到第 30 行的副本插入到文本的第 5 行后面。请注意，原来的第 20 行到 30 行现在将变为 31 行到 41 行。

7. M 命令

这个编辑命令可以把指定的一些行移到文本的另一位置。例如 M20 30 5 可以删除文本的第 20 到 30 行，并且在文本的第 5 行之后重新插入第 20 到 30 行的内容。

8. U 命令

这个编辑命令可以重新显示编辑功能清单。

9. Q 命令

这个命令可以重新显示 CHAINBLD 程序的主清单。

学习 CHAINBLD 程序的最好方法就是使用它。在 NEWDOS/80 磁盘上有一个链文件的示范性程序 CHAINTST/JCL。用户可以把这个程序装入存储器，然后再看一下它的内容。放入字符串区后，你就可以按照自己的要求来修改它，但是千万别用 CHAINTST/JCL 的名字重新写到盘上，这样会破坏这个示范程序，你可以改换一个名字再写到盘上去。示范的链文件 CHAINTST/JCL 程序清单如下：

```
/./2THIS  MESSAGE  SHOULD  NOT  BE  PRINTED
/./3THIS  MESSAGE  SHOULD  BE  PRINTED
FREE
BASIC
CMD"S=DIR  0
/./1TEST  OF  PAUSE.  PRESS  ENTER  TO  CONTINUE
SUPERZAP
VDS
```

```
N
X
EXIT
BASIC  LOAD"CHAINBLD/BAS"
CMD"S=BASIC *
CMD"S
CHAIN,  CHAINTST/JCL:0,  SECTION2
/./OSECTION1
FREE
/./3TEST  ALL  DONE
/./OSECTION2
DIR  0
/./4SECTION1
```

6.7 ASPOOL 自动假脱机程序

1. ASPOOL程序的简要说明。

NEWDOS/80 盘片上的目的模块 ASPOOL 是经 Apparat 公司改编的、H.S.Gentry 编的自动Spooler Program（假脱机程序），使它可以与 NEWDOS/80 一起工作，并具有自浮动定位的能力。ASPOOL 程序能够自动把用户打印机输出直接送到磁盘上，然后自动送到打印机上去打印，只要主程序大约每秒钟发送一个假脱机的字节或者检查键盘是否有一个新的输入字符，就可以实现主程序在前台执行而假脱机程序在后台打印。

这个假脱机程序包含在 NEWDOS/80 盘片上，但是对 NEWDOS/80 而言，它又是一个独立的程序。它是 NEWDOS/80 中不能完全给予保证的部分。

NEWDOS/80 磁盘操作系统的基本操作认为凡是 DOS 发送到打印机的输出将不是通过磁盘 I/O 送到打印机的。所以假脱机程序要把从 DOS 来的（如 PRINT, JKL, 带有 P 参数的 DIR 等命令）打印机输出统统作废，并对每个假脱机文件显示一次警告信息 CANT SPOOL FROM DOS（不能由 DOS 进行假脱机操作）。

ASPOOL 程序不能对要打印的假脱机文件作多次打印；打印一次后就要把文件的EOF置成 0，并关闭文件以便让出文件的空间。从一个假脱机程序转入到下一项工作（包括复位操作）时这个假脱机程序不会记住假脱机的内容。要提醒用户注意：在假脱机程序有效期间，不能使用复位或 DOS 库命令 BOOT 返回 DOS READY 状态。如果没有其它办法，可以用同时按 D,F,G 键进入 MINI-DOS 状态，然后用 DOS 库命令 MDBORT 回到 DOS READY状态，也可以用同时按 1,2,3 键进入 DEBUG 程序，然后用 DEBUG 程序的 Q 命令回到 DOS READY 状态。

2. 初始设定。建立一个工作的假脱机模块。

使用假脱机系统以前，必须建立 ASPOOL 的工作程序模块副本。用户应该为自己想用的每种不同的配置建立相应的工作程序模块。在建立工作程序模块的时候，输入模块的文件标识符 1 必须是 ASPOOL/MAS 及其副本，而输出模块的文件标识符 2 决不能用 ASPOOL/MAS。为了建立一个工作的假脱机程序模块（相对原版的模块而言），打入 DOS 命令：文件标识符 1，I（例如：ASPOOL/MAS:0, I）。然后，该程序将向用户询问一些参数说明：

（1）该程序将询问用户是否要用打印机驱动程序去驱动打印机，在假脱机程序工作的时候，这个驱动程序的地址在 4026H—4027H 中。如果要用软件驱动程序，就回答 Y；如果不

用，则回答 N（这时将由假脱机程序去驱动打印机）。在回答 N 时，这个程序还将询问打印机是并行工作方式还是串行工作方式。如是并行工作方式，则回答 P；若是串行工作方式，则回答 S。如果是串行工作方式，则还将继续询问打印机是否为 H14 型，是 H14 型回答 Y，否则回答 N。

（2）该程序将询问打印输出是否要分页和页间留出空白页边。如果要分页，则回答Y,否则回答 N。如果回答 Y，接着就要询问用户每页多少行，可以打入 10—99 之间的一个数。

（3）该程序要问打印机用软方式还是硬方式换页。所谓软方式换页，就是对打印过的行计数，然后打印回车 (0DH)（可以带换行符——0AH，也可以不带换行符），一直到达该页的终止行为止。而硬方式换页是用单个控制字符实现换页功能。如果你的打印机能够识别硬方式换页的话，请回答 H，否则回答 S。如果是软方式换页，程序还将询问每页的总行数，可以回答 10—99 之间的一个数。

（4）该程序还将询问用户每个文件打完后是否要换页。如果要换页，就回答 Y；否则回答 N。

（5）下一个问题是每次回车时的自动换行问题。有些打印机在回车时就会换行，因此计算机就不应该再输出换行字符，如果你的打印机是这种型式的（即 Radio Shack 标准），则对这个问题可用 N 回答。如果需要用软件控制换行，则回答 Y。

（6）这个程序还要问用于假脱机打印数据的磁盘驱动器编号。可以根据实际情况用 0 到 3 之间的数来回答。

（7）这个程序还要询问在最后一个键盘字符输入以后，直到假脱机程序能够重新开始打印时为止，中间要隔多少秒钟。对这个问题可以用 00—59 之间的一个数来回答。在两者之间要有一定时间间隔的原因是：为了要使键盘工作优先于打印机。当按动键盘上的一个键时，如果假脱机程序正在打印一个文件，打印机将在键盘输入期间暂停工作，一直要到打入最后一个键后，达到设定的延迟秒数才恢复打印机的工作。

（8）这个程序还要询问是用定时中断（在 TRS-80 I 型中每 25 毫秒一次，在Ⅲ型中每33或40 毫秒一次）还是经键盘输入和假脱机输出去驱动打印机。如果要用中断，就回答 Y，否则回答 N。在前台正在执行不是通常的检查键盘或发送输出到假脱机程序的任务期间，允许使用中断去启动假脱机程序进行打印工作。对于有缓冲区的打印机，使用中断的缺点是在输出一行到打印机的整个过程，中断是要被禁止的。然而，与磁盘 I/O 操作相比，时间上的延迟不一定不合算，因为磁盘操作更费时间。如果用了中断，前台程序一点也不发送信息到假脱机程序，也不检查键盘的输入，则打印工作仍将停止下来。这是因为，只有在进行键盘检查或主程序输出到假脱机程序时，磁盘 I/O 才能去读下一个扇区。使用中断还有另外一些缺点，请看后面的讨论。

（9）该程序还要询问是否要用循环缓冲器去作键盘的输入字符的缓冲器。如果要用缓冲器就回答 Y，否则回答 N。循环缓冲器可以防止丢失键盘的输入字符。如果启动 25 毫秒的中断去驱动打印机（请看（8）的讨论），循环缓冲器就要使用 ROM 内的键盘输入子程序，所以在假脱机程序工作以前，要禁止任何工作着的驱动程序（像 NEWDOS/80 的键盘截取子程序、小写字母驱动程序等）。如果不使用25毫秒中断把假脱机的输出发送到打印机，就要使用常规的键盘子程序（在假脱机程序工作时，它由 4016H—4017H 的向量给出）。如果不用循环缓冲区，就不管是否用 25 毫秒的中断，总保持着常规的键盘子程序。

回答了上述所有问题后，假脱机程序就有了应有的预置参数，这个经用户按自己的要求配置好的假脱机程序（即假脱机工作程序模块）就代替了存储器中原有的假脱机程序。然后，ASPOOL程序就要询问用户：保存到盘上的这个工作程序模块的文件标识符。这时你就可以回答文件标识符2,A。千万不能回答ASPOOL/MAS!!!!!! 接着，工作程序模块将被写到磁盘上，这个假脱机程序可以经402DH返回到DOS状态。

3. 启动假脱机程序

要使用假脱机程序时，打入DOS命令"文件标识符2,A"（例如：SPOOLER, A),其中文件标识符2就是你已经建立的假脱机工作程序模块中的一个。文件标识符2决不能是ASPOOL/MAS。如果假脱机程序已经启动，就会在荧光屏上显示：FILE ALREADY EXISTS（文件已经存在）信息。

这个工作模块将装入5200H—5FFFH区间，其自身可以浮动到HIMEM－ARSIZE1+1，并设定新的HIMEM=HIMEM－ARSIZE1,其中HIMEM是DOS最高存储器地址（在TRS-80Ⅰ型中这个地址放在4049H—404AH单元内，在TRS-80Ⅲ型中放在4411H—4412 H单元内),而ASRIZE1是假脱机程序所需要的存储单元总数。然后，为了转向假脱机程序，要截断放在4016H—4017H单元的键盘向量和放在4026H—4027H单元的打印机向量。如果要用中断，就要把一个子程序放进NEWDOS/80 25毫秒用户中断子程序的中断链中。假脱机程序开始工作就会显示出：SPOOLER ACTIVE（假脱机程序已工作）信息，由402DH出口可返回到DOS状态。

现在假脱机程序可以工作了。所有要送到打印机的数据将直接送入5个磁盘文件(POOL1，POOL2, POOL3, POOL4, POOL5) 中的一个。你或许会问为什么要5个文件？情况是这样的，当你想"打印"很多数据，并且发一个文件结束符到ASPOOL程序，使那些数据就像在一个真的打印机上打印一样，就必须建立较多的文件。可以经过DOS命令*ASP, W(在BASIC中可以用CMD "*ASP, W"),或者在通常的打印字符系列中向假脱机程序送一个03H字节（在BASIC中用LPRINT CHR$(3)),这时，将关闭正在进行假脱机处理的文件并把它列入打印处理。现在你可以把刚才"正在打印"的还有一些数据假脱机处理到另一个文件。当正在打印第一个数据文件的时候，这些数据被放到磁盘上。这个过程可以重复做5次（因为预先只设立了5个磁盘文件POOL1—POOL5)。如果第一个文件在真正的打印机上打印完以前,你试图假脱机处理到第六个文件,系统将显示：SPOOL FULL. WAITING ON PRINTER（假脱机已满，请等待打印机打完）信息，并将使系统挂起来，直到这个文件打完为止。所有的数据都是在后台的情况下、在真正的打印机上打印的，这时或者正在执行着主程序、或者系统正在等待用户发命令。不论何时，只要执行*ASP, W或者在输出到假脱机程序的数据中发现一个03H字节，假脱机程序就认为这是文件的结束（如果指定相应参数，就会执行换页到顶行的操作),虽然这不一定真是文件的结束，可能只是你想把一篇报告的假脱机输出分分段落，要打印机往前走一点，也可能是为了防止打出页外。

注意！通常由假脱机程序使用的TRS-80Ⅲ型ROM子程序，当它发现打印机没有准备就绪（包括打印机处于忙态）以及按了BREAK键，将会丢失正在发送到打印机去的当前的字符。假脱机正在后台打印的时候，有可能前台程序正在使用BREAK键，这样丢失了打印字符而假脱机程序并不知道，这种情况是可能发生的。这也就是TRS-80Ⅲ型不用假脱机程序、而只使用ROM内的打印机驱动子程序的原因。

你可以使假脱机程序在任何时候从容不迫地停下来，使用的是 DOS 命令 *ASP, S（在 BASIC 中可以用 CMD "*ASP, S"），或者通常往假脱机发送数据时发送一个 04H 字节（在 BASIC 中可以用 LPRINT CHR $ (4))。这样将会清除当前的假脱机文件，并阻止正在建立的新文件，以及显示：SPOOL STOPPING（假脱机作业已停止）信息。然后继续执行主程序，同时忽略发送到假脱机程序的任何字符，而假脱机继续打印已经排定的文件。当所有的文件打印完毕时，就执行 *ASP, P 命令的功能。注意，如果发现假脱机程序挂起来了，这或许是在等待主程序去检查键盘。如果主程序不能完成检查键盘这一工作，你可以按 D,F, G 联键去试一下，但是一直要等到驱动程序停止时为止。

你可以使假脱机程序在任何时候突然停下来，使用的是 DOS 命令 *ASP, P（在 BASIC 中用 CMD "*ASP, P"）。所有剩下的假脱机数据都要丢失。如果中断子程序有效，也要被撤销。键盘和打印机向量要被恢复到假脱机程序启动时原有的值。如果 DOS 的 HIMEM 值与假脱机程序启动时由假脱机程序设定的值相同，则 HIMEM 值就要置回到使假脱机程序有效以前它原有的值，这样就收回了假脱机程序占用的存储器。显示 SPOOLER PURGED（假脱机程序已被清除）信息，并由 402DH 返回到 DOS 状态。

你也可以在任何时候用 DOS 命令 *ASP, C（在 BASIC 中用 CMD "*ASP, C"）去清除打印队列。假脱机将回答 CLEAR BACKLOG OR PRINT (B/P)? （要清除储备文件还是当前打印文件 (B/P)?）信息。如果你想清除队列内积压的储备文件，可以打入 B 和ENTER，如果只要停止对当前打印文件的打印，则打入 P 和 ENTER。清除储备文件不会去除当前打印的文件，同样，清除当前打印的文件，也不去除储备文件。

在任何时候只要打入 DOS 命令 *ASP（在 BASIC 中用 CMD "*ASP"）就可以检查假脱机系统的状态。系统将打印一分所有等待打印的文件（储备文件）和因为正在打印或假脱机处理而打开着的文件的清单。如果假脱机系统已经停止工作，但尚未撤销，将会显示 SPOOL STOPPING（假脱机作业已停止）信息。如果假脱机程序已经撤销或者尚未启动，将会显示 FILE NOT IN DIRECTORY（文件不在目录中）信息。

第七章　磁盘BASIC（无I/O扩充）

7.1 前言

众所周知，TRSDOS 磁盘BASIC 是 ROM BASIC（即 LEVEL Ⅱ BASIC）的扩充，这主要有以下几个方面。

1．增加了必要的磁盘文件操作的命令

KILL　　删除一个磁盘文件。
LOAD　　装入一个磁盘文件。
MERGE　　合并 ASCII 格式的 BASIC 程序。
RUN "filespec"　装入并执行 BASIC 程序：filespec。
SAVE　　把 BASIC 程序存入磁盘。

2．增加了必要的文件 I/O 语句和函数

OPEN　　打开一个文件。
CLOSE　　关闭已打开的文件。
INPUT#　　以顺序方式读磁盘文件。
LINE INPUT#　以顺序方式由磁盘文件读入一行数据。
PRINT#　　以顺序方式写入文件。
GET　　以随机方式读文件。
PUT　　以随机方式写入文件。
FIELD　　规定随机存取缓冲字段。
LSET　　把字符形式的值放入字段的左边。
RSET　　把字符形式的值放入字段的右边。
MKD$　　把双精度数值转换成字符形式的值。
MKS$　　把单精度数值转换成字符形式的值。
MKI$　　把整数值转换成字符形式的值。
CVD　　把文件中读出的双精度值恢复成数字形式的值。
CVS　　把文件中读出的单精度值恢复成数字形式的值。
CVI　　把文件中读出的整数值恢复成数字形式的值。
EOF　　检测文件结束标志。
LOF　　获得文件中的最大记录号。

3．扩充 ROM BASIC 的功能（与磁盘文件 I/O 无关）

&H　　允许使用 &H 为前缀的十六进制常数值。
&O　　允许使用 &O 为前缀的八进制常数值。
LINE INPUT　由键盘输入一行数据。
INSTR　　在字符串内检索一个子字符串。

MID$= 替换字符串内的一部分。

TIME$ 实时时钟。

DEF FN 用户自定义函数。

DEF USRn= 定义机器语言子程序的入口地址。

USRn 可调用十个机器语言子程序。

CMD"S" 返回操作系统。

CMD"D" 启动查错程序 DEBUG。

CMD "T" 关中断（实时时钟）。

CMD"R" 开中断（实时时钟）。

而 NEWDOS/80 磁盘 BASIC 又对 TRSDOS 磁盘 BASIC 作了进一步的扩充，但是，NEWDOS/80 磁盘 BASIC 的多数性能仍然与 TRSDOS BASIC 保持相同，例如上述对 ROM BASIC 扩充部分中的 KILL, SAVE, INPUT#, LINE INPUT#, PRINT#, FIELD, LSET, RSET, MKD$, MKS$, MKI$, CVD, CVS, CVI, EOF, LOF, &H, &O, LINE INPUT, INSTR, MID$=, TIME$, DEF FN, USRn, DEF USRn=等是完全一样的，其他的语句和函数仍保持它们在 TRSDOS BASIC 中的基本功能并又有新的扩充，另外还增加了一些TRS-DOS BASIC 所没有的新功能。这些扩充主要有：

1. 改变了启动 BASIC 的命令格式并扩充了命令的功能。

2. 增加了一些新命令：

用于显示和编辑的直接命令↓, ↑, /, ;, :, @, · 等；

文本行复制命令 DI, DU；

重编行号命令 RENUM；

用于检索文本中的行号、整数、变量名、字符串的命令 REF；

恢复被 NEW 命令清除的文本的命令 RENEW。

3. 扩充原有命令的功能。主要有 MERGE, RUN, LOAD, CMD 等，特别是增加了很多非常有用的 CMD 命令，如 CMD "doscmd"（执行 DOS 命令），CMD "F", DELETE（动态删除文本），CMD "F=ss"（单步执行程序），CMD "F=SWAP"（交换变量内容）、CMD "O"（数组排序）等。

4. 增加了磁盘文件的类型并相应改变其存取（I/O 处理）语句和方法。

在 NEWDOS/80 BASIC 中，现在有四种文件类型，除了原有的 TRSDOS BASIC 的顺序文件（NEWDOS/80 中称输出/输入文件）和随机文件（NEWDOS/80 中称字段项文件）以外，又增加了固定项文件（又分两个子型：FI, FF）和标记项文件（又分三个子型：MI, MF, MU）。

文件的存取方式现在有五种，除原有的方式 I（顺序输入），O（顺序输出），R（随机存取）以外，又增加了方式 E（顺序输出并可超过 EOF）和 D（随机存取但不可超过 EOF）。

所有的文件类型，除输出/输入文件只能顺序存取外，其他类型既可随机存取还可顺序存取。为此，OPEN, GET, PUT 等语句的格式和功能除保留原有的以外，又增加了新的内容，如用于规定文件类型的 ft，用于规定逻辑记录长度的 lrecl，用于确定文件位置的 fp，以及用于与文件交换数据的项组表达式表 igel 或项组表达式表语句行号 igelsn 等。它们现在的格式如下所示：

OPEN m, fan filespec 1 [,ft] [,lrecl]

GET fan [,fp] [,,igel/,igelsn]

PUT fan [,fp] [,,igel/,igelsn]

而且 GET 和 PUT 语句还扩充成可进行顺序存取操作（但不能用于输出/输入文件）。

另外，LOC 函数的功能也扩充成不但能获得刚存取的记录号，还能获得一组文件项、记录和 EOF 的位置，以及判别当前文件位置是否位于 EOF。

本章及下章主要介绍 NEWDOS/80 BASIC 中这些对 TRSDOS BASIC 原有功能和语句的新的扩充和两者的差异，以及 NEWDOS/80 BASIC 独有的新功能，而对 ROM BASIC 和 TRSDOS BASIC 中与 NEWDOS/80 BASIC 相同的部分将不准备涉及。所以，我们希望读者在使用 NEWDOS/80 BASIC 以前，一定要熟悉 ROM BASIC 和 TRSDOS BASIC，并持有这两种 BASIC 的手册(请参考计声编译，《微计算机实用手册》，海洋出版社，1982 年 7 月)。本章介绍磁盘 BASIC 中与磁盘文件处理无关的部分，第八章介绍与磁盘文件处理有关的部分。

7.2 几点说明

以下几点说明与具体的磁盘 BASIC 命令或语句无关，但它们贯穿于整个 BASIC 中，必须单独给予说明。

1. 当发生 BASIC 句法错误时，在 TRSDOS 中，BASIC 将自动进入编辑该错误行的状态（显示出该行行号，等待用户给予编辑命令)，但在 NEWDOS/80 中，BASIC 不在有错误的该文本行上自动进入编辑状态，而把该行置成当前行。如果用户希望编辑这一行，只要按下逗号键，就可使这一行进入编辑状态。这一改变可使用户不会在无意中清除那些有助于他在查错中想要查看的变量。

2. 要注意 BREAK 键的作用在 NEWDOS/80 BASIC 中不是总是有效的，因为 BASIC 程序可以通过 CMD "BREAK, N" 使 BREAK 键失去作用，并用 CMD "BREAK, Y" 使它重新起作用。

3. 因为 CLOAD 进行磁带输入时要在相邻的字节之间执行 NEW 操作，如果 BASIC正在使用三个以上的文件区，则 CLOAD 操作将失去同步。

4. 在 NEWDOS/80 中，当 BASIC 遇到一个 DOS 错误时，若没有 ON-ERROR 程序的作用，则 DOS 错误信息和 BASIC 错误信息就都被显示出来。

5. BASIC 现在总共使用 8 个覆盖模块。用户将会看到，在执行 RUN 的任何时候，以及中断（因程序中的 STOP 语句、错误或按下 BREAK 键）或者结束（因 END 语句）运行的任何时候，都会发生磁盘 I/O，这样做是为了引入当前执行的操作或准备执行的下一个操作所需要的 BASIC 子程序。

6. NEWDOS/80 磁盘 BASIC 不允许象 TRSDOS 磁盘 BASIC 那样使用打入行号的方法来删除文本行，而必须使用明确的删除命令 DELETE 或 D 来删除文本行。

7.3 磁盘 BASIC 的启动

在 TRSDOS 下，启动磁盘 BASIC 的方法是打入 DOS 命令BASIC，于是在启动 BASIC的同时，会询问以下两个问题，要求用户给于响应：

HOW MANY FILES?

这是系统询问用户准备使用的文件缓冲区数目（0—15）。在用户给予回答后，又问：

MEMORY SIZE?

这是询用 BASIC 可用内存储器的最高地址。只有在回答这两个问题以后，才进入 BASIC READY 状态。

而在 NEWDOS/80 下，如果象以上那样只使用 DOS 命令 BASIC 来启动磁盘 BASIC，则不会询问以上问题，而采用它们的缺项值（下面讲述），否则应在 BASIC 命令中明确指明这两个值。在 NEWDOS/80 中，键入以下 DOS 命令之一都可启动磁盘 BASIC:

1. BASIC
2. BASIC *
3. BASIC n
4. BASIC m
5. BASIC cmd
6. BASIC n, m, cmd
7. BASIC m, n, cmd
8. BASIC n, m
9. BASIC m, n
10. BASIC n, cmd
11. BASIC m, cmd

其中：

* 意味着用户要 BASIC 使用内存储器中已经存在的 m和 n 的值，在文本缓冲区中重新建立程序。也就是说，BASIC *命令允许用户在复位或执行 CMD "S" 命令回到操作系统后，再用上次启动 BASIC 时使用的 m,n 值重新启动 BASIC，并恢复内存中原先存在的程序。如果 BASI 能够完成这项工作，那末它强令 LIST 作为它的第一条命令，列出程序清单。如果 BASIC 不能恢复这个程序，则返回到DOS READY。如果 n 小于 2 或者内存储器中的程序少于 3 行，那末命令 BASIC *就不能工作。

n 它是分配给 BASIC 的文件区数目，也就是 TRSDOS 中启动 BASIC 时对 HOW MANY FILES? 问题的回答。如果在命令中不提供 n 值，则它的缺项值为 3 。n 的取值范围是 0—15，这与 TRSDOS 是相同的。n 这个数也是 BASIC 中进行文件输入输出的任何语句所使用的最大文件区号。如果 BASIC 程序使用非标准记录长度（标准记录长度为 256 字节）的字段项文件，那末必须指定 n，并且必须用字母 V 作后缀。如果 n 后带有字母 V，则给每个文件区再额外分配 256 字节的缓冲区（见下面的例 4 ）。

m m 是内存大小，即 TRSDOS 中启动 BASIC 时对 MEMORY SIZE? 问题的回答。m 值减 1 是允许 BASIC 使用的最高内存单元。若不指定 m 值，则使用当前的 DOS HIMEM（4049H—404AH）的值，而在 TRSDOS 中，若不回答系统询问的这个问题直接按回车键，则要使用到系统的实际的最高内存储器单元。m 以及 m 以上的所有内存储器是用户保留的存储空间，BASIC 不能使用，但用户可以用于其他程序，如打印机驱动程序、专门编写的经 USR 调用的机器语言子程序等等。

cmd 这是一行 BASIC 文本，它由一个或多个 BASIC 语句（语句之间用冒号分隔）组

成。这个 BASIC 文本行就被看作直接键盘输入，并且在初始化完成之后立即执行。

请记住，启动 BASIC 的 DOS 命令被限制为最多 80 个字符（包括 [ENTER] 在内）。如果启动 BASIC 的 DOS 命令是通过系统 AUTO 功能调用的，则进一步被限制为 32 个字符，也包括 [ENTER] 在内。

在初始化过程中遇到任何错误时，就返回到 DOS。

如果 DOS 处于 RUN-ONLY 状态，则启动 BASIC 的 DOS 命令中必须含有 RUN 或 LOAD（带有选用参数R）语句。

举例

1. BASIC

启动BASIC，但它与TRSDOS中不同，它不询问有关文件区数目和最高内存地址的问题，而采用它们的缺项值，即它有三个文件区，高端内存设置为 DOS 中 HIMEM 的当前值。BASIC 启动后，显示“READY”，等待用户的命令。

2. BASIC, RUN "XXX/BAS"

启动 BASIC，它有三个文件区，高端内存设置为当前的 DOS HIMEM 值，并将 BASIC 程序 XXX/BAS 装入文本区域，然后开始执行此 BASIC 程序。

3. BASIC, 9, 48152, LOAD "XXX/BAS"

启动 BASIC，它有 9 个文件区，高端内存设置为 48151（比指定的 48152 小 1），将 BASIC 程序 XXX/BAS 装入文本区域，然后显示“READY”，等待用户的命令。

4. BASIC, 3V

其工作与上述例 1 相同，不同的是给三个文件区的每一个分配了额外的 256 字节的缓冲区。如果程序使用记录长度不是 256 的字段项文件，那末就需要这个额外的缓冲区。

5. BASIC, CLEAR3000:A=1:RUN "XXX", V

启动 BASIC，它有三个文件区，高端内存设置为 DOS 的当前 HIMEM 值，执行CLEAR 命令保留 3000 个字节作为字符串区域，令数字变量 A 等于 1，装入 BASIC 程序 XXX，并且在开始执行 BASIC 程序时不把变量清零，因此，若在此时检查程序中的变量 A 时，就可以看到变量 A 保持不变。

7.4 直接的显示/编辑命令

NEWDOS/80 磁盘 BASIC 允许使用以下“直接”命令进行文本行的显示或编辑。这些命令是 NEWDOS/80 BASIC 独有的，在 TRSDOS BASIC 中没有。它们的使用规则是，每一个直接语句行只允许有一个这样的命令，并且此命令必须是输入行的第一个字符（不允许有行号或退格键）。这些直接命令及其功能如下：

• **（句点）** 显示当前文本行。

↓**（向下箭头）** 显示下一文本行。若不存在下一文本行，就与执行下面的/键一样，显示最后一行文本。

↑ **（向上箭头）** 显示当前行的上一行文本。若不存在上一文本行，就与执行下面的；号键一样，显示第一行文本。

；或 SHIFT ↑　显示第一行文本。

/ 或 SHIFT ↓　显示文本中的最后一行。如果TRS-80 I型机内装的是新的ROM（请参阅梁祖威等译，«TRS-80微计算机驻机解释程序——LEVEL Ⅱ ROM剖析»，海洋出版社，1984年），用户将发现SHIFT ↓不再是一个有效的键，因此应使用/键。

：（冒号）　显示内容向文本的开始方向卷动一页，即显示上一页文本。于是，以前的当前文本行现在已移到显示器的底下消失了（除非前面的命令是；号，这时只显示第一行），而原先显示屏上的第一行现在成为新的当前文本行出现在显示屏的底行上。

@　显示内容向文本的结束方向卷动一页，即显示下一页文本。当卷动完成以后，原先的当前文本行（即显示屏上的底行）现在处于显示屏的第一行上，新的当前文本行是新显示的最末一行。如果前一个命令是/，则只显示最后一行文本。

，（逗号）　编辑当前文本行，即它的作用相当于EDIT. 命令。

7.5 命令的简写

NEWDOS/80磁盘BASIC允许将命令AUTO, DELETE, EDIT和LIST分别简写成A、D、E和L这样的命令形式，但必须在满足以下条件时才能采用简写：

1. 简写命令是输入行的第一个字符。
2. 简写命令的后边必须跟有句点或十进制数字的行号。
3. 行号的范围符号（即－号）不能直接跟在简写命令后边，但可以跟在句点和十进制数字的后面。

例如：

A. 自动编写行号，增量为10，以当前行行号为首行。与AUTO. 相同。

A100，10　自动编写行号，以100为首行，增量为10。与AUTO 100，10相同。

D. 删除当前行。与DELETE. 相同。

D10－100　删除行10－100之间的所有文本行。与DELETE10-100相同。

L.－　与LIST. －相同。显示当前行至最后一行之间的所有文本行。

E. 与EDIT. 相同。编辑当前行。也与直接命令，相同。

E100　编辑行号为100的文本行。

7.6 DI和DU　文本编辑命令

DI和DU是NEWDOS/80 BASIC中新增加的两个BASIC文本编辑功能，它们采用以下格式的直接命令来实现：

1. DI　aaaa, bbbb
2. DI　., bbbb
3. DU aaaa, bbbb
4. DU .,bbbb

其中

aaaa　是被移动的或被复制的文本行的行号。

bbbb　是给予移动后的文本行或复制成的文本行的新行号。bbbb也允许是已经存在的

文本行的行号，这时原文本行内容被新文本行内容替换。

DI　该命令的功能是把文本行 aaaa 移动并插入到 bbbb 处，成为行号为 bbbb 的新文本行，同时删除原先的 aaaa 文本行。

DU　该命令的功能是在 bbbb 处插入一个与 aaaa 处完全相同的文本行，但 DU 命令不删除 aaaa 处的文本行，这是它与 DI 命令的差别。

这两个命令虽然可以在 bbbb 处复制一个行号 aaaa 的文本行，而且 DI 命令似乎与 RENUM 命令相似（见 7.9 节），但是它们不能把文本行中引用行号 aaaa 的地方改变成引用行号 bbbb（见本节举例），若需要这样做的话，那末要用 RENUM 命令去移动该文本行。

在 aaaa 的地方使用句点时，就使用 aaaa 的缺项值，即上一次列出的或编辑的或删除的那一行行号——当前行行号。

举例

假设文本区中有以下 BASIC 程序行：

```
10  INPUT"A=";A:IF  A<>0 ELSE10
20  PRINT"A=";  A
30  PRINT"A*A=";A*A
```

1. 若执行命令 DI 10, 25, 则文本区中的程序为：

```
20  PRINT"A=";  A
25  INPUT"A=";  A:IF  A<>0 ELSE10
30  PRINT"A*A=";  A*A
```

2. 若执行命令 DU 10,25，则文本区中的程序为：

```
10  INPUT"A=";  A:IF  A<>0  ELSE10
20  PRINT"A=";  A
25  INPUT "A=";  A:IF  A<>0  ELSE10
30  PRINT "A*A=";  A*A
```

读者可以看到，这两个命令执行后，都在行号 25 处复制一行原先为行号 10 的文本行，但是文本行中引用行号 10 的地方（即 ELSE 10）并不改变成 25，而且 DU 命令不删除原先的行 10。

7.7　RUN 和 LOAD　运行和装入程序

在 TRSDOS BASIC 中，命令 RUN 的格式为

RUN "filespec 1" [, R]

其功能是清除文本区，由磁盘上装入文件标识符为 filespec 1 的 BASIC 程序，关闭已打开的文件，把所有变量清零，然后执行此 BASIC 程序。若带有选用参数 R，则不关闭已打开的文件。

命令 LOAD 的格式为

LOAD "filespec 1" [, R]

其功能是清除文本区，把文件标识符为 filespec 1 的 BASIC 程序由磁盘装入内存，关闭已打开的文件，清零所有的变量。如果带有选用参数R，则其功能与上述 RUN "filespec 1", R

命令一样。

在 NEWDOS/80 BASIC 中，仍然保持以上的 RUN 和 LOAD 命令的格式和功能不变，但增加了一个选用参数 V，这个参数可使所有的变量在程序开始执行时不被清零。也就是说，RUN 和 LOAD 现在可以用以下格式使用选择项 V 来随意地保留所有的变量和已打开的文件区域：

RUN "filespec 1", V

LOAD "filespec 1", V

其中，filespec 1 是待执行程序文件的文件标识符。

带有选用参数 V 的 RUN 命令在执行 RUN 的过程中，除了 DEFFN 变量（即用户自定义函数）外，将保存所有的变量，因此，在 RUN 语句前已存在的变量可以在 RUN 语句后继续使用其原有的值。并且，在 RUN 以前打开的任何文件区仍然保持打开，以便在 RUN 语句后继续使用。

带有选用参数 V 的 LOAD 命令在执行时完全与带有选用参数 V 的 RUN 命令一样。由此，读者不难看出，使用带有选用参数 V 的这两个语句，对于程序的链接执行特别有用，它们可以在程序执行中链接另一个程序，并把其中的变量值通过变量本身传送给另一个程序和继续使用原先打开的文件进行输入/输出操作。

若在命令中指定了选用参数 V，则可以不指定选用参数 R。

举例

1. RUN "XXX/BAS" 清除文本区，装入程序 XXX/BAS，关闭已打开的文件，清零变量，然后执行该程序。

2. LOAD "XXX/BAS" 除不执行程序以外，其他与例 1 相同。

3.
```
10 PRINT "LINK PROGRAM XXX/BAS"
20 RUN "XXX/BAS", R
```
RUN 在此作语句使用，功能与例 1 相同，但不关闭已打开的文件。

4.
```
10 PRINT "LINK PROGRAM XXX/BAS"
20 LOAD "XXX/BAS", R
```
与例 3 完全相同。

5. RUN "XXX/BAS", V 除不清零变量以外，与例 3 相同。

6. LOAD "XXX/BAS", V 与例 5 相同。

7.
```
10 PRINT "LINK PROGRAM XXX/BAS"
20 RUN "XXX/BAS", V
```
在此作语句使用，功能同例 5 。

8.
```
10 PRINT "LINK PROGRAM XXX/BAS"
20 LOAD "XXX/BAS", V
```
与例 7 完全相同。

7.8 MERGE 覆盖程序的动态装入

NEWDOS/80 BASIC中，MERGE 的功能已经被扩充。它的格式仍然与 TRSDOS BASIC 中相同：

MERGE "filespec 1"

其中，filespec 1 是文本文件的标识符。

在 TRSDOS BASIC 中，MERGE 的功能是把指定的ASCII 格式的文本文件 filespec 1 装入内存，与原先驻留在内存中的程序按行号顺序合并在一起，如果新旧程序中有相同的行号，则在该行上以新的文本行代替旧的文本行。另外，MERGE 还关闭全部已打开的文件，把所有变量清零，并在命令完成后返回到 BASIC 的命令状态。

现在，MERGE 除保留以上基本功能外，还有以下两个方面的扩充。首先，MERGE 既可以连接合并 ASCII 格式的文本文件，还可以连接合并压缩格式的文本文件。其次，MERGE 不但可以作为一个直接命令使用，还可以作为程序语句来执行。

如果把 MERGE 作为程序语句使用，那末 MERGE 语句决不能是 DEFFN 语句、或子程序、或 FOR-NEXT 循环（由于隐含执行 POPS 操作）等的一部分，而且必须是文本行的最后一个语句，其下面必须跟随一个在 MERGE 以后继续执行的文本行，否则不执行 MERGE 功能，并给出 "SYNTAX ERROR"（句法错误）信息。同时，连接合并的文件不能含有行号与开始执行 MERGE 时已存在的文本行行号完全相同的文本行（在执行MERGE前，可使用 CMD "F", DELETE 把冲突的文本行删除）。MERGE 保护所有的变量。一旦进行实际的连接合并时，用户必须保证有足够的内存空间可用，因为如果连接合并失败的话，其错误造成的损失是不可挽回的。

例如：

```
.....
100  MERGE "XXX/BAS"
110  X=1
.....
```

其中，行 110 是必须有的在 MERGE 完成以后由此开始继续执行的文本行。

7.9 RENUM 重编 BASIC 程序的行号

这个 NEWDOS/80 BASIC 命令的功能是把当前的 BASIC 程序重编行号。凡是当前驻留在文本区中的 BASIC 程序或它的一部分都可以重编行号。RENUM 命令的格式有：

```
RENUM sssss, iiiii, ppppp, qqqqq [,U] [,X]
RENUM  ,
RENUM  U
RENUM  X
RENUM  U,X
```

上述格式中的第一行是基本的重编行号命令，它使得所有行号大于等于 ppppp 而小于等于qqqqq 的文本行都分配到新行号：sssss 是赋给的第一个新行号，其后的行号由上一个文本行的行号加增量 iiiii 得到。其中，sssss 和 iiiii 都必须在 1—65529 之间，它们的缺项值为 10。ppppp 必须在 1—65529 范围内，其缺项值是当前驻留在文本区中的 BASIC 程序的最小行号；qqqqq 也必须在 1—65529 范围内，并且必须大于等于 ppppp，其缺项值为当前驻留在文本区中BASIC程序的最大行号。用户一定要注意，新生成的行号范围不能包含重编行号范围ppppp—qqqqq 以外的任何已有的旧的文本行。只要遵循这个规则，那么新生成的行号范围

可以放在文本中的任何地方，同时将重编行号的文本移动到相应的新文本位置。因此，用户可以通过 RENUM 命令适当选择新行号值将部分程序移到程序中别的地方，同时将所有引用到的被移动行行号改变成新的行号。

在RENUM命令中，至少必须指定一个参数。如果用户想要指定全部缺项值并且不指定选用参数 X 和 U，那末使用一个逗号作为唯一的参数。见本节例 2 。

对于命令中的四个参数 sssss, iiiii, ppppp, qqqqq, 如果其中的一个或几个使用缺项值，那末必须在相应的位置放一个逗号，表示四个参数中的哪一个是用缺项值给出行号的。见下面的例 4 。

如果指定选用参数 U, 则 RENUM 实际上不进行重编行号的工作，并且不以任何方式改变文本，RENUM 仅仅搜索文本，进行文本错误的检查，找出没有定义的行号、部分句法错误以及一些与使用行号的 BASIC 语句有关的错误，并且把这些错误以下列格式显示出来：

sssss/U　　不存在文本行 sssss。

sssss/X　　文本行 sssss 有句法错误。

sssss/S　　文本行 sssss 有不正确的行号。

如果指定了选用参数X，则 RENUM 对于程序中引用的不存在的行号，将不作为一个错误显示出来，如果此行号在 ppppp 到 qqqqq 范围以外的话，也是如此。由于它允许程序引用那些不是程序中一部分的文本行，因此选择项 X 对于使用相互引用文本行的基本程序和覆盖程序的用户来说无疑大有帮助。

如果 RENUM 在改变文本以前遇到了任何错误，此命令的功能就象执行指定了选用参数 U 的命令那样，并显示出它所能找到的一切错误。如果在开始修改文本以后遇到一个错误，则显示出 “FATAL ERROR. TEXT NOW BAD” (致命错误。文本有错) 信息，并由 4030H 退出到 DOS，这个 BASIC 程序就不能再恢复了 (不要使用 BASIC *)。

如果在执行 RENUM 时，SYS11/SYS 和 SYS14/SYS 都不在系统中，那末系统将退出到 DOS READY 状态 (见 5.5 节)。

RENUM 将拒绝给第一个文本行行号为 0 的程序重编行号。请在执行 RENUM 之前，先用 DI 命令给此行分配一个不等于 0 的行号。

举例：

1. RENUM U

检查 BASIC 文本，寻找在实际重编行号中将会遇到的不确定的行号等错误。BASIC 文本不变。

2. RENUM ,

使用全部缺项值将整个 BASIC 文本重新编行号。它分配给第一个文本行的行号是 10，增量是 10，于是第二个行号是 20，……依次类推。

3. RENUM 100, 100

使用增量 100 重编整个 BASIC 文本的行号，分配给第一个文本行的行号是100，其后是 200, 300,…依次类推。

4. RENUM 2050,, 2050, 3160

将行号在 2050 到 3160 之间 (包括 2050 和 3160) 的所有文本行重编行号，使用的增量是 10，分配给第一个文本行的行号是 2050，第二个行号是 2060,……依次类推。

5. RENUM 30000, 5, 15365, 18112

将行号在15365到18112之间（包括15365和18112两者）的所有文本行重编行号。使用的增量是5，分配给第一个文本行的行号是30000，第二个行号是30005，…… 依次类推。重编行号后的文本行被移到文本中由30000开始的新位置。

7.10 REF 列出对变量、行号和关键字的访问表

BASIC语句REF是NEWDOS/80 BASIC中新增加的一个语句，它的功能是能够在程序中找到对某个行号、整数、变量、字符串、操作代码、压缩格式的字符序列或非压缩格式字符序列等进行访问的所有地方，列出（显示或者打印出）这些访问的参照表。REF具有的格式及其意义如下：

1. REF *

显示出程序中对所有的行号、整数和变量进行访问的参照表。

参照表的显示格式为：首先以升序形式显示对行号或整数的访问表，每个行号或整数显示一行。每一行的第一列是被访问的行号或整数，第二列以后各列是对该行号或整数进行访问的程序行行号（自左至右以升序排列）。然后以变量名的ASCII值的升序形式显示对变量名的访问表，每个变量（不分变量类型）显示一行。每一行的第一列是被访问的变量名，第二列以后为对此变量进行访问的行号。对于变量类型的区分是在访问它的行号后面用/%，/#或/$作后缀表示，无后缀则为单精度变量。

例1. 假设有以下一个简短的程序：

```
10  A=10
15  A%=10
20  B=20
25  A#=20
100 A$="AAAAAAA"
110 B$="BBBBBBB"
140 GOTO 10
```

若打入REF * 命令，则显示参照表如下：

```
10   10 15 140
20   20 25
A    10 15/% 25/# 100/$
B    20 110/$
```

2. REF$

与REF* 命令的功能相同，不过它是在打印机上打印出对所有的行号、整数和变量进行访问的参照表。

3. REF nn

显示对变量名为nn的变量的全部访问。如果nn只有一个字符，则假设第二个字符为空格。变量名nn不能多于两个字符并且不能带有类型后缀（即%，！，#和$）。

例2. 仍采用例1的程序，若打入REF A，则显示如下参照表：

```
A   10 15/% 25/# 100/$
```

4. REF sssss

显示对行号和整数 sssss 的全部访问，其中 sssss 是一个 1 到 5 位数字的 0—99999 之间的十进制数。REF 命令不列出文本中对十六进制或八进制数的访问。

例3. 对于例 1 的程序，若打入 REF 10，则显示如下参照表：

```
10  10 15 140
```

5. REF *nn

与 REF * 命令的功能相同，只是它从指定的变量名 nn 开始（若变量 nn 存在），或者从比变量名 nn 的 ASCII 值高的变量开始（若变量 nn 不存在），列出所有的 ASCII 值更高的变量的访问表（包括指定的变量 nn）。

例4. 对于例 1 的程序，若打入 REF*A，则显示如下访问表：

```
A  10 15/% 25/# 100/$
B  20 110/$
```

6. REF $nn

与格式 5 命令 REF*nn 相同，只是在行印机上打印出访问参照表。

7. REF *sssss

与格式 1 命令 REF* 相同，只是它从指定的十进制数 sssss 开始（若 sssss 存在），或者从比 sssss 大的十进制数开始（若 sssss 不存在），列出对所有的大于 sssss 的整数或行号（包括 $ssss 在内），以及全部变量的访问参照表。

例5. 对于例 1 的程序，若打入 REF *20，则显示如下访问参照表：

```
20  20 25
A  10 15/% 25/# 100/$
B  20 110/$
```

8. REF $sssss

与格式 7 命令 REF *sssss 相同，只是本命令在行印机上打印出访问参照表。

9. REF

显示下一个文本行，这个文本行至少含有一个对上次执行的 REF nn 或 REF sssss 语句所指定的变量或十进制数的访问。如果再也不存在这种文本行，则执行此命令后将显示 "TEXT END"（文本结束）信息。这时候如果再输入 REF 命令，则显示出第一个进行这种访问的文本行。请记住，在执行任何调用 DOS 的命令时，通常会（但不总是）丢失要在文本中寻找的变量名或十进制数以及当前要寻找的行号。

例6. 对于例 1 的程序，若打入 REF 10，则显示出对十进制数 10（行号和整数）的访问参照表，如果接着打入 REF，则显示以下文本行，这个文本行就包含有对 10 的访问。

```
10  A=10
```

若再打入 REF，则显示出下一个文本行：

```
15  A%=10
```

可如此不断进行，直到显示出：

```
TEXT  END
```

此时若再打入 REF 命令，则又显示出第一个进行这种访问的文本行即行 10 的文本行。

10. REF=xxx

命令中的字符序列 xxx 是由标准的 BASIC 文本压缩子程序进行压缩的。这个命令将搜索 BASIC 文本，以找到相符的压缩值 xxx，并列出含有 xxx 压缩值的所有文本行的行号。如果压缩值 xxx 长于 16 个字节，则只对前 16 个被压缩字节部分进行比较。当用户想知道一个 BASIC 操作代码（即 PRINT, LPRINT, GOTO 等）在文本中的什么地方时，就应使用这个格式的 REF 命令，然后可以重复使用格式 9 的 REF 命令，一次显示一个含有 xxx 的文本行。

例7. 仍使用例 1 中的程序

（1）若打入 REF=10，则显示出如下行号：

10 15 140

注意，此时第一列的 10 与前面讲的参照表中第一列的意义不同，它不是被访问的十进制数，而与其后各列一样，是含有字符序列 10 的文本行的行号。

（2）若打入 REF==，则显示出含有等号（=）的文本行的所有行号：

10 15 20 25 100 110

（3）若打入 REF==10，则显示出如下行号：

10 15

如果此时象例 6 那样，使用格式 9 的 REF 命令，则可以不断显示出含有=10 的文本行：

10 A=10

……

（4）若打入 REF=GOTO，则显示行号： 140

（5）若打入 REF=A#=20，则显示行号：20

11. REF″ xxx

这个格式的命令类似于格式 10，只是此命令中的 xxx 没有被压缩，可以把 xxx 看作字符串，除非 xxx 本身含有″号。这个格式的 REF 命令允许在字符串和注释中寻找 xxx 字符序列。

例8. 对于例 1 的程序文本

（1）若打入 REF″ AAA，则显示出找到字符串 AAA 的所有文本行的行号：

100/2

这表示在行 100 的文本行中找到了 AAA，其中的后缀/2 表示该行有 2 个 AAA。

（2）若打入 REF″ AA，则显示：

100/3

（3）若打入 REF″ A，则显示：

10 15 25 100/8

12. REF @ sssss

这个命令与格式 9 命令 REF 相同，只是它从行号大于等于 sssss 的文本行开始往下寻找。

例9. 如例 6 那样，先打入 REF 10，在显示出参照表以后打入 REF @ 20，则从行号 20 的文本行开始往下寻找对 10 进行访问的文本行并显示出下个文本行的行号，由例 1 程序可

知，在行 20 以后，只有行 140 的文本中含有对 10 的访问，所以这个命令将显示出这行文本：

```
140 GOTO 10
```

如果此时再打入 REF 命令，则显示：

```
TEXT  END
```

要暂停 REF 的功能时，请按 BREAK 键。如果再按 ENTER 键，则可继续执行。要结束 REF 操作，请按向上箭头键。

若执行 REF 语句时，SYS12/SYS 不在系统中，则系统将退出到 DOS（见 5.5 节）。

7.11 抑制文本串中的小写字母（只适用于 TRS-80 Ⅰ型机）

在 TRSDOS BASIC 中，那些没有英文小写字母修改功能硬件的用户，或者他们虽然有但不使用小写驱动程序来代替 ROM 显示程序时，那末他们有时候会被正确语句中出现的某些字符串比较的错误结果和某些句法错误搞糊涂。这是由于接收到的输入字符串中的小写字母被显示成了大写字母，或者在文本的语句中接受了小写的@符号（这个符号只有一种显示形式）。请记住，虽然 ROM 显示程序把小写字母显示成大写字母，但它们在内存中依然是小写字母的 ASCII 代码，因此，对于字符串比较的内容，可能在显示器上看是相等的，而实际上在内存中是不相等的。

在 NEWDOS/80 中，为防止以上问题，可通过设置 SYSTEM 参数来抑制文本中的小写字母。对于输入的文本字符串，若 SYSTEM 的选用参数 AS=Y，那么文本串中的小写字母将被强制变成大写，但小写的@符号不变（请参阅 2.46 节），因此消除了许多这样的问题。

7.12 RUN-ONLY（只能执行）方式

在 NEWDOS/80 BASIC 中，除了正常的命令方式外，还有一种 RUN-ONLY 方式，即只能执行方式。在这一方式中，BASIC 只能执行用户的程序，而不接受用户的直接命令，BREAK 键也不能用于打断程序的执行，并且禁止使用 JKL 把荧光屏内容复制到打印机上，禁止用 123 调用 DEBUG，禁止通过 DFG 调用 MINI-DOS 等。

对于 NEWDOS/80 磁盘 BASIC，有两种方法可使 BASIC 运行于 RUN-ONLY 方式：

（1）若 SYSTEM 选用参数 AB=Y。

（2）若 BASIC 程序文件被通行字保护着（允许使用通行字时）。在 RUN 或 LOAD（带有选用参数R）语句中指定了存取通行字并且存取级为 EXEC。

如果 SYSTEM 选用参数 AB=Y，由于进入 RUN-ONLY 方式后不允许用户输入任何直接命令或语句，所以启动 BASIC 的 DOS 命令必须含有 RUN 或 LOAD（带有选用参数 R）语句，以启动 BASIC 并在 RUN-ONLY 方式下执行用户程序。

在 RUN-ONLY 方式中，BREAK 键无效，并且禁止 BASIC 接受用户的直接命令（可以接受数据）。因此，程序具有完全的控制权。用户可以使用一个菜单(MENU)程序发出RUN或 LOAD（带选用参数 R或 V）命令来执行别的 BASIC 程序，当然这些程序也同样可以这样做，以返回到 MENU 程序，或转到下一个程序序列。或者让基程序按需要永远驻留在内存中，并通过语句 CMD "F"，DELETE 和 MERGE，在需要时引入覆盖程序。为此用户应仔细地学习RUN，MERGE，LOAD 和 CMD "F" 等操作的可用的选用参数。

7.13 NEWDOS/80 和 TRSDOS 之间 CMD 操作的比较

NEWDOS/80 BASIC 与 TRSDOS BASIC 在 CMD命令的使用上有很大的差异，NEWDOS/80 BISIC 保留了一部分 TRSDOS BASIC 的 CMD 功能，而不执行另一部分 CMD 的功能（虽然使用它们时不会给出错误信息），这一部分的 CMD 的功能在 NEWDOS/80 中要用另一种形式的命令来执行，另外还增加了一些 TRSDOS 中没有的 CMD 功能。现在按字母顺序将它们之间的功能和用法比较如下：

1. CMD″A″

NEWDOS/80 BASIC 不用此命令，而使用 CMD″S″ 来返回操作系统。

2. CMD″B″

在 TRS-80 Ⅰ型机上，NEWDOS/80 BASIC 和 TRSDOS BASIC 都不用这个命令。在 TRS-80 Ⅲ型机上，TRSDOS BASIC 用这个命令，而 NEWDOS/80 BASIC 不用这种形式的命令，而使用 CMD ″BREAK，Y/N″ 来允许和禁止 BREAK 键的作用(见2.8节和7.14节)。

3. CMD″C″

这个命令是 NEWDOS/80 BASIC 中新增加的，它的功能为（1）压缩程序文本中的全部空格，但字符串中的空格除外。（2）从文本中删除所有的注释（包括全是注释的行)。

这个命令还有另外两种格式，它们只完成上述两个功能之一。

CMD″C″，S 这个命令的功能是上述功能之（1），即压缩程序文本中的所有空格，但不压缩字符串和注释中的空格，并且不删除文本中的注释。

CMD″C″，R 这个命令的功能是上述功能之（2），即从文本中删除所有的注释，而不压缩空格。

在某些情况下，如果程序中的 GOTO，GOSUB 等访问一个全是注释的文本行，那末从文本中删除注释将使得这种被访问的行消失，在执行程序时就会产生错误。所以应把程序中的这种 GOTO，GOSUB 语句进行修改，使它访问的不是注释行。另外，当从程序中删掉注释以后，最好执行 RENUM U 命令来检查一下整个文本，看看在删除注释后文本中是否存在没有定义的行号。

虽然 BASIC 允许忽略文本中不属于注释和字符串中的空格，但是有时候从文本中去掉空格仍然会引起混乱。例如，若将下列程序：

```
10  FIELD  1,20  AS  C$
20  IF  F  OR  D  THEN  10
```

压缩成以下形式：

```
10  FIELD1,  20ASC$
20  IFFORDTHEN10
```

那末，无论在（1）程序已经以 ASCII 格式存储到盘上，然后再读入；还是（2）编辑这些程序行以后，在执行过程中遇到这两行程序时都会引起句法错误。为避免这些问题，CMD ″C″ 功能自动地分解每一个压缩的文本行（即以压缩格式存储的文本行），再重新压缩它们（即执行 CMD ″C″ 功能），并将新压缩的文本与老的文本（即空格被压缩掉以前的文本）进行比较。对于任何文本行，若两种压缩的文本中有任何差别（不包括注释)，则将空格再放入文本行（注释若已被删掉，则仍然保持删掉)，并在显示器上列出其行号。于是用户可以检

查这些程序行，将其中没有作用的空格去掉。对于任何给定的程序，只有极少数的程序行被CMD〃C〃所拒绝。

4. CMD 〃D〃

在TRS-80 Ⅲ型机上，NEWDOS/80不执行TRSDOS的这个语句的功能，而要使用CMD〃doscmd〃这种形式的命令。在TRS-80 Ⅰ型机上，虽然NEWDOS/80通过'123'联键功能调用DEBUG是较好的方法，但仍可使用CMD〃D〃这个语句来调用DEBUG。

5. CMD〃E〃

这个命令在NEWDOS/80与TRSDOS中是一样的，它用于显示与BASIC最近遇到的DOS错误有关的DOS错误信息。

6. CMD〃F〃

这个命令在TRSDOS中没有，在NEWDOS/80中有以下两种格式：

(1)CMD 〃F〃, fc 当功能代码fc必定能被REF, RENUM等找到的时候使用。

(2)CMD〃F=fc〃 在功能代码fc不能被REF, RENUM等查看的时候使用，或者特别规定的功能代码可能被正常的文本压缩子程序搞乱的情况下使用。

这些CMD 〃F〃命令的功能将在7.15节到7.20节中详细讲述。

7. CMD 〃I〃

在TRS-80 Ⅰ型机上，NEWDOS/80和TRSDOS都不用这个命令；在TRS-80 Ⅲ型机上，NEWDOS/80不用TRSDOS的这个命令，而使用CMD 〃doscmd〃来执行DOS命令，见7.14节。

8. CMD 〃J〃

这个NEWDOS/80 BASIC命令用于日历日期的转换，命令格式为：

CMD 〃J〃, date 1, date 2

功能是把字符串表达式date 1转换成适当的格式，其格式取决于date 1自身的格式，并把转换结果放入字符串变量date 2。如果date 1是mm/dd/yy格式，则转换成ddd格式存入date 2；如果date 1是－yy/ddd格式，则把它转换成mm/dd/yy格式存入date 2。其中：

mm 是01到12之间的两位数字的月份值。

dd 是01到31之间的两位数字的一个月内的日期值。

ddd 是一年内的日期值，在001到366之间。

yy 是两位数字的相对的一个世纪中的年份值，在00到99之间。对于闰年的转换，假设yy是在第二十世纪内，即只适用于从1900年到1999年之间。

举例

例1. CMD 〃J〃, 〃04/20/84〃, D$:PRINT D$

将显示出从元月一日算起，1984年4月20日是这一年的第几日：111

例2. CMD〃J〃,〃01/30/84〃, D$: PRINT D$

显示：030

例3. CMD〃J〃,〃02/29/83〃, D$:PRINT D$

则显示：ILLEGAL FUNCTION CALL（非法的功能调用）

这是因为1983年不是闰年，2月份只有28天，而1984年是闰年，如下例所示。

例4. CMD〃J〃, 〃－84/060〃, D$:PRINT D$

将显示：02/29/84

例5. CMD "J", "-83/060", D$:PRINT D$

则显示：03/01/83

9. CMD"L"

在TRS-80 Ⅲ型机上，NEWDOS/80 BASIC不执行这个TRSDOS的CMD "L"命令的功能，而要使用CMD "LOAD, filespec" 来装入指定的文件，见7.14节。这个命令在TRS-80 Ⅰ型机上也不用。

10. CMD"O"

在NEWDOS/80 BASIC中，这个命令的功能是进行数组排序，请参阅下面7.21节关于这个命令的详细论论。

11. CMD"P"

这个命令在TRS-80 Ⅰ型机上不用，在TRS/80 Ⅲ型机上，NEWDOS/80中不执行TRSDOS的这个命令的功能，而要使用PEEK (&H37E8)来得到关于当前打印机状态的值(0—255)。

12. CMD"R"

在TRS-80 Ⅲ型机上，NEWDOS/80 BASIC不执行这个TRSDOS的CMD "R"命令的功能，而要使用CMD "CLOCK, Y" 来启动实时时钟（见2.12节和7.14节)。在TRS-80 Ⅰ型机上，这个命令的功能在NEWDOS/80和TRSDOS中是一样的，CMD "R" 仍象以前那样重新启动实时时钟。

13. CMD"S"

这个命令在NEWDOS/80 BASIC中的功能仍然与TRSDOS BASIC中一样：退出BASIC并返回到DOS READY状态。然而，在NEWDOS/80中，若命令的形式为CMD "S=doscmd"，则如下进行：

(1)将DOS命令doscmd送入DOS命令缓冲区。

(2)退出BASIC而不出现DOS READY。

(3)立即执行放在DOS缓冲区的DOS命令。

(4)在执行完该命令后，控制返回到操作系统而不返回到BASIC，出现DOS READY。

14. CMD "T"

在TRS-80 Ⅲ型机上，NEWDOS/80 BASIC中不执行TRSDOS的这个命令的功能，而要使用CMD "CLOCK, N" 来关闭实时时钟（见2.12节和7.14节)。在TRS-80 Ⅰ型机上，CMD "T" 命令在NEWDOS/80 BASIC中仍象以前在TRSDOS BASIC中一样关闭实时时钟（禁止中断)。

15. CMD"X"

在TRS-80 Ⅰ型机上，NEWDOS/80不用此命令。在TRS-80 Ⅲ型机上，NEWDOS/80 BASIC不执行TRSDOS BASIC的这个命令的功能，而使用REF命令（见7.10节)。

16. CMD"Z"

在TRS-80 Ⅰ型机上,NEWDOS/80不用这个命令。在TRS-80 Ⅲ型机上，NEWDOS/80不执行TRSDOS的这个命令的功能，而使用CMD "ROUTE, ..."(见2.43节和7.14节)。

7.14 CMD "doscmd" 执行DOS命令

这个NEWDOS/80 BASIC命令用于在BASIC中执行一个DOS命令。其中，doscmd是一个规定命令功能的字符串表达式。如果这个与CMD功能有关的字符串表达式有两个以上的字符，并且不以"S="或"F="开头，那么就假设此字符串是DOS执行的一个命令。BASIC将此命令送入DOS的命令缓冲区，将DOS置成MINI-DOS方式，并通过4419H的DOS-CALL调用DOS去执行这个命令。在返回时，BASIC关闭DOS的NINI-DOS方式。如果由于此命令在MINI-DOS下是不合法的而被DOS拒绝，那末BASIC就进行以下工作，设法把此命令重新发送给正常方式下的DOS:

如果在BASIC数组区的顶端与BASIC堆栈的底部（堆栈紧接在字符串区之下）之间没有大约8000个字节的空间可以使用，那末BASIC将宣布"OUT OF MEMORY"（内存不够）错误并结束当前执行的语句。如果有这么多空间可用，BASIC就把5200H—70FFH之间的全部内存内容移到该空闲区，规定它本身使用7100H—71FFH堆栈区，并在7100H到BASIC的内存顶端之间的区域上计算检查和（大约要2秒钟时间）。然后，BASIC调用DOS去执行此DOS命令。当从DOS返回时，BASIC将此存储区域移回到5200H—70FFH，并重新计算检查和（再要2秒钟）。如果核对这两个检查和不相符合，则意味着执行的这个DOS命令已改变了一些BASIC的字节，于是BASIC就不能继续工作，而退出到DOS，并给出"BAD MEMORY"（内存储器有错）错误信息。

在无论用哪个方法执行DOS命令后，BASIC都要检查由DOS送回的返回代码。如果发生了一个错误并已显示出错误信息，BASIC就以"PREVIOUSLY DISPLAYED ERROR"（前面显示过的错误）错误状态结束CMD "doscmd"语句。如果发生一个DOS错误，BASIC就调用4409H去显示DOS错误信息，并以"DOS ERROR"（DOS错误）错误状态结束CMD "doscmd"语句。如果没有发生错误，则BASIC继续正常的处理。

任何DOS库命令或汇编语言程序(它将只用5200H—6FFFH区域和主存储器的非BASIC非DOS区域来执行）都可以用这个形式的BASIC命令来执行。SUPERZAP和DIRCHECK是两个可以通过CMD "doscmd"命令来执行的程序。FORMAT和多数格式的COPY命令也可以这样执行，但是在单驱动器上，两个盘片的复制不能进行，因为它们需要大量的内存储器，另外，在COPY命令中不要指定UBB参数。

请记住，CMD中的DOS命令，包括ENTER字符在内不能超过80个字符，此ENTER字符是在BASIC将doscmd移到DOS命令缓冲区去时加到命令中去的。

在此还要告诫用户，你的程序应保持TRS-80 I型机的内存储区4080H—41FFH(TRS-80 III型机的4080H—41E2H）不受干扰，除非其中的变化与BASIC的当前用法是相符合的。

CMD "BASIC"命令是永远不会被执行的。如果出于某种理由，用户想退出BASIC再返回BASIC，请使用CMD"S=BASIC"命令。

几乎所有的DOS命令都可以通过CMD "doscmd"执行。例如：

1. CMD "DIR 1" 列出1号盘的目录。
2. CMD "COPY XXX:0 YYY:1" 复制一个文件。
3. CMD "COPY 0 1 07/10/81 FMT" 复制整个磁盘并进行格式化。

4. CMD "SUPERZAP" 执行 SUPERZAP 程序并返回到 BASIC。

5. CMD "DO CHAINFIL" 执行链接文件功能并返回。

7.15 CMD "F", DELETE 动态地删除文本行

本节介绍 7.13 节 6 的 NEWDOS/80 BASIC 命令 CMD"F" 的第一种格式CMD "F", fc。此格式只有这一个命令，其格式为

CMD "F", DELETE ln1—ln 2 。

其中，ln 1 和 ln 2 是文本行行号，并且 ln1<ln 2 。

这个语句的功能类似于 DELETE 命令，但它允许在程序执行过程中将行号 ln 1 到行号 ln 2 （包括 ln 1 和 ln 2 ）之间的文本行删掉，即所谓动态删除文本行，并且清除被删除范围内的 DEFFN 语句定义的变量而其他的所有变量都保持不变，字符串区的大小也不变。任何字符串变量，若其当前的字符串实际在被删除的文本区中，则将该字符串移到字符串区中。

CMD "F", DELETE 语句不能作为直接语句执行，不能包含在 DEFFN 语句中，不能包含在子程序或 FOR-NEXT 循环中（由于隐含执行下节讲述的 POPS 功能）。另外，CMD "F", DELETE 语句必须是文本行中的最后一个语句，并且在它下面必须跟随一个文本行，使删除完成以后由此继续执行。

例如：

```
100  CMD"F", DELETE 10500—15000
110  X=1  完成 DELETE 后由此继续执行。
```

7.16 CMD"F=POPS", CMD"F=POPR" 和 CMD"F=POPN" 清除堆栈中控制信息

本节至 7.20 节介绍 7.13 节 6 的 CMD"F" 命令的第二种格式 CMD"F=fc" 各语句的功能。本节讲述的三个语句都与堆栈操作有关，用于清除堆栈中残留的 RETURN、FOR-NEXT 控制信息。

1. CMD"F=POPS"

其功能是把所有的 RETURN 和 FOR-NEXT 控制都清除掉，使 BASIC 不存在未完成的 RETURN 和 NEXT 操作，在做完以后，继续执行下一个语句。本语句的目的是使用户“跳出”复杂的程序，返回到 BASIC 的初始状态，这就避免了在 BASIC 的控制堆栈中留下残剩信息。如果程序中有没有完成的 RETURN 和 NEXT 操作，而又没有 CMD "F=POPS" 语句，那末当此堆栈递归地返回到高层时，堆栈中的这些残留信息最终将导致程序执行的失败。

2. CMD "F=POPR"

这个语句的功能是清除当前的 GOSUB 级以及该级的任何未完成的 FOR-NEXT，其功能与 RETURN 是相同的，只是 RETURN 语句把控制传送给相应的 GOSUB 后面的语句，而它把控制传送给 CMD "F=POPR" 语句后面的语句。

3. CMD "F=POPN"

这个语句把最近建立的FOR-NEXT控制数据清除掉，其功能与超过循环范围时的 NEXT 语句一样，然后，执行将由 CMD "F=POPN" 语句后面的语句继续进行。

如果语句是 CMD "F=POPN" Vn, 其中 Vn 是一个变量名，那末这个语句把与 Vn 有

关的 FOR-NEXT 循环，以及在 Vn 循环未完成时建立的任何其他的 FOR-NEXT 循环（即 Vn 循环的内层循环）都清除掉。这个语句的功能与循环结束时的 NEXT Vn 语句是一样的。这个语句执行完以后，就由 CMD〃F=POPN〃Vn 语句后面的语句继续执行。CMD〃F=POPN〃的目的是允许脱出一个循环而不留下残余的控制信息，如果用户不用这个语句而用 GOTO 等语句来脱出循环，并且接着用与原先的循环变量的相反次序使用 FOR-NEXT 变量时，这些残余在堆栈中的控制信息就会把他搞糊涂，使程序的运行得不到预期的正确结果。

例如：

CMD 〃F=POPN〃 I　清除I循环和它的内层循环在堆栈中的控制信息。

7.17 CMD〃F=SASZ〃 改变字符串区的大小

这个 NEWDOS/80 BASIC 命令用于改变 BASIC 的字符串区的大小而不影响或清除变量。其格式为：

CMD 〃F=SASZ〃, exp 1

其中，exp 1 必须是一个数值，并足够大，使字符串区包含该语句被执行时已含有的字符串。如果 exp 1 太小或太大（使得字符串区覆盖了文本区、变量区和数组区），则要发生错误。

例如：

CMD 〃F=SASZ〃, 4000　把字符串区大小改成 4000 个字节。

7.18 CMD 〃F=ERASE〃和 CMD 〃F=KEEP〃 清除或保留变量

这两个 NEWDOS/80 BASIC 的命令的功能是有选择地清除或保留 BASIC 变量。格式如下：

1. CMD 〃F=ERASE〃, Vn1, Vn2, Vn3…

其中 Vn1, Vn2 和 Vn3 等是变量名。其功能是清除这些指定的变量。若指定的变量在一个数组中（即是一个下标变量），则清除整个数组。这个语句不改变字符串区的大小。这个语句可以在一个数组不够大时，或想用后边的 DIM 语句重新定义一个数组时使用。这个语句可以象下面的 CMD 〃F=KEEP〃 语句那样，由多个文本行组成。

例如：

CMD〃F=ERASE〃, A$, B%, C, D#, E(1)

这个语句清除了变量 A$, B%, C, D# 和数组 E。

2. CMD 〃F=KEEP〃, Vn1, Vn2, Vn3…

如上面的命令一样，Vn1 等是变量名。这个命令的功能是保留这些指定的变量和特殊定义的变量（如由 DEFFN 语句定义的变量）不被清除，除此以外的所有变量都被清除。这个命令也不改变字符串区的大小。命令中若不指定变量名，则除特殊的变量以外，清除所有的变量。如果指定的变量名是数组中的一个下标变量，则保留整个数组。这个语句可以根据用户的需要随意指定很多变量名，以致这个语句长到由几个文本行组成，这时候，除了语句的最后一行以外，其他各行的最后一个变量名后都要放一个逗号作为文本行的结尾。

例如：

CMD 〃F=KEEP〃, A$, B%, C, D#,　语句的第一行。

E!, F, G$, 语句的第二行。

REM this line is bypassed 注释行。

H!, I 语句的最后一行。

7.19 CMD"F=SWAP" 交换变量内容

这是一个 NEWDOS/80 BASIC 的命令，其格式如下：

CMD "F=SWAP", Vn1, Vn2

其中 Vn1, Vn2 是变量名。这个语句的功能是将变量 Vn1 的值与变量 Vn2 的值交换。两个变量必须类型相同，即两者都是字符串，或都是单精度浮点变量等等。例如：

CMD "F=SWAP", A$, B$ 交换字符串变量 A$ 与 B$ 的内容。

CMD "F=SWAP", B#, C# 交换双精度变量 B# 与C# 的内容。

7.20 CMD"F=SS" 单步执行 BASIC 程序

这个 NEWDOS/80 BASIC 命令有以下三种格式：

1. CMD"F=SS" 启动单步功能。
2. CMD"F=SS", ln1 从行 ln1 开始单步执行。
3. CMD"F=SS", N 撤消单步功能。

使用上面的格式 1 或格式 2 命令，可将 BASIC 置成单步方式。在启动单步功能以后，仍要用 RUN 命令来执行程序。格式 1 命令使程序中行号最小的文本行成为待单步执行的第一行。但是，对于格式 2 命令，实际上在文本行 ln 1 成为待执行的下一行以前并不开始单步执行。

在单步方式下，每一步只执行一个 BASIC 文本行。在每一步之间，待执行的下一行行号将以"@ nnnnn" 格式显示在荧光屏的右上角，以表明 BASIC 正等待用户给予响应。若按 ENTER键，就执行行nnnnn, 然后 BASIC 又等待用户的响应。若按 BREAK 键，就使执行以正常方式中断，应注意，按 BREAK 键时显示的行号是刚执行的行（或正在执行的行），而单步执行显示的行号 "@ nnnnn" 是待执行的下一行。如果用户在 BREAK 期间不改变程序文本，就可以通过 CONT 命令继续单步执行，在这种情况下，行号 "@ nnnnn" 将立即重新出现在荧光屏右上角，但此时一行也没有执行，按 ENTER 键才执行此行。在 BREAK 期间用户可以随意地启动或撤消单步功能，而不影响 CONT 的功能。当然，与通常情况下一样，在 RUN 或 LOAD（带有 R 参数）执行任何文本行以前就发生 BREAK 时，CONT 是不起作用的。

启动后的单步功能或单步调度在遇到特定的文本行以前是一直保持有效的，要一直保持到执行 CMD" F=SS", N 语句关闭单步功能为止，或执行格式 2 形式的单步命令为止，在后一种情况，单步功能在遇到指定行以前一直关闭着。执行 NEW, RUN "filespec 1", LOAD 等命令都影响单步状态，它们都使单步状态成为执行格式 1 命令(即 CMD "F=SS") 后那样的状态，而不论前面启动单步功能的命令是 CMD "F=SS" 还是 CMD "F=SS", ln1。

7.21 CMD"O" BASIC数组排序

NEWDOS/80 BASIC 的 CMD"O" 语句的功能是在内存中进行数组元素的排序。

在说明这个排序命令时，要使用到术语 REN，其意义是标识一个数组元素的相对元素号（Relative Element Number）。任何 BASIC 数组中的元素，不论数组的维数是多少，REN 都从 0 开始往上按整数编号。如果一个数组只有一维，则元素的 REN 就简单地是它的下标值。如果你只使用一维数组，那末可以忽略这一节关于 REN 计算方法部分。但是，如果你使用多维数组，则必须知道使用什么方法去增加数组下标值，以便按排序的次序取出元素。因为这个 CMD "O" 语句不管数组有多少维，它只按 BASIC 在内存储器中存储数组元素的次序来排序。作为一个程序员，当你必须使用下标去存取数组元素时，也要知道计算数组元素的 REN 的方法。对于多维数组，计算 REN 的规则是很复杂的，但如果使用一个三维数组作例子，则可以使用以下两个语句来很好地说明 REN 的计算方法：

```
DIM A(R1, R2, R3)
Y=A(X1, X2, X3)
```

那末这个数组元素 A(X1,X2,X3)的 REN 按 X1+X2*(R1+1)+X3*(R1+1)*(R2+1)计算。如果此数组只有两维，则 A(X1,X2) 的REN 等于 X1+X2*(R1+1)，当然，如果数组只有一维，则 A(X1) 的 REN 就简单地为 X1。

CMD "O" 语句有以下两种格式：

1. CMD "O", n, av1[,av2,…] 直接排序。
2. CMD "O", n, *iav1, av2[,av3,…] 间接排序。

其中 av1, av2, av3等是要进行排序的数组的下标变量名，最多可以有 9 个。n 是一个大于等于 0 的整数，它表示参与排序的元素数目。

如果 CMD "O" 语句指定几个数组（不包括 iav1），那末每个数组（iav1 除外）中的第一个排序项的 REN 必须相等。

对于每一个指定的数组（不包括 iav1），使用的排序次序有一个等级，在 CMD "O" 语句中，从左边的数组变量到右边的数组变量，其次序为由最高级到最低级。在每一级中，字符串数组的正常排序次序是上升的 ASCII 值（实际为十六进制数值）；对于数值数组为从最大的负数（绝对值最大）到最大的正数。但是，如果在 CMD "O" 语句中的数组变量前放一个减号（例如，－A#(0)），则该级中的排序次序，对于字符串数组为下降的 ASCII 值；而对于数值数组为从最大的正数到最大的负数（绝对值）。空字符串在比较中可以认为它是个小于 0 的数值。

通常在字符串比较中，使用整个字符串进行比较，但是，如果 CMD "O" 语句中的数组变量用一个 (X,Y) 形式的字段结尾（例如，A$(1)(5,4)），那末它表示由该字符串变量的第X字符开始比较，并只用Y 个字符进行比较。例如，A$(1)(5,4)，表示由 A$(1) 的第 5 个字符开始用 4 个字符进行比较（即比较第 5—8 个字符）。但是，如果该字符串变量的字符串少于X个字符，则把它作为空字符串处理。

语句中指定的整数 n 是参与排序的每个数组中的元素数，即每个数组中只有 n 个元素参与排序，而在指定的 n 个元素以下或以上的数组元素都不参与排序。如果 n 等于 0，那末在排序中，n 被置成语句所指定的第一个数组中从指定元素开始到该数组的最后一个元素之间的元素数（包括首尾两元素）。

如果任意数组中从指定元素到该数组的最后一个元素（包括首尾两元素）之间的元素数小于 n，则宣布 FC 错误。

CMD "O" 语句中最多可以指定 9 个数组。所有的数组变量的下标，除用于间接排序的数组（若指定时）以外，都必须具有相同的 REN 值。例如：CMD "O", 100,A$(1), B!(1), C#(1)

格式 1 的 CMD "O" 语句是直接排序语句，它意味着数组 1 到数组 9 的所有元素都要被前后移动，以符合所要求的排序次序。

在这个格式中，必须指定 av1，而 av2 及以上各项为可有可无。在排序完成以后，每个数组中 n 个元素的结果序列对于每个数组是相同的（即各数组不是独立地排序的）。因此，如果数组 1 的第 j 元素被排序到第 k 元素的位置，那末对于其他的每一个数组（若有的话），第 j 元素也放入第 k 元素的位置（见例 2 得到的 AM! 和 LN$ 的结果序列）。

格式 1 的 CMD "O" 语句在 TRS-80 Ⅲ型机上与 TRSDOS BASIC 的 CMD"O" 是兼容的，但是仅仅在只指定一个数组变量，而且是字符串数组以及 n 是整变量时才兼容。

格式 2 的 CMD "O" 语句的功能是进行间接排序。在这种排序中，只改变数组 iav1 的 n 个元素，其他数组一点不变。这个语句的目的是允许确定一个排序序列而实际不改变提供排序值的数组。一个用户可能有一组散布在一些数组中的数据记录，由此，由每个数组的一个元素组成一个记录，用这些元素的每一个的 REN（相当于记录号）来编制这个记录。使用有间接数组的格式 2 排序语句，用户可以用项目的子集作为排序准则有效地排序此记录，而实际上不重新排列该记录的次序，因此，保持了它们的记录号次序。

格式2语句与格式1不同，它用指定 iav1 数组变量并在其前边放一个＊号来表示。

在格式 2 的 CMD "O" 语句中，iav1 必须是一个整数数组变量。并且必须指定数组 av2，而数组 av3 及以上数组项是可有可无的。

在排序中，位于 iav1 的 n 个相邻的元素，首先将用对应数组 av2 的 n 个相邻元素的 REN（也对应其他数组的 REN，若有这些数组的话）进行预置。

在相对排序中，只有数组 iav1 被改变，数组 av 2 及以上数组都不变。

排序完成以后，数组 iav1 的 n 个元素就按所需的排序次序排列好了，由此，用数组 iav1 的这 n 个连续元素的值作为下标，用户就可以按此排序次序由其他任何一个数组（它们是一维的）中存取元素。存取多维数组比较复杂，在此作为一个练习留给水平较高的用户去做（用 iav1 的 n 个连续元素的值作 REN 去存取多维数组的元素）。

例1. 下面是一个使用排序语句的例程序：

```
10  DIM  NM$(200), AM!(100), LN$(100), IX%(100), ZC!(50), L$(50)
.....
30  X=150
40  CMD"O", X, NM$(0)
60  CMD"O",  X,  -NM$(25)
70  CMD "O", 0, -AM!(1), LN$(1)(5,3)
80  CMD "O", 50, *IX%(0), ZC!(1), L$(1)
```

此例程序中，行 40、60 和 70 都是直接排序，行 80 是间接排序。

在行 40 的排序中，数组 NM$ 的前 150个元素（即元素NM$(0)到LM$(149)）按上升次序进行排序。如果其中有的元素的字符串值是空，那末它们将出现在结果序列的最前边。数组NM$ 的后 51 个元素（元素 NM$(150)到 NM$(200)）不参与排序并保持不变。

在行 60 的排序中，元素 NM$(25) 到 NM$(174)(也是 150 个元素) 被排序成下降序列，而空字符串（若有的话）就作为这 150 个元素的最后元素出现。数组 NM$ 的前25个元素和后 26 个元素都不参与排序。

在行 70，数组 AM! 和 LN$ 是以相同的次序排列的，它首先用 AM! 数组元素值的降序进行排序（因为语句中 AM! 前有一个负号），然后按照 AM! 的结果序列的次序来排列 LN$，并且不管LN$(1)(5,3) 这样的形式。这种形式只有作为 CMD "O" 语句格式中的 av1 出现时，才使用 LN$ 数组元素的第 5、6、7 三个字符对 LN$ 进行排序。在行 70 排序中，数组的 AM!(0)和LN$(0) 元素不参与排序。由于排序的元素数被指定为 0，所以排序的元素数取 100，即 AM! 数组的 AM!(1) 元素到数组的最后一个元素 AM!(100) 之间的元素数。

行 80 与行 40 等不同，它是一个间接排序语句。在这个间接排序中，IX%数组的前50个元素首先用 REN 值（1 到 50）顺序地预置，并且 IX%(0)=1，IX%(1)=2，… IX%(49)=50。这个语句规定排序是升序的，它用 ZC! 数组值对 IX% 数组进行排序。数组 ZC! 和 L$ 没有一个元素被改变。排序中只有 IX%数组中对应的 REN 被移动，而不移动 ZC! 和 L$ 数组元素。在排序完成以后，就可以用这些 REN 作下标去索引数组 ZC! 和 L$，例如，可以用 IX%(0)中的 REN 作为下标从数组 ZC! 和 L$ 中去检索排序好的第一个元素，而 IX%(49) 中的 REN 可用于从每一个 ZC! 和 L$ 数组元素中检索排序好的最后一个元素。最后，请记住，元素 IX%(50)—IX%(100) 以及 ZC!(0) 和 L$(0) 丝毫不参与排序。

例2. 为了使读者对排序语句的功能和使用方法有一个比较完整的概念，在这个例子中，我们利用例 1 的数组，但缩小了数组的规模，并用随机函数赋给各数组元素一些无规的值，然后用例 1 的排序语句进行排序。此例程序还把排序前后的各数组元素值打印出来，以进行比较。例程序清单如下所示：

```
10  CLEAR  2000
20  DIM  NM$(20), AM!(10), LN$(10), IX%(10), ZC!(5),L$(5)
30  RANDOM: LPRIN "****  BEFORE SORT  ****"
40  FOR  I=0  TO  20
50  NM$ (I)=CHR$(32+RND (90))
60  NEXT:  GOSUB 1000
70  FOR  I=0 TO  10
80  AM! (I)=RND(100)/0.1
90  LN$ (I)=CHR$(32+RND(90))+CHR$(32+RND (90))+CHR$(32+
    RND(90))+CHR$(32+RND(90))
100 NEXT:  GOSUB 2000
120 FOR  I=0 TO 5
130 ZC! (I)=RND (50)/0.1
140 L$(I)=CHR$(32+RND(90))
150 NEXT:GOSUB 4000
170 X=15:LPRINT" ":LPRINT "****  AFTER  SORT  ****"
180 CMD "O", X, NM$(0):GOSUB 1000
```

```
190 CMD "O", X, -NM$(5):GOSUB 1000
200 CMD "O", 0, -AM!(1), LN$(1)(2, 3):GOSUB 2000
210 CMD "O", 5, *IX%(0), ZC!(1), L$(1)
220 GOSUB 3000:GOSUB 4000
230 LPRINT"ZC!=":FOR I=0 TO 4:LPRINT ZC!(IX%(I));:NEXT I:LPRINT" "
240 LPRINT"L$=":FOR I=0 TO 4:LPRINT L$(IX%(I)); " "; :NEXT I:
    LPRINT" "
250 END
1000 LPRINT"NM$=":FORI=0TO 20:LPRINT NM$(I);" ";:NEXT I:LPRINT
    " "
1010 RETURN
2000 LPRINT "AM!=":FORI=0TO 10:LPRINT AM!(I); :NEXT I:LPRINT
    " "
2010 LPRINT "LN$=":FOR I=0 TO 10:LPRINT LN$ (I);" ";:NEXT I:
    LPRINT" "
2020 RETURN
3000 LPRINT "IX%=":FOR I=0 TO 10:LPRINT IX% (I);:NEXTI:LPRINT" "
3010 RETURN
4000 LPRINT "ZC!=":FOR I=0 TO 5:LPRINT ZC! (I);:NEXTI:LPRINT" "
4010 LPRIN "L$=":FOR I=0 TO 5:LPRINT L$(I);" ";:NEXTI:LPRINT" "
4020 RETURN
```

在这个例程序中，行 40—150 利用随机函数 RND (X) 给各数组赋值。行 230，240 以及子程序 1000，2000，3000 和 4000 用于打印排序前后各数组的元素值。行 180—210 的排序语句与例 1 的行 40—80 是一样的，只是规模较小而已，它们已在例 1 中进行了详细的解释。

以下是各数组在排序前的元素值：

```
    **** BEFORE SORT ****
NM$(0)—NM$(20):
Z B q C X; 't B u T w ← ' ↓ z u 3-z $
AM!(0)—AM!(10):
160 750 70 260 890 600 150 90 980 900 920
LN$(0)—LN$(10):
2g`@ 2koU D6wk x7xU j2,- A3! N EJNI JLUW }3@i vE*v %c/d
ZC!(0)—ZC!(5):
30 90 280 220 450 170
L$(0)—L$(5):
6 ↓ ← " . j
```

在执行 CMD "O" 语句以后，各数组的值如下所示：

```
    **** AFTER SORT ****
NM$:  (直接排序 NM$(0)—NM$(14), 升序排列)
```

' ' ; B B C T X Z↓ ← q t u w z u 3 - z $

NM$: (直接排序NM$(5)—NM$(19), 降序排列)

' ' ; B B z z w u u t q - ← ↓ Z X T C 3 $

AM!: (直接排序 AM!(1)—AM! (10), 降序)

160 980 920 900 890 750 600 260 150 90 70

LN$: (直接排序 LN$(1)—LN$(10), 次序与 AM! 相同)

2g\@ }3@i %c/d vE*v j2,- 2koU A3! Nx7xU EJNI JLUW D6wk

IX%: (间接排序，改变了 IX%(0)—IX%(4))

1 5 3 2 4 0 0 0 0 0 0

ZC!: (间接排序，不变)

30 90 280 220 450 170

L$: (间接排序，不变)

6 ↓ ← " . j

ZC!: (用排序后的 IX% 作索引)

90 170 220 280 450

L$: (用排序后的 IX% 作索引)

↓ j " ← .

请注意上述排序后的 LN$ 数组，它的次序与 AM! 的排列次序是相同的，例如，现在的 AM!(1) 是原来的 AM!(8), 所以现在的LN$(1) 也对应原来的LN$(8); 现在的AM!(2) 是原来的AM!(10), 所以现在的 LN$(2)对应原来的 LN$(10);……依次类推。而 LN$(1)—LN$(10) 在排序后本身是无序的。

另外，间接排序后用 IX% 索引 L$ 数组时，其次序也对应 ZC! 的次序，例如：IX%(0)=1, 所以 ZC!(1)=90, L$(1)="↓"; IX%(1)=5, 所以第二个元素是 ZC!(5)=170, L$(5)="j";...依次类推，虽然 ZC! 在间接排序后用 IX% 索引时是升序排列的，而 L$ 是无序的。

7.22 RENEW 恢复用 NEW 命令删除的程序命令格式：

RENEW

这个 NEWDOS/80 BASIC 直接命令的功能是使得刚用 NEW 命令在表面上删除了的 BASIC 程序文本得以恢复。RENEW 所做的全部工作是：置文本区的第一个字节为非零，重建文本的向前队列指针并执行 CLEAR。于是，以前的程序就在文本区中得以恢复，并可以用于编辑和执行。但是，如果在执行 NEW 命令以后，曾建立或装入了哪怕一条文本行，那末以前的文本就不能恢复。此外，如果在调用这一 BASIC 功能时，文本区中从未有过文本，那末 RENEW 仍然认为文本区中存在着文本，并企图恢复它，这对 BASIC 来说其后果将是非常严重的。

第八章　磁盘BASIC的文件处理（I/O扩充和差别）

8.1 前言

本章讨论NEWDOS/80中BASIC的文件处理与NEWDOS/21、TRS-80 I型机的TRSDOS2.3和TRS-80 Ⅲ型机的TRSDOS1.3所提供的文件处理之间的差别和重大的扩充。7.2节所作的那些说明也适用于本章。

搞懂这些文件处理(I/O)的扩充远比使用它们困难得多，就象电学一样，每个人都使用电但不见得懂得很多。总之，这些扩充将使BASIC中文件处理(I/O)的编程更容易，但用户必须记住，由于TRSDOS不具备这些扩充，因此，使用这些扩充的程序就再也不能在TRSDOS上运行了。

本章的目的是想详细说明这些扩充，第九章和第十章含有文件I/O操作的补充讨论和例子。如果在本章与第九章和第十章之间存在冲突，则以本章为准。

本章使用的许多术语在附录D的“词汇表”中都有定义，用户需要时可参考它。对于涉及TRSDOS BASIC的地方，其详细内容请参考有关的手册（计声编译，《微计算机实用手册》，海洋出版社，1982年7月）。若对于NEWDOS/80 BASIC中的扩充部分有不明白的地方，可阅读第九章和第十章的相应内容。

8.2 文件类型

在以前的TRSDOS BASIC中，磁盘BASIC文件的类型有两种，即现在称为输出/输入print/input)文件的顺序文件，和现在称为字段项(field item)文件的随机文件，在NEWDOS/80 BASIC中，除上述两种类型外，又增加了另外两个文件类型：一个是标记项(marked item)文件，它有三个子类型，即MI，MU和MF。另一个是固定项(fixed item)文件，它有两个子类型，即FI和FF。

输出/输入（顺序）磁盘文件和字段项（随机）磁盘文件在TRS-80 I型机的TRSDOS手册（计声编译，《微计算机实用手册》，海洋出版社，1982年7月）和TRS-80 Ⅲ型机的TRSDOS手册中都有详细说明。希望读者在学习NEWDOS/80的这一章和使用它进行文件处理以前，首先阅读以上的相应章节。如果需要，则应运行一些试验程序以增进了解。

在NEWDOS/80 BASIC中，输出/输入文件与TRSDOS BASIC的顺序文件完全一样，而且在某些情况下还可使用GET, PUT语句进行文件I/O；字段项文件与TRSDOS BASIC的随机文件相比又有新的扩充：（1）在TRSDOS中，它只能随机存取，现在还可以顺序存取；（2）在记录长度上，与TRSDOS随机文件一样，字段项文件的所有记录都具有相同的长度，但是在TRSDOS中只能使用标准记录长度（256字节），而在NEWDOS/80中，记录长度可以为1到256个字节。

对于字段项文件，如果记录长度不等于256，则启动BASIC的初始化命令中必须指定要分配的文件区域数，并且在这个数字后边必须带有字母V（见7.3节）。例如：

BASIC, 3V

使BASIC分配3个文件区，每个文件区有两个缓冲区，第一个缓冲区与FIELD语句一起使用，第二个缓冲区用作全扇区缓冲区。请记住，这个特殊的V后缀只用于想使用记录长度小于256的字段项文件（TRSDOS中称随机文件）这种情况，否则分配给每个文件区的这个额外的256字节就浪费了。对于记录长度小于256字节的字段项文件，NEWDOS/80中使用的OPEN语句与TRSDOS中有点不同，现在要用：

OPEN "R", fan, filespec 1 , lrecl

其中，lrecl就是指定的逻辑记录长度，它的值为1—256。

8.3 文件类型的差别

NEWDOS/80 BASIC的四种文件类型之间的基本差别如下：

输出/输入文件只能顺序使用；字段项文件、固定项文件和标记项文件都能顺序地或随机地使用。

输出/输入文件全部以ASCII字符格式存储，将所有的数字数据由二进制格式转换成十进制字符。字段项文件、固定项文件和标记项文件都以二进制格式存储数字数据，因此，通常节省了磁盘空间和数据转换时间。

输出/输入文件使用PRINT语句进行写操作，这是很麻烦的，因为它需要用5个字符序列；",";去分隔两个字符串项，即必须在两个字符串项之间插入一个逗号（字符串形式）作分隔符，以便在读入时能作为两个字符串项处理。例如：PRINT#1, A$;",";B$。而字段项文件、固定项文件和标记项文件是用PUT语句进行写操作的，它隐含文件项的分隔，不需要用户去插入分隔符。对于字段项文件，文件项按照FIELD语句处置；对于固定项文件，按照IGEL中指定的隐含的或明显的文件项长度处置；而对于标记项文件，则按照文件项标记处置。

输出/输入文件使用INPUT语句进行读操作，而字段项文件、固定项文件和标记项文件要使用GET语句。

字段项文件要求在执行PUT语句前先将数据送入记录缓冲区，这是通过RSET或LSET功能进行的，在数字值的情况下，还要用MKD$, MKI$或MKS$函数把数字值转换成字符串形式。这种转换对于输出/输入文件、固定项文件和标记项文件都是不必要的。

字段项文件要求在使用由GET语句从文件输入的数字数据以前，要相应地通过CVD, CVI或CVS函数把字符串形式的数字数据转换成数字值。这对于输出/输入文件、固定项文件和标记项文件是不必要的。

输出/输入文件允许任意大小的记录长度。字段项文件允许的最大记录长度为256。固定项文件和标记项文件允许的最大记录长度是4095字节。固定项文件中，每个记录的长度都是固定相等的。固定项文件有两个子类型：FI文件不分段成记录；FF文件分段成记录（关于固定项文件请参阅8.5节）。标记项文件也可划分成记录，记录长度可以不相等，每个记录及记录中的各文件项都以标记字节开始。标记项文件又分三个子类型：MI 文件不分段成记录，即所谓密集型，其中的分段只有用户自己知道；MF 文件分段成记录，每个记录的长

度是相等的；MU 文件分段成记录，每个记录的长度可不等（关于标记项文件请参考8.6节）。

在TRSDOS中，顺序文件是不能修改的；在NEWDOS/80中，除输出/输入文件和MI文件不能修改以外，其他的所有文件都可以修改。

输出/输入文件处理把字符串数据不作任何改变地传送到文件中，但是在由文件输入时将切去字符串项前边的空格。字段项文件中的字符串在有关的LSET或RSET操作过程中用必要的空格填满左边或右边。固定项文件中的字符串也将根据需要在右边用空格填满，使文件项达到指定的长度，或者在实际的字符串长度超过文件项允许的长度时，截去右边的超过部分。虽然标记项文件在字符串超过文件项允许的最大字符个数时，字符串右边会被截去其超过部分，但是字符串不够长时不会填充空格。除了这种切尾以外，标记项文件是四种文件中完全不变地由相应的BASIC变量传送字符串的唯一一个。

8.4 GET和PUT语句中的成分

在TRSDOS中，GET和PUT语句的一般格式是：

```
GET  fan[,rn]
PUT  fan[,rn]
```

其中，fan为文件缓冲区号，rn为要存取的记录号。

在NEWDOS/80中，GET和PUT语句的一般格式已改成如下形式：

```
GET  fan[,fp]  [,,igel/,  igelsn]
PUT  fan[,fp]  [,,igel/,  igelsn]
```

GET和PUT语句按两个不同的阶段以下列顺序执行：

（1）文件定位阶段

这个阶段进行文件定位，以确定GET、PUT进行数据传送的文件位置。文件中的位置是根据GET或PUT语句的第二个参数——文件位置参数(fp)设定的。

（2）数据传送阶段

把数据由内存送到文件（PUT语句）或从文件送到内存储器（GET语句）。

GET和PUT语句将分别在8.8节，8.9节以及9.4节，9.5节详细讲述。但是，在此以前必须首先定义GET和PUT语句中使用的三个术语，其中有一个是字段项文件（即TRSDOS随机文件）的GET和PUT语句中原有的（即fp），可是它在字段项文件中受到了很大的限制（只允许fp为rn形式），另外两个术语是新的，即igel和igel表达式。

1. fp 文件位置

参数fp用于确定文件区中的操作位置。对于每一个GET或PUT操作，文件就根据fp说明进行初始定位。fp为以下格式之一：

（1）null（空缺）

若REMRA（见8.10节）是有效的并且文件是记录分段的，那末空缺fp就使文件区往前定位于下一个记录，否则fp=null的功能与fp=＊一样。例如：

```
PUT 1,,1000
```

其中，1为文件区号，两个逗号即表示fp为空缺，1000为下面即将定义的igelsn。

（2）＊

保持文件区位置不变。对于记录分段文件，fp=＊ 不能用来从一个记录前进到下一个 记录。例如：

```
GET  1,*, 1000
```

(3)#

文件区重新定位于 REMRA（见8.10节）。这样就允许重新处理前边刚处理过的记录。若 REMRA 当前无效，则产生错误。例如：

```
PUT 1,#,1000
```

(4)$

文件区重新定位于 REMBA（见8.10节）。这就允许返回到前边用 fp=null、＊、#、$、rn 或 !rba 的 GET/PUT 的记录中的字节位置，重新进行处理。如果 REMBA 当前无效，则产生错误。例如：

```
GET 1,$,1000
```

(5)%

用于规定 GET 和 PUT 语句为伪 FIELD 功能。请参考8.11节关于伪 FIELD 语句的讨论和9.1节文件定位。

(6)&

用于把文件区缓冲区的内容强制地写入磁盘文件。详细内容请见8.9节 6 关于 PUT fan,&的讨论和9.1节文件定位。

(7)& &

用于把 EOF 写入文件目录，而保持文件的打开状态和保持文件位置不变。请参考8.9节 7 关于 PUT fan && 的讨论和 9.1 节文件定位。

(8)!rba

rba 是一个计算 RBA 的表达式，RBA 等于文件中所需的相对 字 节 位 置，范 围 为 0 到 16777,215。GET 或 PUT 就由 !rba 指定的文件位置 (RBA) 开始传送数据。如果文件是记录分段的，则采用 !rba 指定一个记录的开始位置。例如：

```
GET 1,!1357,1000
```

注意：fp=!rba 的功能很强，但使用不当时后果也是十分严重的！请参考 9.1 节文件定位。

另外，fp 的表达式中不能含有访问文件区的函数，例如 LOC, LOF 等。

(9)!%

除了把当前的 EOF 值作为 RBA 使用以外，其概念与 !rba 相同。例如：

```
GET  1,!%,1000
```

(10)!$ rba

将文件定位于相对文件位置 rba, 不进行数据传送。见8.8节 6 关于 GET 的讨论。例如：

```
GET  1,!$ 1354
```

(11)!$ %

除了把当前文件的 EOF 值作为 RBA 使用以外，其概念与 !$ rba 相同。例如：

```
GET  1,!$ %
```

(12)!#rba

使表达式 rba 作为新的 EOF 值。见8.9节 9 关于 PUT 的讨论。例如：

PUT 1,!#1354　将1345作为新的 EOF 值。

(13)rn

rn 是一个表达式，它得到一个范围在 1—32767 之间的整数，这代表文件中目标记录的记录号。文件区定位于记录的第一项的开头，这与 TRSDOS BASIC 的随机文件是一样的。该文件区必须用 m=I、R 或 D 打开，同时，如果指定 ft 参数（即用于固定项文件或标记项文件）时，则要用 ft=FF 或 MF。例如：

GET 1,30　读取记录号30的记录。

2. IGEL　项组表达式清单

IGEL 对应于一组文件项的表达式清单。一个 IGEL 是一串以分号结尾的，用逗号分隔开的一个或多个表达式，它们对应于从当前文件位置开始的连续的文件项，当前文件位置由 GET 或 PUT 的文件定位参数确定。如果在寻找分隔的逗号、结尾的分号或表达式的开头时，遇到了一个标记（即 EOL），那末就到下一个 BASIC 语句去继续寻找。IGEL 的目的是在处理标记项文件或固定项文件的 GET 或 PUT 语句执行过程中，作为一组文件项与一组 BASIC 变量或表达式之间的链使用。以下是一些 IGEL 的例子：

例1.　(30)LN$,(15)FN$,AM!,　DT#(X);

例2.　"3",　AN%,NM$;

例3.　(32)A$(X,Y),　B%(2+X),C!,　第一行。

E$,　K#,　FS$;　最后一行。

如果在处理 IGEL 时遇到错误，则错误行号将是处理 IGEL 的相应的 GET/PUT 语句的行号，而不是有实际错误的含有 IGEL 的文本行的行号（在 GET/PUT 语句通过 IGELSN 引用 IGEL 时）。

3. IGEL 表达式

一个 IGEL 的表达式是项组表达式清单(即 IGEL) 中的一项。对于 PUT 语句，IGEL 表达式指定了赋给当前文件项的值。对于 GET 语句，IGEL 表达式指定了一个变量，它接受当前文件项的值作为它的值。IGEL 表达式为以下格式之一：

(1)exp

(2)(len) exp

(3)(len)$　仅用于固定项文件

(4)(len)#

(5)空表达式

其中：

(1)exp

这种格式是 IGEL 表达式的主要部分。通常，exp 命名一个 BASIC 变量，但是在对标记项文件作 PUT 操作时，exp 几乎可以是 LET 语句右边的任何合法的东西。当 exp 是一个命名的变量时（无论是简单变量还是数组变量），我们极力建议在变量名的后边加上四种类型符号（即 $、%、!或#）之一，虽然这并不是必需的。例如，我们建议：

用A$,B%(X, Y), C!D#;

代替A,B(X,Y), C,D;

但是，这个建议不适用于作为下标的变量（例如上例中的X和Y）。

（2）(len)exp

这是一个用(len)作前缀的表达式，len本身是一个数值范围为0到255之间整数的表达式。(len) exp只能用于那些是字符串的IGEL表达式。

Ⅰ．对于标记项文件，len是PUT操作中送往文件的一串字符的最大数目，或者是GET过程中从文件那儿接收到的一串字符的最大数目。如果实际的字符数目小于这个值，那末就只有这些较少数目的字符被传送。对于标记项文件，可以随意用这种(len) exp格式代替字符串表达式的exp格式。但是，对于MF文件，建议使用这种(len) exp格式。

Ⅱ．对于固定项文件，IGEL中的字符串表达式必须使用(len)exp格式，由于len指定了一个字符串文件项所具有的或将具有的精确的字符个数。在PUT语句传送数据过程中，如果变量的字符串具有的字符少于len，那末将根据需要在文件项（不是变量）右边填充空格。如果变量字符串具有的字符多于len，那末就不把其右边多出来的字符传送给文件项。在GET语句传送数据过程中，则把len个字符由文件传送给字符串变量。

Ⅲ．使用(len)exp表达式的IGEL举例：

(30)LN$, (20)FN$, AN%, DP#, (2)CD$(X);

它规定参与传送的最大字符数目是：LN$为30个，FN$为20个，CD$(X)为2个。

（3）(len)$

这个表达式只是在固定项文件中合法。len表示要跳过的文件字节的数目。对于GET操作，则跳过指定数目的文件字节。对于在一个已经存在的记录上的PUT操作，则跳过并且不改变指定数目的文件字节；对于一个新记录的PUT操作，(len)$等同于(len)#，对此请参阅下面关于(len)#的说明。

例如，在下例的IGEL中，跳过了开始的10个字节，接着传送12个字节（AN%变量2个字节，ST$变量10个字节），再跳过17个字节，然后传送最后8个字节：

(10)$, AN%, (10)ST$, (17)$, DP#;

（4）(len)#

固定项文件中，对于GET操作，(len)#的作用与(len)$相同；而对于PUT操作，则把len个零字节发送给文件。对于标记项文件，在GET操作中，(len)#跳过当前文件项；而在PUT操作中，则发送给文件len个空字符（十六进制00的字符，即零字节）。例如：

(10)#, AN%, (10)ST$, (17)#, DP#;

（5）空表达式

空表达式只能用在标记项文件GET语句的IGEL中，一个空表达式的作用是跳过相应的文件项。例如，将下述的GET语句与GET语句的一般格式进行对照，可知该语句fp=空，其IGEL的第一个、第二个和第四个IGEL表达式为空表达式，因此在执行中将跳过第一、第二和第四个文件项：

GET 1,,,,, X!,,A$;

在IGEL处理过程中，如果遇到一个IGEL表达式所特有的错误，那末就把该表达式在IGEL中的位置放在错误信息的前边显示出来。例如，若第四个IGEL表达式是错误的，则将在错误信息前边放一个4。

8.5 固定项文件的特性

1. 含有多个（或一个也没有）文件项。

2. 每一个文件项的类型和长度由 GET 或 PUT 语句的相应的 IGEL 所决定，而不取决于文件本身。这是固定项文件与标记项文件之间的一个基本差别。

3. 一个文件可以细分成具有相同长度的记录。

4. 最大的记录长度为4095字节。

5. 一个记录的文件项的数目和特性唯一地取决于记录长度和读/写（GET 或 PUT）记录所使用的 IGEL。

6. 与一个固定项文件往来的 I/O 链由 BASIC 的 OPEN 语句建立，其中 ft=FI 或 FF。

7. 通过 GET 语句把固定项文件的文件项内容传送给 IGEL 指定的 BASIC 变量。

8. 在 IGEL 中指定的 BASIC 变量要通过 PUT 语句建立或替换固定项文件的文 件项。

9. BASIC 的 CLOSE 语句终止程序与固定项文件之间的 I/O 链。

10. 在相邻的文件项之间或一个记录的末尾与下一个记录的开头之间不留磁盘空间。

11. 在建立一个 FF 文件记录时，记录末尾的任何无用空间都用零字节填充。

8.6 标记项文件的特性

1. 含有多个（或一个也没有）文件项。

2. 一个标记项文件的文件项总是用一个控制（或标记）字节开始，其后边是多个附加的控制字节（或一个也没有），接着是多个数据字节（或一个也没有）。

3. 标记文件项具有以下格式，具体采用哪种格式取决于第一个控制（或标记）字节的十六进制值：

（1）80—FFH

若标记字节为 80—FFH 之间的一个值，则其后是 0—127 个字节的二进制字符串，字节数目等于此标记字节值减去 80H 的差。

（2）70H

这个标记字节是 SOR (Start Of Record——记录开始）字节。每一个 MU 文件（分段成记录长度不同的标记项文件）的记录都用这一项开始。

（3）00H

这是个填充项。必要时用于填满 MF 或 MU 文件记录。

（4）71H

这个标记字节表示下一个字节是第二标记字节（或次标记字节），这个次标记字节含有其后的二进制字符串的字节计数（128—255）。

（5）72H

这个标记字节表示其后两个字节是一个以补码表示的二进制整数，这是 BASIC 的内部表示格式。

（6）73H

这个标记字节表示其后四个字节是一个 BASIC 格式的单精度浮点数，其格式为：

Ⅰ. 三个字节的已归一化的绝对值尾数，形式为 .mmmmm，其中，各二进制位 m 在这

些字节中以上升的量级表示：

i) 字节之间　从左到右为最低有效字节(LSB)到最高有效字节(MSB)。

ii)字节内部　从右到左为位0到位7，即最低有效位到最高有效位。

归一化的尾数值是左满的，所以最高阶尾数的那一个二进制位总是1。由于又用这一位（MSB的位7）来作尾数的符号位：0为正；1为负，所以当尾数为正时，又将这一位置0，当尾数为负时，则保持这一位为1不变。

Ⅱ．第四字节含有以2为底的指数，并已偏移128。当该字节为0时，不论其他字节的内容是什么，该浮点数就是0。

例如：

十进制值7，可归一化为二进制形式.111×2^3，所以其二进制浮点数的内部存储格式为：

尾数：第一字节（LSB）等于0，第二字节为0，第三字节（MSB）等于96，即二进制的01100000。

指数：即第四字节等于128加3，即131。

若是十进制值－7，则除第三字节为224（二进制11100000）以外，其他字节都与上面一样（关于浮点数的BASIC存储格式，请参考计声编译，《微计算机实用手册》第一篇之八，海洋出版社，1982年7月）。

（7）74H

这个标记字节表示其后8个字节是一个二进制浮点数，其格式与上述项目类型“73H”一样，只是前7个字节是尾数，而指数是在第8字节中，这是BASIC的双精度浮点数格式。

4．文件可细分成记录，所有记录或者是等长的（MF文件），或者有不同的长度（MU文件）。

5．一个文件记录的最大长度是4095字节，它包括所有的记录控制字节、文件项控制字节和数据字节。

6．如果文件被分成不等长（MU文件）的记录，那末每一个文件记录由BASIC自动提供的SOR项开始。

7．文件中邻接的记录可以含有不同数目的文件项。这将出现在文件中具有多种记录类型的情况下。对于具有固定记录长度的文件，必须极力避免记录溢出。

8．文件记录中相对定位的文件项可以象类型一样，在不同的记录中不一样。这将出现在文件中具有多种记录类型的情况下。

9．一个与标记项文件往来的I/O链是由BASIC语句OPEN建立的，其参数ft＝MI，MU或MF。

10．通过GET语句把标记项文件的文件项内容传送到IGEL中指定的BASIC变量之中。

11．标记项文件的文件项要通过PUT语句按照IGEL中指定的BASIC变量和（或）BASIC表达式来建立。

12．BASIC的CLOSE语句终止程序与标记项文件之间的I/O链。

13．在标记项文件的相邻项之间或记录之间不留磁盘空间。但是，必要时将插入SOR和填充项。

8.7 OPEN语句

打开文件或建立文件 I/O 链的 OPEN 语句，在 TRSDOS BASIC 中为以下格式：

OPEN m, fan, filespec

其中，m 为指定的存取方式，它等于字符串值 I, O 和 R 之一，分别对应于顺序输入、顺序输出和随机存取。fan 为设置的文件区号，它等于1—15之间的整数。filespec 是要打开的用于文件处理的文件标识符。

在 NEWDOS/80 BASIC 中，由于支持四种文件类型和五种操作方式，另外，对字段项文件（TRSDOS 随机文件）还支持顺序存取，所以 OPEN 语句已改成以下一般格式：

OPEN m, fan, filespec[,ft] [,lrecl]

其中，ft 为指定的文件类型，lrecl 为指定的逻辑记录长度。

NEWDOS/80 磁盘 BASIC 的 OPEN 语句有以下四种具体格式：

(1)OPEN m, fan, filespec

(2)OPEN m, fan, filespec, len

(3)OPEN m, fan, filespec, ft

(4)OPEN m, fan, filespec, ft, len

其中：

1. 上述格式中 fan 和 filespec 的含义与 TRSDOS BASIC 是一样的。关于它们的定义，请参阅附录 D “词汇表”。四种格式的举例：

OPEN "I", 1, "XXX/DAT: 1"

OPEN "R", 2, "XXX/DAT", 128

OPEN "O", 1, "XXX/DAT: 0", "MU"

OPEN "D", 3, "XXX/DAT", "MF", 71

2. 上面的格式 1 就是 TRSDOS 中的格式，在 NEWDOS 中，它用于输出/输入文件和字段项文件。格式 2 用于字段项文件。格式 3 用于 FI、MI 和 MU 文件。格式 4 用于 MU、MF 和 FF 文件。

3. m

给文件区指定操作方式。它是一个表达式，其值是下列字符串之一：

(1) I

这个操作方式与 TRSDOS BASIC 一样，指定顺序输入方式。用于打开由 fan 指定的文件区，对该文件只进行顺序的输入操作（若没有指定 ft, 用 INPUT 语句；若指定了 ft, 则要用 GET 语句)。文件区被定位于文件的开头。

(2) O

这个文件存取方式也是与 TRSDOS BASIC 一样的，它指定顺序输出方式。如果由 filespec 指定的文件不存在，则建立该文件。打开由 fan 指定的文件区，对该文件只进行顺序输出操作（若没有指定 ft, 用 PRINT 语句，若指定了 ft, 则用 PUT 语句)。若指定文件已经存在，则令 EOF=0，于是丢失了原有的文件内容，文件区定位于 EOF（即定位于文件的开头)，以进行顺序输出。

(3) E

这是 NEWDOS/80 BASIC 中新增加的一种存取方式，它的功能基本上与 "O" 相同，只是保持原来的 EOF 不变。由此允许在一个已经存在的顺序文件上添加新的内容。

（4）R

这个存取方式也与 TRSDOS BASIC 相同，用于指定随机存取方式。如果由 filespec 指定的文件不存在，则建立该文件。打开由 fan 指定的文件区，对文件进行 GET 或 PUT 操作。如果指定的文件已经存在，则保持原有的 EOF 不变，文件定位象 "I" 那样定位于文件的开头。如果接着的 PUT 操作所指定的记录在 EOF 处或超过 EOF，文件就被自动地扩展以容纳该新的记录。

（5）D

这是 NEWDOS/80 BASIC 新增加的存取方式，它的功能基本上与 "R" 相同，只是由 filespec 指定的文件必须已经存在，并且对位于 EOF 处或超过 EOF 的记录进行 PUT 操作时，将作为错误处理，即不允许象"R"那样超过EOF进行文件存取。

4. ft

ft 参数在 OPEN 语句中用于指定文件类型。它可以是一个表达式，其值为下列字符串之一：

（1）FI

一个记录不分段的固定项文件。此时在 OPEN 语句中不可指定 len。

（2）FF

一个记录长度固定的固定项文件。对于 FF 文件，必须在 OPEN 语句中指定 len。

（3）MI

一个不分段成记录的标记项文件。对于 MI 文件，不可指定 len。MI 文件中的文件项不能修改。

（4）MU

一个分段成不同长度的记录的标记项文件，其中，记录长度由 SOR 项到 EOF（或下个记录的 SOR 项）之间的字节数决定。在 OPEN 语句中，len 是任选的。如果指定 len，则把它作为 MU 文件的记录所允许的最大长度。一个 MU 文件的记录是可以修改的，只要记录长度不增加到超过原先的长度值。如果修改后记录被缩短了，则用填充项填满到原先的长度。

（5）MF

一个分段成固定长度的记录的标记项文件。若在 OPEN 语句中指定 MF 文件，则必须指定 len。

5. 如果指定 ft 参数，则用下列规则：

（1）如果 GET 语句现在是将数据由文件传送给 BASIC 变量，那么在 GET 语句中必须指定 IGEL 或 IGELSN。

（2）如果 PUT 语句现在是把数据由 BASIC 变量或表达式传送给文件，那么在 PUT 语句中必须指定 IGEL 或 IGELSN。

（3）不可使用 BASIC 语句 FIELD。

（4）程序不可修改文件区的 I/O 缓冲区中的信息，并且不可信赖该缓冲区中的值或 FCB 的 LRECL, NEXT 或 EOF 字段中的值。

6. 如果不指定 ft，并且 m=D 或 R，则用下列规则：

（1）该文件是一个字段项文件（随机文件），除非另有说明，其规定与 TRS-80 I 型机的 TRSDOS 2.3 是相同的。

（2）必须使用 FIELD 语句将 BASIC 变量适当地覆盖在文件区的缓冲区中。虽然，记录中定义的任何一个字符串的长度不能超过 255，但 FIELD 能处理 256 个字节的记录。FIELD 语句定义的字节数通常等于 len，它不应超过 len 并且不能超过 256。

（3）GET 和 PUT 语句中不能指定 IGEL 或 IGELSN。

（4）如果在 OPEN 语句中不指定 len，则假设 len 等于256。

（5）如果在 OPEN 语句中指定 len，则 len 必须是 1 到256之间的值。如果 len 小于 256，那么启动 BASIC 时必须明确指定带有后缀字符 V 的文件区数（见7.3节）。

7. len

OPEN 语句中，len 用于指定文件的记录长度，它是一个表达式。对于字段项文件，其值为 1 至 256 之间的一个整数；对于固定项文件和标记项文件，其值为 1 至4095之间的整数。对于字段项文件、FF 或 MF 文件，len 是文件记录的标准长度。对于 MU 文件，len 是文件记录所允许的最大长度。现在，文件的 FPDE 不携带正确的 len (LRECL) 值，所以总是使用 OPEN 语句所提供的 len 值（明确的或隐含的）。对 len 的检查是在 GET 或 PUT 操作时进行的。对于 MF 和 MU 文件，用户在计算长度时，必须考虑下列额外的字节：

（1）每项加一个字节（主项控制字节）。

（2）对于实际内容多于 127 个字符的每一个字符串加一个字节。

对于 MU 文件，用户还必须考虑在每一个记录开头的 SOR 项字节。

分配给带标记的文件项的字节数，等于标记（或控制）字节数（1 或 2）加 BASIC 用来容纳字符串或数字值的字节数：

（1）字符串：一个或两个标记字节加实际的字符串长度（要考虑到表达式前缀引起的截尾）。第二标记字节只在字符串长度大于127字节时使用。

（2）整数：一个标记字节加 2 个字节。

（3）单精度浮点数：一个标记字节加 4 个字节。

（4）双精度浮点数：一个标记字节加 8 个字节。

对于固定项文件，分配给每个文件项的字节数由 IGEL 决定：

（1）字符串：包括(len) $ 和(len)#，这个字节数目由表达式前缀指定。

（2）整数：2 个字节。

（3）单精度浮点数：4 个字节。

（4）双精度浮点数：8 个字节。

8. 如果 FCB 中的 EOF 被 OPEN 修改了，即使没有执行过 PRINT 或 PUT 语句，那么后边的 CLOSE 或 PUT fan, && 语句将在 FPDE 中把 EOF 改成新的 EOF。

8.8 GET 语句

在8.4节已讲过，TRSDOS 中 GET 语句的一般格式为：

GET fan [, rn]

这个语句从磁盘文件中读出数据记录并放入指定的文件缓冲区内。而此文件缓冲区已事先用 FIELD 语句划分好变量字段，所以数据记录读入缓冲区就赋给了相应的各变量，并可供程序使用。在 NEWDOS/80 磁盘 BASIC 中，可以不必用 FIELD 语句划分字段，而直接使用 IGEL 与文件交换数据，所以 GET 语句已被修改成以下一般格式：

GET fan [,,fp] [,,igel/, igelsn]

其具体格式有以下四种：

(1)GET fan (fp 为空)

(2)GET fan, fp

(3)GET fan, fp, IGELSN

(4)GET fan, fp,, IGEL

其中：

1. 语句格式中，fan 为 OPEN 语句中指定的文件区编号，文件存取将通过这个文件区进行。fp 是文件位置，GET 操作将在这个位置上进行。IGEL 是接收文件数据的项组表达式表。IGELSN 是 IGEL 所在的语句行号。以上的所有术语的详细定义，可参考 8.4 节和附录 D。

上述四种格式的 GET 语句的例子有：

```
GET 1
GET 1,30
GET 1,!X,1000
GET 1,,,X%,Y!,Z#,(20)A$;
```

2. 在成功地执行完 GET 语句后，文件区靠左定位于：

(1)对标记项文件操作，定位于文件的下一项。

(2)对固定项文件操作，定位于文件的下一字节。

(3)对字段项文件操作，定位于文件的下一个记录。

3. 若遇到 EOR 或 EOF:

(1)对于字段项文件操作，文件区缓冲区被置成二进制零，因此对随后的所有的数据访问都给出二进制零值。不发生错误。

(2)对于标记项文件和固定项文件的操作，则发生错误。

4. 如果在 GET 操作过程中遇到错误，那末文件区控制数据就恢复成 GET 语句执行之前存在的状态，在 IGEL 或 FIELD 中命名的变量的结果值不定。在改正错误以后，可再执行该语句。

5. 如果在 GET 语句中指定 IGEL 或 IGELSN，则相邻的文件项就被赋给 IGEL 相邻命名的变量。

对于标记项文件操作：

(1)若 IGEL 表达式是空的，则跳过相应的文件项。

(2)IGEL 表达式前缀可用来限制字符串变量的字符数目。如果文件项的字符较少，则变量的字符串长度就置成此较小的值。如果文件项的字符较多，就跳过其右边多余的字符而不把它们传送给变量。

(3)在遇到填充项时就跳过它们。

(4)若命名的变量与文件项在类型上不兼容，则发生 TM（类型失配）错误。

(5)对于记录分段的文件，在第一项的 GET 操作后边，可以跟随对其余项的 PUT 操作。

(6)对于记录分段文件，如果 GET 操作发现记录中没有足够的文件项，则发生 RECORD

OVERFLOW（记录溢出）错误。

（7）除了表达式前缀的限制作用以外，字符串按原样由文件传送给变量，并且保留前边的空格（若字符串前边有空格）。

对于固定项文件操作：

（1）对于每一个命名的字符串变量，表达式前缀中指定数目的字符就从文件中传送到字符串区。

（2）对于记录分段文件，如果 GET 语句发现记录中没有足够的字节，则发生 RECORD OVERFLOW（记录溢出）错误。

（3）对连续的数据，GET 和 PUT 操作可以一个接一个进行，只要求：

Ⅰ．对记录中的当前位置，用户要始终做到心中有数。

Ⅱ．对记录分段文件，要注意观察记录的边界。

对于标记项和固定项文件：

记录项的输入可以分散在两个以上的 GET 语句中进行。

6．以下形式的 GET 语句：

```
GET  fan,!$rba
GET  fan,!$%
```

允许用户在该文件区为下一个 GET，INPUT，PUT 或 PRINT 语句进行文件定位。这个 GET 语句不进行数据传送。

!$rba 表示将文件定位于该 RBA 值。

!$% 表示 EOF 的当前值，并作为 RBA 值使用，将文件定位于 EOF。这种形式的语句把 REMRA 和 REMBA 标记为无效。

例如：

```
GET  1,!$2550   将文件定位于 RBA 2550。
GET  1,!$X   将文件定位于变量 X 中的 RBA 值。
GET  2,!$%   将文件定位于 EOF。
```

8.9 PUT 语句

前面8.4节中已讲过，在 TRSDOS 中，PUT 语句的一般格式是：

```
PUT  fan  [,rn]
```

该语句的功能是把数据从文件缓冲区传送到文件的指定位置。与 GET 语句一样，必须事先用 FIELD 语句划分好缓冲区，并用 LSET 和 RSET 语句把数据放入缓冲区中相应的变量字段。

在 NEWDOS/80 BASIC 中，由于可以在 PUT 语句中指定 IGEL 直接与文件交换数据。所以，NEWDOS/80 磁盘 BASIC 的 PUT 语句已修改成以下一般格式：

```
PUT fan [,fp] [,, IGEL/, IGELSN]
```

其具体格式有以下四种：

（1）PUT fan (fp=空)

（2）PUT fan, fp

（3）PUT fan, fp, IGELSN

(4)PUT fan, fp,, IGEL

其中：

1. 语句格式中各参数的定义与 GET 语句中一样：fan 是打开文件时 OPEN 语句中指定的文件缓冲区号码；fp 是进行操作的文件位置；IGEL 是传送数据的项组表达式表；IGELSN 是 IGEL 所在的语句行号。详细可参考 8.4 节和附录 D "词汇表"。这四种格式的 PUT 语句举例：

```
PUT  2
PUT  1,X
PUT  3,,1000  (本例空缺 fp)
PUT  1,RN!,,(20)A$,B%,C!,D#;
```

2. 在成功地执行完 PUT 语句后，与 GET 语句一样，文件区被靠左定位于相应文件位置（请参考 8.8 节之 2）。

3. 如果在 PUT 处理过程中遇到错误，那么文件区的控制数据被恢复成执行 PUT 语句之前存在的状态。文件中的结果数据不定，并且可能导致其后的 GET 语句发生错误。但只有在修改已经存在的记录时，才可能产生这个问题。在改正错误以后，无论如何应该接着对那个记录再执行一个 PUT 语句。为了减少文件遭受破坏的机会，在文件被打开(m=R或D)时，应对 IGEL 全面地处理一次，以发现非 I/O 错误，然后再进行实际的文件修改。

4. 如果 PUT 语句中指定 IGEL 或 IGELSN，则将连续的 IGEL 表达式的值发送给文件的连续的文件项。

对于标记项文件操作：

(1)在必要的时候，将在文件中自动地插入 SOR 项和填充项。

(2)除了访问文件区的函数以外，IGEL 表达式可以是 LET 语句等号右边的任何合法的东西。

(3)除了 IGEL 表达式前缀的限制作用以外，将结果字符串按原样送入文件。

(4)把数目字或表达式按 BASIC 数字类型送入文件，它们在 BASIC 内部进行转换。

(5)对于固定长度的记录和被修改的可变长度的记录，每执行一个 PUT 语句就替换了记录的一部分，即从 PUT 操作的文件位置起到该记录的末尾，必要时将使用填充项。

注意！记录中相对位置比 PUT 语句写入的最后一项高的那些旧文件项就丢失了，这是由于从 PUT 的最后一项之后到记录末尾的该记录的全部磁盘空间现在已含有填充项。

(6)一个记录的字节总和（用于标记、数字数据和字符串的字节总和）的最大理论值可以超过 len（在 OPEN 语句中定义），只要记录的 PUT 过程中实际使用的字节数不超过 len。

对于固定项文件操作：

对每一个字符串变量，把所需要的表达式前缀中指定数目的字符从变量传送到文件，必要时将在右边添加空格或截尾。

5. 对于标记项和固定项文件

(1)可以用两个以上的 PUT 语句来进行一个记录的文件项的输出。

(2)把数据送入文件缓冲区以后，在下列情况之一发生以前，是不实际写入磁盘的：

Ⅰ. 关闭文件区。

Ⅱ. 需要用缓冲区容纳文件的另一部分数据。

Ⅲ． 执行一个“PUT fan, &”或“PUT fan, &&”语句。

（3）如果超过可允许的记录长度，则发生 RECORD OVERFLOW(记录溢出)错误。

（4）关于数字文件项使用的字节数的讨论，见 OPEN 语句（8.7节之7）。

6． 以下格式的 PUT 语句：

PUT fan, &

如果文件缓冲区含有的数据尚未写入磁盘的话，这个 PUT 语句允许用户强制地将文件缓冲区写入磁盘。如果文件缓冲区中没有数据，则忽略这个语句。用户必须记住，假如在此缓冲区中还有更多要写的数据的话，对于标记项文件、固定项文件和字段项文件(其中 len 小于256)，数据实际写入磁盘未必在 PUT 时进行。“PUT fan, &” 强制地把这些悬而未决的数据输出到磁盘上，并在以下任何一个条件存在时使用：

（1）在重新使用文件区以前，但用户不想执行 CLOSE 语句的某些情况下。

（2）与其他文件区的交互作用取决于磁盘上的悬而未决的数据时。

（3）数据是非常重要的。

文件区的文件定位不受 PUT fan, & 功能的影响。

例如：PUT 3,&

7． 以下格式的 PUT 语句：

PUT fan, &&

允许用户强制地将当前在文件区的控制数据中的 EOF 写入目录。这个特殊的 PUT 语句可免除用户执行以下必要的功能：一个 LOC (fan)!, 以了解当前的文件位置；一个 CLOSE，把 EOF 写入目录；一个 OPEN，重建文件的 I/O 链；以及一个进行文件定位的 GET 或PUT，以便把文件区定位到它原先所在的位置。在把 EOF 实际写入目录以前，“PUT fan, &&” 功能先执行 “PUT fan, &” 功能，把数据写入磁盘。文件区的文件定位不为 PUT fan, && 功能所改变。

例如：PUT 2,&&

8． 以下格式的 PUT 语句：

PUT fan, !$RBA

PUT fan, !$%

它们的功能与 GET 语句中相同，用于为后续的操作定位文件，而不进行数据的传送。详细内容请参考8.8节6。

9． 以下格式的 PUT 语句，

PUT fan, !#rba

使得文件的 EOF 被置成表达式 rba 的值。rba 求得的值必须是一个 RBA。对于该文件区来说除此之外无任何改变。在执行本语句以后，必须执行一个 CLOSE 语句或 PUT fan, && 语句，以强制把新的 EOF 写入文件的 FPDE。

例如：PUT 2, !#2000

把文件区 2 的控制数据中的 EOF 置成 2000。

8.10 REMRA 和 REMBA

在每个文件区的控制数据中，BASIC 存放了两个额外的相对的文件位置值：

(1)REMRA　记忆的记录地址

(2)REMBA　记忆的字节地址

其中：

1. 使用 REMRA 的场所是：

(1)在 GET 或 PUT 语句具有 fp=# (请参阅8.4节) 时，用于定位文件。

(2)在LOC (fan) $ 、LOC(fan)! 和 LOC(1)# 等函数中使用 (见8.12节)。

2. 使用 REMBA 的唯一场所是在 GET 语句或 PUT 语句具有 fp= $ (见 8.4 节) 时定位文件。

3. REMRA 和 REMBA 都是 RBA 格式。

4. 每一个 OPEN 语句和每一个带有 fp=!$ RBA 或 !$ % 的 GET 语句或 PUT 语句都标记 REMRA 和 REMBA 为无效。

5. 每个 INPUT 和 PRINT 语句把 REMRA 置成语句开始执行时的文件位置。输出/输入文件操作不使用 REMBA。

6. 每个带有 fp=null、rn、!rba、 !% 或 * 的 GET 语句或 PUT 语句(* 只能用于 REMRA 在语句开始位置是无效的或文件不是记录分段的时候) 把 REMRA 置成该 fp 值得到的文件位置。

7. 每个带有 fp=null,rn, !rba, !% 或 * 的 GET 语句或 PUT 语句把 REMBA 置成该fp值得到的文件位置。

8. 不要让 REMRA 和 REMBA 的概念把你迷惑住了。如上所述,使用 REMRA 的场所只有两个 (在 fp=# 时和用于 LOC 函数时)，而使用 REMBA 的地方只有一个 (在 fp= $ 时)。如果你不使用部分记录 I/O，那么 REMRA 和 REMBA 总是相同的。最通常的用法是用于对刚读的记录执行 PUT (带有 fp=#) 操作。

8.11 伪 FIELD 功能

对于固定项文件和标记项文件，用于字段项文件 (TRSDOS 随机文件) 的 FIELD 语句是不合法的。然而存在这种情况，即用户想把与 IGEL 有关的字符串置成它们指定的长度，并且对它们仍然使用 LSET 和 RSET 语句的方法。虽然用户可以使用 STRING $ 函数来进行这项工作，但 NEWDOS/80 提供另一个方法，即使用具有下列格式的伪 FIELD 功能：

(1)GET　fan,%,IGELSN

(2)GET　fan,%,,IGEL

(3)PUT　fan,%,IGELSN

(4)PUT　fan,%,,　IGEL

其中：

1. fan 是文件缓冲区号，IGELSN 是含有 IGEL 的语句行号，IGEL 是项组表达式。它们都已经在 8.4 节讲过，词汇表中也有它们的定义。

2. 只是文本格式约定才需要 fan 说明。无论文件区是否打开或为什么打开文件，都与此伪 FIELD 功能无关，这个功能只与 IGEL 有关，并且丝毫不改变文件区。

3. 对 IGEL 的处理：

(1)数字变量保持不变。

(2)跳过(len)$和(len)#形式的表达式。

(3)字符串变量在IGEL中必须有前缀(len)。

(4)赋给字符串变量的长度等于IGEL表达式前缀，并按需要在其右边添加空格或截尾。除了填充空格或截尾外，字符串的内容不变。但是，如果字符串当前不在字符串区域中，则将它们送到那儿。接着，就可以使用LSET和RSET把数据送入这些字符串中。

4. 举例

```
PUT 2,%,, IX%,(30)A$,DP#,(10)B$;
```

这个伪FIELD功能使字符串A$和B$成为长度分别为30和10个字符的字符串，必要时在右边添加空格或截尾。没有数据被送入文件，并且文件位置不变。

8.12 LOC函数

TRSDOS BASIC中也有LOC函数，但是NEWDOS/80中新增加了许多功能。在NEWDOS/80磁盘BASIC中具有如下定义的一些LOC函数：

1. LOC(fan)

其中，fan是一个文件区号，其值为1—15，这个文件区是为字段项文件、MF或FF文件的操作而打开的。这个函数得到一个整数(1—32767)，它是前面对该文件区域进行GET或PUT处理的记录号。若它的值等于0，则表示没有对任何记录进行过GET或PUT操作或者REMRA无效。这个格式的LOC函数基本上与TRSDOS BASIC中一样。以下格式2到格式5是NEWDOS/80 BASIC新增加的。

例如：

```
PUT 1,34: X=LOC(1)
```

变量X将得到LOC函数的值34，此即前面PUT操作的记录号。

2. LOC (fan)$

对于记录分段文件，如果下个记录的开始位置(若REMRA有效)或当前文件位置(若REMRA无效)大于或等于EOF，则这个函数得－1(即IF语句为真)；如果小于EOF，则得0(即IF语句为假)。对于非记录分段文件和输出/输入文件，如果当前文件位置大于或等于EOF，则此函数得－1；如果小于EOF，则得0。LOC(fan)$函数与EOF函数的不同在于EOF函数仅测试是否正处在EOF。

例如：

```
IF LOC(1)$ THEN END
```

如果下个记录位于或超过文件的EOF，则结束程序的执行。

3. LOC (fan)%

这个格式的LOC函数得到等于文件的EOF的RBA。例如，假设文件含有3142个字节：

```
X=LOC(1)%
```

将在X中得到数值3142。

4. LOC (fan)!

对于记录分段文件，这一函数得到一个RBA值，它等于：

(1)若REMRA有效，则为文件的下一个记录的位置。

(2)若 REMRA 无效，则为当前文件位置。

对于非记录分段文件和输出/输入文件，这个函数得到的 RBA 值为当前文件位置。

例如，如果在文件区 1，最后一个完全或部分处理的记录开始于相对文件位置 1667，而下一个记录开始于相对文件位置1701，那么：

```
X=LOC(1)!
```

将使 X 等于 1667。

使用 LOC (fan)! 和（或）LOC(fan)## 可使程序得到一组项目（非记录分段文件）或一个记录(记录分段文件)的文件位置，请记住，在以后作进一步应用时，可以通过 fp=!rba 或 fp=!$rba 将文件重新定位于那个文件位置的数据。这就允许用户去建立索引所有文件类型的索引文件，以进行随机存取。

8.13 I/O 错误的纠正

NEWDOS/80 磁盘 BASIC 语句 PRINT、PUT、INPUT 和 GET 的操作已被修改，以致在语句处理过程中发生错误时，文件区控制数据保持不变。这允许用户或程序员在发生错误时有更多的选择余地。例如：

1. 程序正在给一个顺序的输出/输入文件进行输出时，发生了 DISK FULL（磁盘已满）错误。这时得到的 EOF 仍然是该语句开始时它所在的位置，然后可以关闭此文件；如果在该驱动器上没有打开别的文件，则可以装配另一个盘片，以相同的文件区域打开一个新的文件，然后再执行发生过错误的语句，继续进行处理。于是，以后的输入处理可以对两个文件进行处理，它使用第一个文件上的 EOF 来启动向第二个文件的转移。

2. 程序使用两个以上的 PUT 语句正在把单个记录输出给 MU 文件时，在当前记录的第二个 PUT 中发生 DISK FULL（磁盘已满）错误。EOF 被复位于该错误语句开始执行时它所处的位置，而不是记录的开始。在转向一个新文件以前，必须通过以下两个语句将EOF置回到记录的开始：

```
X!=LOC(fan)##: PUT fan, !##X!
```

然后，可以关闭该文件区，装配一个新的盘片，再打开文件区，并回到记录的开始（而不是 PUT 的开始）继续处理，由于 MU 文件总是必须以 SOR 项开始，如果两个 MU 文件连在一起使用，则第一个文件不能以部分记录结束，并期待下一个文件含有记录的其余部分。

注意，当在原先的驱动器上除打开此错误文件区外，还打开了更多的文件时，用户或程序员在一个驱动器上交换盘片或将给定盘片换到另一个驱动器上时，必须极其小心。

用户也要记住，虽然文件区控制数据被复原成语句开始时它所具有的值，但是与PUT有关的文件数据是不定的，并且在 GET 中接受数据的变量的内容也是不定的。

为便于纠正错误和平时的编程，BASIC 使用一个独立的控制区域来执行 GET、PUT 或其他有关的文件区的操作，在操作正确完成以前，文件区的控制数据保持不变。但是，在 NEWDOS/80 中，只有一个暂时的控制区域，因此，文件区的功能不能嵌套在另一个使用文件区的功能之中，即使文件区是相同的也如此。例如，上面给出的两个语句不能如下那样合并在一个语句中：

```
PUT  fan,!##LOC(fan)##
```

8.14 有关 NEWDOS/80 磁盘 BASIC I/O 的一些说明

1. 对于标记项文件和固定项文件，用户一次只能用 GET 或 PUT 处理一个数据项组。对传送的数据总数的唯一限制是文件的大小，以及记录的大小（如果可用的话）。逻辑记录的长度可以是 1 到4095字节之间的任意长度。用户绝对不应该去访问文件缓冲区，因为其内容在任何时候都是不可预测的。

注意！如果用户打开文件的目的不是进行字段项操作（其中 FIELD 语句是合法的），而是进行其他操作，在这种情况下，如果改变文件缓冲区中的数据，那么其后果是不堪设想的。用户必须尽最大努力来避免文件遭受破坏，例如，在某种情况下已合法地使用了 FIELD 语句，然而它对于准备存取的文件区是不合法的，如果用户继续使用 RSET 或 LSET 给用于该文件区的由 FIELD 定义的字符串变量赋值，并通过该文件区去存取文件，那末就会破坏文件。

2. 为字段项文件操作特设的那些函数（MKD$、MKS$、MKI$、CVD、CVS、CVI、LSET 和 RSET）的作用和以前一样。但是，对于标记项和固定项文件操作，可以放弃使用这些函数，因为 GET 和 PUT 语句可以象传送字符串数据一样传送数字数据。

3. 对于使用 IGEL 或 IGELSN 的 GET/PUT 语句，用户必须记住，在 IGEL 处理过程中检测到的任何错误，都作为含有 GET 或 PUT 语句的文本行而不是实际的 IGEL 的文本行发生的一个错误被显示出来。

4. 为便利检测使用 IGELSN 的 GET 或 PUT 语句的错误，GET 或 PUT 语句应是文本行上的唯一语句。

5. 只有在文件是以 R 或 D 方式打开的时候，文件才能被修改。MI 文件和输出/输入文件虽然可以往上添加，但也不能修改。MU 文件记录可以修改，只要新记录长度不超过原记录长度。MU 文件的最后一个记录可以不受这种限制而进行扩展。

6. 对于为输出/输入文件打开的文件区，如果 fp 参数是!$ rba, !$ %, !#rba,&,&&或%，则可以执行 GET 或 PUT 语句。

7. 使用 fan 的 BASIC 函数（即 EOF, LOC, LOF 等）不能存在于 IGEL 中或 OPEN, GET, PUT, CLOSE, PRINT（往磁盘写）或 INPUT（从磁盘读）语句中。这是 NEWDOS/80 的一个限制（在 TRSDOS 中不存在）。

8. 对于记录可以跨两个以上磁盘扇区的磁盘文件（记录长度不是标准的 256 或不能将256除尽的文件），当一个记录或项目组实际被分在两个或更多的文件扇区上时，实际的磁盘 /O 次数就增加200%（与记录长度是标准的或可以整除256的文件相比较）。在磁盘 I/O 中增加的总百分比大约为（LEN/256)*200，其中，LEN 是所处理的记录或项目组的平均长度，并且 LEN<256。对于 LEN>256 的情况，在此不给出近似值。

第九章 文件处理的进一步讨论

本章对第八章中讲述的内容，特别是关于标记项文件和固定项文件的操作，再作一些补充，以有助于读者对这两种文件的深入理解和使用。但是，本章讲述的内容不能代替第八章，所以读者在阅读这一章以前，应先学习第八章的内容，在学习完这两章以后，再认真研究一下第十章的应用举例，这样，你对标记项文件和固定项文件的使用就不会有什么困难了。

9.1 文件定位

文件位置(fp)是 NEWDOS/80 BASIC 的所有 GET 和 PUT 语句中的一个操作数（已在8.4节作了说明)。当省略 fp 时，就假设为空操作数。否则，fp 操作数通常由一个特殊字符组成，其后有时跟一个别的字符和（或）表达式。有一种格式的 fp 操作数只由一个数字表达式组成。在它所遵循的这些格式中，使用的特殊字符如表 9.1 所示。在那些一个特殊的前缀字符后连接着一些其他字符串的格式中，特殊字符不必与表达式的其余部分邻接，它可以用一个空格或空间隔开。

表9.1 文件定位参数

fp值	意 义
空缺	如果文件是一个 MU、MF 或 FF 型的文件，并且 REMRA 是有效的，则空缺的fp值使文件往前推进到下一个相邻的记录；在其他任何情况下，当前的文件位置不变，并且从前面一个GET 或 PUT 操作的数据传送结束时所离开的位置开始继续进行处理。OPEN（打开文件）操作使 REMRA 对于所有文件类型都被标记为无效，并令当前文件位置等于 0（除方式 E 以外，方式 E 令当前文件位置等于FPDE 的 EOF 值)。对记录分段文件的第一次顺序存取，总是从当前文件位置开始的。
*	当前文件位置不变。这个位置说明符允许 GET 或 PUT 操作连续处理一个特定的记录。它主要用于连续处理一个已经部分地读或写的记录。对于 MU、MF 和 FF 型文件，即使此文件实际上已经定位于该记录并用完了当前记录的字节，也不能用它来将文件推进到下一个相邻的记录。要顺序地推进到下个记录，请用空缺的 fp值。
#	如果 REMRA 是有效的，则把文件定位于当前记录并再次处理该记录；如果 REMRA 是无效的，则引起一个错误状态。对于 MU、MF 和 FF 型文件，这个 fp 值允许从头开始，并且多半在 IGEL 中用不同的变量名或表达式，重新处理当前处理的记录。对于 MI 和 FI 型文件，允许重新处理同一个数据项组，就象用前面的 GET 或 PUT 语句直接处理一样。
$	如果 REMRA 是有效的，则文件被定位于前面的 GET 或 PUT 操作结束其文件定位阶段时所处的位置，再开始处理；如果 REMRA 是无效的，则引起一个错误状态。这个 fp 值允许 GET 或 PUT 操作重新处理一个特定的数据组。它主要用于重新定位一个文件以进行部分记录 I/O。对于所有的 NEWDOS/80 文件类型，它以相同的方式操作。
%	这个 fp 值执行一个 "伪FIELD" 操作。这个伪 FIELD 操作不进行数据传送；文件区的 FCB 不变。当使用这个 fp 值时，不必打开文件。它用于 FF 和 FI 文件，以便在高端内存由 BASIC 字符串存储区域来分配固定大小的用户数据串。
&	这个 fp 值仅用于 PUT 操作，对文件定位没有影响。然而它使文件缓冲区的当前内容写入磁盘。当缓冲区中的数据对用户特别关键时应使用这个 fp 值。它可以用于指定

续 表

fp值	意 义
	一个PRINT（输出）文件的FAN（文件区号）。
& &	这个 fp 值与 & 相似，不同的是它根据 FCB 修改了 FPDE 中文件的 EOF。PUT fan, & & 语句允许程序员不必进行 CLOSE 操作而强制修改 FPDE 的 EOF。
!rba	使用这种形式的 fp 值使 GET 或 PUT 语句从文件中指定的位置开始处理，其中，rba 是一个等于 RBA 值的 BASIC 表达式。对于 MU、MF 和 FF 型文件，系统进行检查以保证记录开始于这个指定位置。在 MU 文件的情况下，RBA 值必须指向一个 SOR 项。这种形式的 fp 值要求用户极其谨慎和胸有成竹，因为如果使用不正确，尤其在 FF 和 FI 型文件中，则可能造成最严重的破坏。而它恰好是随机地存取贮存在 MI、MU 和 FI 型文件中数据的唯一方法。
!%	这个 fp 值基本上与 !rba 相同，只是它把当前的 EOF 值作为 RBA 使用。它通常用于定位一个文件以进行扩展，即把记录或数据添加到文件的末尾。要扩展一个文件，必须用R 方式把文件打开，如果试图用 D方式打开去进行扩展，将产生错误。
!$rba	这个 fp 值允许用户定位文件，以便把下一个数据传送到该特定的文件区域，而不管用于传送数据所指定的存取技术或操作语句。使用这个fp值时将没有数据被传送到用户数据区域。在使用这个 fp 值的 GET 或 PUT 语句中，不能引用或包含 IGEL。使用这个 fp 值的文件定位，在下一个 INPUT/PRINT 操作或 GET/PUT 操作以前不产生作用，但这是有条件的，即只有在不指定另外的位置时才不起作用。它可以用来定位一个文件，在一个使用子程序的程序中进行随机存取，其中，这个子程序包含着一个进行所有文件存取的具有空缺 fp 值的 GET/PUT 操作，这样的一个程序可以随机地处理分布在整个文件上的顺序的记录组。
!$%	这个 fp 值的基本功能与 !$rba 是相同的，只是它使用当前的 EOF 值作为 RBA。使用这个 fp 值的 GET 或 PUT 语句不能引用和包含 IGEL。另外，由这个 fp 值得到的文件位置，在下一个 INPUT/PRINT 操作或下一个 GET/PUT 操作（如果不指定另一个 fp 值）以前不起作用。
!#rba	只用于 PUT 语句中，它使文件区的 EOF 值等于数值 rba。为了改变文件的真正的 EOF 值，即 FPDE 中的 EOF，必须关闭文件区或执行 PUT fan, & & 语句。提供的 EOF 值对于涉及的文件类型必须是合理的。对于 MF 和 FF 文件，它必须是文件标准记录长度的整倍数。
rn	这是记录号。它与 TRSDOS 支持的相同，rn 是一个 BASIC 数字表达式，其值为 1 到 32767 之间（包括两者）的一个整数值。指定的记录号被转换成 RBA，然后以 !rba 相同功能的方式使用它。

如上所述，某些 fp 形式改变了 REMRA, REMBA 或 EOF。为方便起见，把 fp 值和它们对这些字段的作用概括于下列判定表中。

表9.2 fp的作用判定表

fp值	REMBA	REMRA	EOF
空缺	1	1	6
*	1	2	6
#	3	4	6
$	4	4	6
%	4	4	4
&	4	4	4
& &	4	4	4
!RBA	1	1	6
!%	1	1	6
!$RBA	5	5	4
!$%	5	5	4
!#RBA	4	4	1
RN	1	1	6

在此表格中，各代码的意义如下：

1——字段被置成由该 fp 值得到的 RBA。

2——如果 REMRA 在语句执行的开始是无效的，或者文件是一个 MI 或 FI 文件，则字段被置成由此 fp 值得到的 RBA。换言之，如果当前文件位置是在记录的开头，则设置它，否则不变。

3——令字段等于 REMRA。

4——字段不变。

5——字段被置成一个无效的值。

6——对于输出文件或修改文件，如果 PUT 操作扩展此文件，则改变该字段。

在与文件位置有关的 FCB 中，总共有四个区域，它们是：

(1)当前文件位置

这一个字段可以看作为三个不同的值，它取决于GET或PUT在它的处理中所处的位置：

GPP1　它是 GET 或 PUT 操作执行之初的文件位置。除非文件已经关闭并重新打开，否则它的值相同于由最后对该文件区的 GET 或 PUT 操作留下的 GPP3 值。

GPP2　这是在文件定位完成以后并在任何数据传送以前得到的 RBA 值。GPP2 总是在设置 REMBA 和 REMRA 这些值时作为 REMBA 和 REMRA 存储的值。

GPP3　在数据的最后字节传送完（或者被跳过）以后的 RBA 值。

(2)REMRA

对于 MU, MF, FF 和字段项类型的文件，它含有处理中的记录开头的 RBA 值。对于MI、FI 和输出/输入文件，它等于 REMBA。见上述的 GPP2。

(3) REMBA

是 RBA 值，前面的进行数据传送的 GET 或 PUT 语句就从这儿开始其数据传送。如果文件是记录分段的，并且 REMBA 处于记录的开头，则令 REMRA 等于 REMBA。见上述 GPP2。

(4) EOF

文件中数据的最后字节的 RBA 值加 1。对于 MU、MF、FF 和字段项类型的文件，它有效地指向要写入文件的下一个顺序记录。对于 MI、FI 和输出/输入文件，它有效地指向要写入文件的下一个顺序字节。

管理 FCB 中各种 fp 值的一般方法如下：

如果必要，把文件从当前文件位置(GPP1)移到所需要的位置。这可以包括把修改过的缓冲区写回到磁盘上、计算新的扇区地址和把该扇区读入缓冲区。

把由所需要的位置得到的 RBA 放入当前文件位置(GPP2)。

令 REMBA 等于当前文件位置 (GPP2)。

如果文件是一个输出/输入文件，是用户分段或记录分段的，并且当前文件位置(GPP2)指向记录的开头，则令 REMRA 等于当前文件位置 (GPP2)。

数据传送（如果有任何要求的话）完成以后，当前文件位置 (GPP3) 含有传送的最后一个数据后面的字节的 RBA。

如果文件已被扩展，或fp=!#rba, 则 EOF 被置成相应的值。

9.2 OPEN 操作

任何文件都必须打开方能处理其中的数据。OPEN 操作本身在文件和应用程序之间建立一个 I/O 链。此 I/O 链的控制信息被保持在文件区中（它含有 FCB)。文件一旦打开，文件中的数据就可以通过 INPUT 或 GET 语句由程序读入使用，以及通过 PRINT 或 PUT 语句把数据放入文件中。当数据处理完成以后，必须关闭文件，这样就撤消了文件和程序之间的 I/O 链。

NEWDOS/80 支持五种 OPEN 方式：

″I″，用于顺序输入（INPUT 操作)。

″O″，用于顺序输出（PRINT 操作)。

″R″，用于随机存取的输入/输出(GET/PUT 操作)。

″E″，用于从已存在的文件或新文件的当前 EOF 位置开始顺序输出（E 即“Extend——扩展”)。

″D″，用于随机存取，但用户不想用 PUT 操作在超过当前 EOF 的位置使文件扩展或加长。

NEWDOS/80 BASIC 的标记项文件和固定项文件允许 GET 和 PUT 操作使用所有的五种方式。

NEWDOS/80 OPEN 操作的一般格式为：

(1) OPEN m,fan,filespec

(2) OPEN m,fan,filespec,lrecl

(3) OPEN m,fan,filespec,ft

(4) OPEN m,fan,filespec,ft,lrecl

其中：

m 是一个表达式，它的值是等于″I″，″O″，″R″，″E″或″D″的一个字符串。它指定了用于文件的存取方式，以及文件的初始位置。

fan 是要打开的文件区的号码。

filespec 是一个表达式，它的值是等于″FI″，″FF″，″MI″，″MU″或″MF″的一个字符串。它用于识别一个特定的 NEWDOS/80 文件子类型。对于这些子文件类型，我们将全部给予简要的解释。每当在 OPEN 语句中使用 ft 的时候，GET 和 PUT 操作是把数据取出和送入文件的唯一方法，而不能使用 INPUT 和 PRINT 操作，也不能使用 BASIC 的 FIELD 语句。所有的 GET 或 PUT 语句都必须指定一个IGELSN或者本身包含 IGEL。应用程序必须不以任何方法改变或直接引用文件区中的数据。有两个 ft 值（″MF″和″FF″，即 MF 文件和 FF 文件）要求在 OPEN 语句中说明 lrecl；还有一个 ft 值（即″MU″）允许任选的 lrecl 说明。

lrecl 是一个表达式，它的值是一个表示逻辑记录长度的整数值。对于字段项文件，该整数值在 1 到 256 之间；对于标记项和固定项文件，是在 1 到4095之间。所有的记录分段文件都必须指定 lrecl（字段项文件除外，其中若不指定 lrecl，则假设为 256)，对于字段项、″FF″和″MF″文件，它指定了文件中所有记录的精确的长度，或者指定了文件类型″MU″的任选的最大逻辑记录长度。

注意，BASIC OPEN 语句的标准格式没有改变（格式 1 和 2)，因此，允许已存在的字

段文件和定向应用的输出/输入文件继续运行。标准格式的扩充识别文件为 NEWDOS/80 文件，并规定文件类型和用于取回和控制其中数据的存取技术。

NEWDOS/80 支持的所有文件类型中，最容易使用和理解的一个是"MU"。它定义一个含有标记项的文件，并将它分段成各种长度的记录。记录长度被定义为记录的 SOR 的 RBA 与下一个记录的 SOR 的 RBA，或者与文件的 EOF 的 RBA（两者中无论哪一个跟在其后）之间的差。在 OPEN 语句中不必指定记录长度，但是，如果提供了记录长度，则它指定了文件中允许的最大记录长度。

"MU" 文件中的一个记录可以用另一个记录修改，其长度可以相同或比它起始建立时的长度短，但是不能将它加长。当一个记录被一个长度短于原记录长度的记录修改时，就在新记录的右边至原记录的末尾之间用填充项（十六进制00字节）装填。这个较短的记录随后可以用一个较长的记录替代，只要新的记录不长于最初写入文件的记录。

"MU"文件类型代替了 BASIC 的通过 INPUT/PRINT 操作进行存取的顺序输入/输出文件。它的最大长处是程序员不必提供特殊的分隔符把两个相邻字符串项隔开（在BASIC 顺序文件中，必须在两个字符串之间输出一个分隔符逗号，以便输入时能够分开它们）。"MU" 的第二个优点（也是所有 NEWDOS/80 BASIC 文件的优点）是数字值是以它们的内部格式存储到磁盘上的。也就是说，例如，一个双精度值是作为一个 8 字节项写入磁盘的，而不是在输入时需要转换回内部格式（8 字节）的一个多达14个字符的项。不要忘记，在标记项文件(如 "MU")的情况下，一个双精度项实际需要 9 个字节，这是由于要前缀控制字符的原因。如果在 OPEN 语句中指定了 lrecl，则它为文件设置了最大的可容许的记录长度，而且其中必须考虑到每个记录中的所有控制字节（包括 SOR 项）。

使用上比较简单的 ft 格式是"MF"和"FF"。两者都按照记录分段并且具有固定长度的记录来识别文件。它们都意味着所有的记录有相同的内部数据结构，但并不保证该条件。OPEN 语句必须指定文件中所有记录的精确的逻辑记录长度。在 "MF" 的情况下，标记字节也必须计算在长度之中（注意，"MF"文件在每个逻辑记录的开头不使用 SOR 项，因为 BASIC 知道每个记录从何处开始)。每个 GET/PUT 操作要对照 OPEN 时指定的 lrecl 检查 IGEL 的数据长度，如果 IGEL 的长度较大，则产生一个错误状态。

使用上最困难的 ft 格式是"MI"和"FI"。它们规定文件是完全在用户控制下记录分段的。对这些文件类型，在 OPEN 语句中不必指定 lrecl。这些 ft 格式允许文件含有非常复杂的数据关系，及 BASIC 所不了解的用户数据结构。BASIC 不能从一个用户记录前进到另一个，这必须由用户自己来处理。

9.3 CLOSE 操作

CLOSE 操作打断了 BASIC 应用程序与文件之间由 OPEN 操作建立的 I/O 链。它的一般格式没有被 NEWDOS/80 修改，仍与 TRSDOS 相同：

CLOSE [fan,…]

根据文件的操作方式和类型，文件区缓冲区的内容可以用这个操作写入磁盘。对于文件的输出和随机存取，文件的目录项被修改，以反映存储在文件区的 FCB 中的当前 EOF 值。

9.4 GET 语句

在字段项文件（TRSDOS 随机文件）处理中，GET 语句用于把一个特定的记录读入文件区的缓冲区。在此以前，要使用 FIELD 语句把缓冲区划分好字段，以便在记录读入缓冲区后调整字符串变量的数据指针来寻址缓冲区本身。这种数据存取方法使这种文件在NEWDOS/80中被称为字段项文件，因为所有的其他文件类型也可以随机地使用。

除了继续支持字段项文件外，NEWDOS/80 的 GET 语句被用于标记项和固定项文件处理中，把数据从一个文件传送到用户指定的变量中、定义变量本身、或者为后面的操作定位文件。数据从磁盘到缓冲区的实际传送，只有在必要时根据 BASIC 对 IGEL 数据的测定，在当前缓冲区中需要有关的数据时才发生。

GET 语句的一般格式为：

(1) GET fan （假设为空缺的 fp）

(2) GET fan, fp

(3) GET fan, fp, igelsn

(4) GET fan, fp,, igel

格式 1 和 2 用于字段项文件，并与 TRSDOS BASIC 兼容。它们当然也可以用于 NEWDOS/80 BASIC 的应用程序中。

格式 3 和 4 是 NEWDOS/80 BASIC 独有的。它们必须在文件区打开的情况下，用于对标记项文件和固定项文件作数据传送的 GET 操作中。格式 2 的功能已经被 NEWDOS/80 独有的额外几个新的 fp 参数扩充。

格式 3 指定了 IGEL 的位置，即含有 IGEL 的语句的起始行号。格式 4 含有作为 GET语句本身的一个整体部分的 IGEL，此 IGEL 含有数据名称（变量），在 GET 操作完成后，这些名称将含有数据。

在 NEWDOS/80 中，IGEL 或 fp 参数不具备访问一个文件区的功能，即使该文件区就是 GET 或 PUT 语句使用的哪一个。因此，IGEL 或 fp 中不能含有访问文件区的函数，如 EOF、LOC 和 LOF 等。

在成功地完成 GET 语句时，文件区靠左定位于：

（1）对固定项文件，定位于文件的下一个字节。

（2）对于标记项文件，定位于文件中的下一个文件项。

（3）对于字段项文件，定位于下一个 256 字节的记录。

如果在 GET 语句处理的过程中遇到一个错误，文件区域就被恢复成语句执行以前的状态和内容。在改正错误以后，可以再执行 GET 语句。当检测到一个错误时，在IGEL 中命名的变量的内容是完全不可预测的，并且是不能使用的，除非已经成功地执行了 GET 操作。

当 GET 语句引用或含有 IGEL 时，相邻的文件项就被传送到IGEL 中命名的相邻变量之中。

1. 对于固定项文件

IGEL 的字符串变量用表达式前缀中指定数目的字节填充。结果使变量的长度等于前缀的值。

IGEL 的数字变量用该项的内部格式所对应数目的字节填充（整数项为 2 个字节，单精度项为 4 个字节，双精度项为 8 个字节）。

在把数据传送到用户变量的第一个 GET 操作之前，可以执行一个使用参数 fp=%的GET

语句。当执行这个 GET 语句时，通过 fan 访问的文件不必打开，由于这个 GET 语句的目的是对固定项文件执行一个伪 FIELD 操作。伪 FIELD 操作在处理 IGEL 项时，数字变量保持不变，忽略(len)$和(len)#这样的项，字符串变量的长度规定为表达式前缀的值，并且根据需要进行截尾或在其右边添加空格。如果执行伪 FIELD 操作时字符串变量已经存在，并且它的值（内容）不驻留在BASIC字符串区域中，则把它的内容移到那里。这是在证实有足够的字符串空间可供后续的操作使用时进行的，由于后续的数据传送的GET将实际地把数据移入变量，而不是仅仅改变变量的数据指针。字符串变量一旦被一个伪FIELD操作引用，它的内容就只能被 LSET 或 RSET 改变，以保证变量长度不变。在 NEWDOS/80 版本 1 中，对固定项文件进行任何 PUT 操作之前都需要进行伪 FIELD 操作，而在 NEWDOS/80 版本 2 中不需要这样做，并且使用固定项文件的许多程序根本不使用伪 FIELD 操作。

如果文件是记录分段的，并且在文件项的数据传送开始时，记录中从当前文件位置起所存在的字节少于 IGEL 项所需要的，则发生〃RECORD OVERFLOW"（记录溢出）错误。

2. 对于标记项文件

一个空缺的 IGEL 表达式使得相应的文件项被跳过。

字符串变量的表达式前缀被用于限制实际传送给变量的字符数目。如果文件项比表达式前缀所容许的短，则字符串变量的长度被置成该文件项的长度。如果文件项比表达式所容许的长，则文件项在右端被截尾成该长度，就象 LSET 语句所做的那样。

在遇到 SOR 和填充项时就跳过它们。

如果文件项的类型和 IGEL 项类型是不兼容的，则发生"TYPE MISMATCH"（类型失配）错误。例如，如果文件项类型是单精度数，IGEL 项类型是字符串，就产生此错误。但是，如果 IGEL 项是整数，则不发生错误，除非文件项的值超过了整数项的合法值。

如果文件是记录分段的，并且在文件项的数据传送开始时，记录中从当前文件位置起所剩余的项少于 IGEL 中存在的，则产生〃RECORD OVERFLOW〃（记录溢出）错误。

两个特殊格式的 fp 参数可以用于为后续处理设置文件位置，而不管对文件进行正常处理的类型。它们是 fp=!$rba 和 fp=!$%。使用这些格式中的任何一个都使 REMRA 和 REMBA 被标记为无效。在格式 3 或 4 的 GET 语句中使用这两个 fp 值的任何一个都是无效的，由于不发生实际的数据传送。

多个 GET 语句可用于从单个记录中取回相邻的文件项目。这一技术叫做部分记录的 I/O。例如，一个记录中的第一项可以根据含有姓名和地址、受理号和金额、或发票号和要求的船期等来区别该记录。第一个字节可以单独地读出并用于在程序中把控制传送给相应的处理其后数据的程序。

部分记录 I/O 作为一种存取技术可容易地用于固定项文件和字段项文件。在字段项文件中，当新的记录与以前的记录类型不同时，这技术叫做重新规定字段。在标记项文件中，记录中要跳过的文件项只要作为空项放在 GET 语句的 IGEL 中。在固定项文件中，必须确定要跳过的字段的长度，并指定此总和为 IGEL 项(len)$的长度前缀 len，以便定位该记录于要传送的相应字节。部分记录 I/O 的实际优点，在固定项文件中，是可以修改如此小的一个嵌在记录中的单个字段，而与记录中的所有其他数据无关；在标记项文件的情况下是，必须首先读出被修改项以外的所有项目，然后再与修改项一起重新写入，以维护它们的内容。部分记录 I/O 的主要好处是，在一个文件中可以持有几种记录格式，并且只要象传送许多数据所

需要的那样去区别特定的格式。

9.5 PUT 语句

在字段项文件处理中，程序员要执行一个 FIELD 语句(如果没有预先做过)，以定义覆盖主存储器单元的变量缓冲区。接着，使用 LSET 或 RSET 语句把这些变量的值移入缓冲区中。最后，使用 PUT 语句写（或缓存）该记录。

对于标记项文件或固定项文件的处理，BASIC 变量的内容也使用 PUT 语句写(或缓存)，但不需要象上述字段项文件那样首先把数据移入特殊编程的变量。替而代之使用 IGEL 来指定 PUT 过程中要把哪个变量的内容送入文件。

记住，在 IGEL 表达式或 fp 表达式中不可以含有引用文件区的函数，如 EOF,LOC,LOF 等。

PUT 语句的一般格式是：

(1) PUT fan (假设为空 fp)

(2) PUT fan, fp

(3) PUT fan, fp, igelsn

(4) PUT fan, fp,,igel

格式 1 和 2 用于字段项文件操作，并与 TRSDOS BASIC 兼容。它们当然可以继续用于 NEWDOS/80 控制下运行的应用程序中。

格式 3 和 4 是 NEWDOS/80 BASIC 特有的。在标记项文件或固定项文件处理过程中，当要把数据传送到文件去的任何时候，必须使用其中之一或两者。格式 3 指定了 IGEL 的位置，此 IGEL 含有要送往文件的表达式。格式 4 含有作为 PUT 语句之一部分的 IGEL本身。在格式 2 的 PUT 操作之前可以先执行格式 3 或格式 4 的 PUT 语句，以到达后续数据传送所必须的文件位置。

在成功地完成 PUT 语句后，文件区域靠左定位于：

(1)文件的下一个字节（对于固定项文件）。

(2)文件中的下一项（对于标记项文件)。

(3)下一个 256 字节的记录（对于字段项文件)。

如果在 PUT 语句的处理过程中遇到一个错误，文件区域就被复原成 PUT 语句执行以前所处的状态和位置。错误所造成的结果是，文件中的数据完全不可预测，并且极大可能在后续的 GET 操作中引起错误。这种情况只发生在现有记录的修改过程中。如果可能和可行的话，应在改正错误之后再执行 PUT 操作。当文件用 "R" 或 "D" 方式打开时，为尽量减少破坏的可能性，NEWDOS/80 BASIC 一旦在 IGEL 说明中找到错误，就把全部 IGEL 处理两遍，再实际地把数据传送到缓冲区。

当 PUT 语句引用或含有 IGEL 时，相邻的 IGEL 表达式的值就被传送到文件缓冲区，并变成文件项目。

1. 对于固定项文件

字符串变量或表达式的长度可以和 IGEL 中表达式前缀所允许的长度不同。较短的字符串所对应的文件项，将在文件项右边填充空格；较长的字符串所对应的文件项，将按 LSET 使用的方式在右边截尾。换句话说，表达式前缀值精确地确定了把多少字节移入文件项。

如果超过逻辑记录长度，就发生记录溢出错误状态。在全记录 I/O 过程中，如果 IGEL 中所有项目的长度总和超过逻辑记录长度，就发生此错误。在部分记录 I/O 中，如果 IGEL 中所有项目的长度总和超过记录中剩余的字节数，则发生此错误。

在把用户数据实际传送到缓冲区的第一个 PUT 语句之前，可以执行一个使用 fp=% 的 PUT 语句。用此 fan 引用的文件在这个 PUT 操作时不必打开，由于它的目的是执行伪 FIELD 操作。伪 FIELD 操作在处理 IGEL 项时，数字项保持不变，(len)$ 和(len)#都被忽略，字符串表达式的长度规定为等于表达式的值，并根据需要在右边截尾或添加空格。如果在执行伪 FIELD 功能的 PUT 语句时，字符串变量已经存在，并且字符串本身不在 BASIC 字符串空间中，则把它移到那里。字符串变量一旦被一个执行伪 FIELD 功能的 PUT 语句引用，它的内容就只能被 LSET 或 RSET 语句改变，以保证它们的长度不变。在 NEWDOS/80 版本 1 中，对固定项文件进行任何 PUT 操作以前，都需要进行这个伪 FIELD 操作。

2. 对于标记项文件

SOR 和填充项根据文件的 ft（文件类型）的需要被插入文件缓冲区，PUT 操作的 fp(文件位置）和 IGEL 数据长度依文件的记录长度为转移。

作为 PUT 语句所引用的 IGEL 中的表达式，在 LET 语句等号右边任何句法上合法的内容几乎都是合法的，只是在这样的表达式中不可以引用文件区域。应从任何 IGEL 表达式中明确地除掉 LOC, LOF, EOF 和引用 fan 的任何其他表达式。

当 IGEL 中的字符串表达式有长度前缀时，此前缀确定了要写入文件的最大字符数目。如果字符串比表达式前缀所允许的短，则字符串按原样写入文件。如果字符串较长，则对应文件项在右边被截尾，就象 LSET 操作所做的那样。

每个字符串文件项都需要一个或两个标记字节，这取决于字符串中的字节数目。如果字符串中有 0 到 127 个字节，则它只需要一个标记字节，用以在文件中描述它。如果它有 128 个以上的字节，那末就需要两个标记字节来描述它。当在 OPEN 语句中说明 lrecl 时，必须考虑到所有这些标记字节。

数字的 IGEL 表达式以它们的内部的 BASIC 格式放在缓冲区中：整数为 2 个字节；单精度数为 4 个字节；双精度数为 8 个字节。不要忘记，每个独立的文件项都有与它相关的标记字节，并且刚才提到的那些文件项类型的正确长度应是 3,5 和 9 个字节。

IGEL 中的数字和表达式，在送入文件之前，首先被转换成最紧凑的内部的 BASIC 数据类型，并保持其精度。例如，数字3.14159将按单精度数（包括标记字节在内共5个字节）送入文件。又如，由表达式 LEN(A$)−LEN(B$)得到的值将按整数（包括标记字节共 3 个字节）送入文件。

可用两个以上的 PUT 语句输出一个记录的所有项目。构成一个逻辑记录的实际字节数不能超过 OPEN 语句中指定的 lrecl 值，或系统允许的最大值 4095 字节。超过这些极限之一都会引起“RECORD OVERFLOW”（记录溢出）错误。

在打开 MU 和 MF 类型文件用于修改目的的随机存取情况下，文件中现有的记录，从位于 PUT 操作所传送数据的开头的当前文件位置起到记录结束（由下个 SOR 或 EOF确定）为止，就被全部替换。如果累计的 IGEL 数据长度小于文件记录的剩余长度，则把 IGEL 数据送入文件，并用填充项填充，以完全填满文件记录。在以这种方式操作时一定要十分小心，因为如果在修改时，PUT 语句的 IGEL 所定义的项目少于文件记录中现有的项目，则多余的

文件项就被除去了，如果以后的 GET 语句想要该记录提供原先数目的文件项时就会遇到问题。

MI 类型文件中的文件项不能修改，因为系统没有用户记录结束于何处的概念，所以不能象 MU 和 MF 文件那样在记录的末尾添加新内容。

3. 对于固定项文件和标记项文件

文件区域的缓冲区被实际写入磁盘是在：

(1)数据由 IGEL 填入缓冲区的最后一个字节，并且还有更多数据要传送时。

(2)执行带有"&"或"&&"的 fp 参数的 PUT 语句时，使缓冲区以它当前状态写入磁盘。

(3)明显地用 fan 或隐含地用一般的（不明确指定 fan）CLOSE语句关闭文件时。

如果文件中的数据是特别关键的，则程序员应考虑使用带有"&"的 fp 的 PUT 语句，这将使得文件区域的缓冲区被写入磁盘而不影响当前文件位置。如果缓冲区中没有数据要写入磁盘，则忽略这个特殊的 PUT 语句。假如程序使用的某些其他文件区域要求这个区域中的数据驻留磁盘时，则必须使用"&"的 fp 参数。不要忽略"&"的 fp 只用于格式 2 的 PUT 语句中这一事实，要写入文件的任何数据都必须首先用格式 3 或格式 4 的 PUT 操作将它们放到缓冲区。fp=&的使用不限于标记项文件，它也可以用于字段项文件或输出/输入磁盘文件。

上面讲述的 PUT fan, & 语句的规则也适用于 PUT fan, && 语句，此外，此语句根据文件区域的控制信息把文件的 EOF 写回到文件的目录项中。

有两个特殊的fp格式可用于设置文件位置，以便对文件正常地进行后续的处理，而不管涉及 GET, PUT, INPUT 或 PRINT 等操作的实际处理类型。这两个特殊格式是fp=!$ rba和fp=!$ %。使用其中的任何一个都使 REMRA 和 REMBA 被标记为无效。如果不进一步指定fp, 就定位文件，于是在 rba 或 EOF 开始处理下一个 GET/PUT/INPUT/PRINT 操作。使用这两个fp值不发生数据移动，由于只有在格式 2 的 PUT 操作中允许使用它们。

使用 !#rba 的 fp 值的 PUT 语句，使文件的 EOF 被置成 RBA 值 rba。不要忘记，在执行 CLOSE 或 PUT fan, &&语句之前，EOF 值不写入文件的 FPDE。在 EOF 最后形成以前，它可以用这种方式改变多次。如果 OPEN 语句的方式是"D"，并且 RBA 超过当前 EOF 时，则产生错误状态。这个 fp 值只能用于格式 2 的 PUT 语句中。

如同用 GET 语句进行顺序输入那样，PUT 语句可以用于顺序输出方式中。在用方式"O"打开文件以后，用 PUT 语句可以顺序地建立标记项文件或固定项文件，随后，在用方式"I"打开文件以后，可用 GET 语句顺序地读此文件。当打开方式是"R"或"D"时，使用 GET 和 PUT 语句可以随机地修改同一个文件。使用部分记录 I/O 存取技术，可以修改FF 和 FI 型文件中的单个数据字段。

假如一特定的数据文件是特别关键的，并要求进行只读的随机存取，则不要使用"R"方式打开文件，可以使用方式"I"来代替。使用这种方式打开的文件，将使任何 PUT 操作的企图得到"BAD FILE MODE"（文件方式不对）错误。

9.6 LOF 函数

LOF 函数的功能是把文件的最后一个记录的记录号送回给程序员。它与 TRSDOS BASIC 是完全一样的。它的一般格式是：

LOF (fan)

其中，fan 指定了文件区的号码，这个文件区就是要求得到最后记录号的文件区。如果文件是空的，则 LOF 得到零。LOF 当然只能用于字段项文件、MF 和 FF 型文件。

9.7 LOC 函数

在 TRSDOS BASIC 中，LOC 函数给程序员送回对指定文件区通过 GET 或 PUT 语句最后存取的记录号。在 NEWDOS/80 BASIC 中，它的功能已扩充成允许程序员求得一组项目、记录或文件 EOF 的文件位置，或者确定当前文件位置是否正好位于文件的 EOF 或超过文件的 EOF。它的一般格式如下：

(1) LOC (fan)

(2) LOC (fan) $

(3) LOC (fan) %

(4) LOC (fan)!

(5) LOC (fan)#

其中，fan 指定了含有所需信息的文件区号码。

1. 格式 1 （无后缀字符） 是 TRSDOS BASIC 中使用的一种，功能基本上与TRSDOS 中相同。对于字段项文件和 MF 文件、FF 文件，它得到通过 GET 或 PUT 语句最近读或写的记录号。如果文件没有被存取过，则得到零值，除非用方式"E" 打开文件，在这种情况下，得到文件中最后一个记录的记录号。如果被访问的文件不是由固定长度的记录构成的，则产生"BAD FILE MODE" （文件方式不对）错误状态。

2. 格式 2 （后缀"$"） 用于提供文件区域的位置与文件的 EOF 之间相互关系的真或假表示。它如下所述分别得到－1 （BASIC IF 语句的"真"）或 0 （BASIC IF 语句的"假"）：

（1）对于记录分段（固定项、MU、MF 和 FF 型）文件：

如果 REMRA 是有效的，并且下个记录开始（未必是当前文件位置！）的 RBA等于或大于 EOF 值，则得到"真"值；否则得"假"值。

如果 REMRA 是无效的，并且当前文件位置的 RBA 等于或大于 EOF 值，则得到"真"值；否则得到"假"值。

（2）对于用户分段（MI 和 FI 型）文件和输出/输入文件：

如果当前文件位置的 RBA 等于或大于 EOF 值，则得到"真"值，否则得"假"值。

3. 格式 3 （后缀"%"） 以 RBA 格式给程序员送回当前文件 EOF 的文件位置。这个值可用于开发对文件的索引，其中，索引项在数据记录添加到文件去以前就建立在 EOF 位置。使用这种形式的 LOC 允许在主数据文件顺序建立过程中建立索引。

4. 格式 4 （后缀"!"） 若 REMRA 是有效的，此函数得到字段项文件、MU、MF 和 FF 型文件的下个逻辑记录的 RBA 值。在所有其他情况下（包括输出/输入文件），它得到当前文件位置的 RBA 值。对于记录分段文件，LOC 函数得到的这个值可以用来在数据记录写入以前建立顺序记录的索引项。对于用户分段文件和输出/输入文件，得到的这个值可以用来在数据项组写入文件之前为它们建立一个索引项。为了使索引正确无误，必须使用带有空缺 fp 的 PUT 语句或用 PRINT 语句去写数据；几乎所有其他的 fp 格式都使得到的 RBA 值与数

据的实际位置不同。如同使用格式3那样，这个格式可以用来在写入顺序文件时建立索引。

5. 格式5（后缀"#"） 以RBA格式得到当前的REMRA。如果REMRA是无效的，则产生"BAD FILE MODE"（文件方式不对）错误状态。例如，因为使用了fp=!$%，使REMRA无效。这也可以用于所有的文件类型给记录或数据组建立索引项，但是是在记录或数据写入以后。

通过使用由LOC(fan)%、LOC(fan)! 或者LOC(fan)# 函数得到的值，程序员就能够建立记录（记录分段文件）或项目组（用户分段文件和输出/输入文件）的索引。得到的这个值可以包含在记录或文件项之中，并在以后通过fp 类型 !rba 或 !$rba, 用来定位文件。

9.8 MU文件

MU文件类型是所有NEWDOS/80的文件类型中最容易实现的。最初的考虑是，希望由它取代TRSDOS的顺序文件。在TRSDOS中，顺序文件不能修改；在NEWDOS/80中，除输出/输入文件（TRSDOS顺序文件）和MI型文件以外，所有的文件都可以修改。

MU型文件被分成各种长度的记录，并且每个记录都能被系统检出。这个特征使程序员从必须深知每个记录的大小中解放出来。程序员可以通过在OPEN语句中指定lrecl值（小于系统最大值4095字节）来利用记录长度的最大值。任何超过最大记录长度的记录都将引起'RECORD OVERFLOW"（记录溢出）错误状态。

除记录分段外，MU文件中的文件项都带有标记。标记字节要占用文件中的空间，并必须包括在任何的记录长度计算中，同时要包括标记每个记录开始的SOR字节。这些标记字节识别该字节后边的数据类型，在字符串情况下，它告诉系统该字符串的长度。字符串长度可以为0到255字节，就象在BASIC中那样。128到255字节的字符串需要2个标记字节，而小于128字节的字符串以及其他所有项目只需要1个标记字节。数字项以它们的内部格式存储到磁盘上：整数为2个字节，单精度项为4个字节，双精度项为8个字节。不要忘记由于是标记文件项，这些长度必须加1，相应成为3，5和9个字节。

即使数字项在所有NEWDOS/80文件类型中以它们的内部格式存储在磁盘上，也不使用（的确不必）BASIC的CV×和MK×来执行伪字符串的转换，以便进行这种格式的数据存储；对于字段项文件，仍必须使用CV×和MK×来完成这种格式的数据存储，就象TRSDOS BASIC中那样。

MU文件可以通过指定"O"作为OPEN语句中的方式来建立，文件将用连续的PUT语句中的数据建立，而不管打开时文件的存在与否。MU文件也可以用OPEN语句中的方式"R"建立，但只有在打开文件以前该文件不存在时。建立MU文件的第三种方法是，对于以前不存在的文件或者存在的空文件，在OPEN语句中使用方式"E"。

一个已经存在的MU文件可以通过在OPEN语句中指定方式"E"顺序地进行扩展。如上所述，如果文件是空的，它将被有效地建立起来而不是扩展或延长。顺序扩展MU文件的另一个方法是在OPEN语句中指定方式"R"，在这一方式中，如果指定非空的fp，系统就把填充字节从当前EOF开始写到指定的新记录的开头。对小于EOF的文件位置进行的任何PUT操作都产生修改动作，而不产生文件的延长。

用指定"I"作为OPEN语句中的方式，可以顺序存取MU文件，使用这一方式可防止发生意外的修改。当用方式"I"打开文件时，也可以随机存取该文件。如果文件在打开时不存

在，则产生错误状态。MU 文件也可以在 OPEN语句中指定"R"或"D"方式进行随机存取。使用这些方式时，如果文件在打开以前是不存在的，并且不先进行 PUT 操作和相继的重新定位就执行任何 GET 操作，则发生错误。

MU 文件可以通过在 OPEN 语句中指定方式"R"或方式"D"进行修改。使用方式"D"不能进行文件的扩展。在这些方式的任何一个中，根据使用的 fp 值和 IGEL 的内容，可以修改整个记录乃至单个项目的任何内容。

为理解系统在 MU 型文件中的作用，我们进行以下工作。首先，我们用一个非常简短的 BASIC 程序建立一个 MU 型文件。然后，我们在所谓计算器方式（即 BASIC 的命令方式）中通过一些工作去存取文件和修改文件。为建立此文件，输入并运行以下 BASIC 程序：

```
10  CLEAR  250
20  OPEN "O", 1, "MU/DAT", "MU"
30  PUT 1,,, "ABCDEF", "2ND STRING";
40  PUT 1,,, STRINGS (120, "*")+"0123456789";
50  I%=2:I!=4:I#=8
60  PUT 1,,,I$,I%,I!,I#;
70  CLOSE
```

把此程序用一个适当的名称（下面称 MUFILE）存起来，因为以后需要用它。

现在，请注意该程序在语句30和40中使用了最简单的 IGEL 格式，要写入文件的值是在 GEL 本身中。在行60的 PUT 语句中引用了四个不同的 BASIC 数据类型：字符串、整数、单精度和双精度数。也请注意，在 OPEN 语句中没有指定 lrecl，这允许记录长达4095字节。

运行此程序，建立名叫"MU/DAT"的文件。为了学习的目的，运行 SUPERZAP 程序，用 DFS 功能读出由 MUFILE 程序写的扇区。

此扇区的第一个字节是十六进制70。这是第一个记录的 SOR 字节。MU 文件中的所有记录都用这个字节开始。然而，要知道不是所有的十六进制70都是记录的开始字节，这个特殊字节可以出现在数字值以及字符串中，在字符串里它是小写字母"p"。

第二个字节是表示后 6 个字节为字符串的文件项标记字节(86H)。该字符串第一个字节的位移量加 6 给出了第二个字符串的文件项标记字节(8AH)的位移量 (08H)。这个标记字节定义一个10字节长的字符串。如果你现在从该标记字节往下数到第十一个字节，就会发现第二个记录的 SOR 字节(在位移量 13H 处)。下面的标记字节(71H)表示长度大于 127 字节的字符串，该标记字节后面的字节是含有字符串长度的第二标记字节，而不是字符串数据本身的一部分。在此用一个很小的十六进制算术运算 (13H+2+82H+1=98H) 就可得出第三个记录的 SOR 字节的位移量 98H。该 SOR 字节后边的标记字节 (80H) 表示字符串长度为 0 字节——一个空字符串。下一个标记字节 (72H) 表示后2个字节为一个整数。在整数后边是表示后 4 个字节为一个单精度数的标记字节 (73H)。在该数后边是表示后 8 个字节为一个双精度数的标记字节(74H)。到此为止（位移量ABH），我们已用完了实际写入磁盘的数据，其后的任何数据都是不可预测的。现在，我们把这个"MU/DAT"文件的详细内容列出如下，其中字节位移量和字节的内容都以十六进制表示（无后缀 H），SOR 表示 SOR 字节，MK 表示文件项标记字节，pad 表示填充字节，EOF 表示文件结束。

"MU/DAT"文件内容：

```
位 移 量  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12
字节内容  70 86 41 42 43 44 45 46 8A 32 4E 44  20 53  54  52 49 4E 47
(第一记录)
注    释  SOR  MK  "ABCDEF"   MK  "2ND  STRING"
位 移 量  13 14 15 16 17……       8D 8E 8F 90 91 92 93 94 95 96 97
字节内容  70 71 82 2A 2A……       2A 30  31 32 33 34 35 36 37 38 39
(第二记录)
注    释  SOR  MK  MK  "*……*"    "0123456789"
位 移 量  98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB
字节内容  70 80 72  02 00  73  00 00  00  83  74  00  00  00  00 00  00 00  84
(第三记录)
注    释  SOR MK MK I%=2 MK I!=4 MK    I#=8              EOF
```

由此，我们看到了数据是如何存储在MU文件中的，这也同样适用于其他标记项文件。下面，我们将用 GET 语句在“计算器方式”存取此数据并分析结果。随后，我们将介绍一些错误。在进一步深入研究之前，先返回到 BASIC READY 状态，输入 CLEAR 50和 NEW 命令，并打入以下三行程序，并把它也存起来（这将节省以后的步骤），因为它随后还将用于所有的其他文件类型的试验中。

```
10  PRINT LOC(1)$; "$ EOF TEST"; LOC (1)%; "% EOF RBA"
20  PRINT LOC (1)!;"! NEXT RCD RBA      ";
30  IF LOC(1)!=0 THEN PRINT ELSE PRINT LOC (1)#; "# REMRA"
```

这程序的目的是显示出我们可用的文件位置值。为清楚起见，字符串的第一个字符表示用于得到被显示值的 LOC 函数的后缀，字符串的剩余部分是和特定的 LOC 功能有关的记忆符号。

现在要做的第一件事情是打开文件，以进行输入。打入：

```
OPEN "I", 1, "MU/DAT", "MU"
```

现在打入 GOTO 10 命令，运行不久前输入的显示文件位置的程序（必须用 GOTO 而不用 RUN 命令来运行程序，因为 RUN 关闭任何打开着的文件）。系统将响应：

```
0  $ EOF TEST   171 % EOF RBA
0  ! NEXT RCD RBA
```

注意，REMRA 值没有被打印出来，这是因为还没有设置该值，并被系统标记为无效。由于我们编制的这个程序是不太精巧的，它只对下个记录的 RBA 值是否为零进行检查，而不企图对 REMRA 的实际有效性进行检查。

现在，我们读整个的第一个记录，打入：

```
GET 1,,, A$, B$; :PRINT A$, B$:GOTO 10
```

系统响应为：

```
ABCDEF    2ND STRING
0 $ EOF TEST   171% EOF RBA
19 ! NEXT RCD RBA   0 # REMRA
```

注意，两个有关 EOF 的值(0 和 171) 没有改变，但是下个记录的 RBA 改变了。它现在

含有第二个记录的 SOR 字节的十进制位移量(19)。这是记录分段文件中 GET 操作的正常作用。也请注意，REMRA 现在已经出现了，并且它的值为 0。记住，对于记录分段文件，REMRA 含有的值是对该文件区的 GET 或 PUT 操作中涉及的最后一个记录的 RBA，除非由于使用了 OPEN 语句或 !$ RBA 参数使它标记为无效。

现在我们回过头来通过使用 fp 值 (#) 使文件位置回到 REMRA 值，再读第一个记录的全部。为了证明已第二次读了该记录，我们把变量名的次序反过来。打入：

```
GET 1,#,, B$, A$ ; : PRINT A$, B$ : GOTO 10
```

系统将响应：

```
2ND STRING ABCDEF
0 $ EOF TEST 171 % EOF RBA
19 !NEXT RCD RBA 0 # REMRA
```

EOF值仍然不变，但这时另外两个值也没有改变。这是因为文件的下个记录指针在数据传送前被改变成 REMRA 值。于是下个记录指针被移到 REMRA，接着把数据传送给命名的变量。当指定 fp 为 !rba 时，遵循此相同的一般方法。

现在，让我们不管下个记录的内容（其中有 120 个星号 * 的记录），在定位的同时处理第三个记录。打入：

```
GET 1,,,; : GOTO 10
```

系统响应：

```
0 $ EOF TEST 171 % EOF RBA
152! NEXT RCD RBA 19 # REMRA
```

这实际上没有什么意外的东西，在 GET 语句没有指定文件位置的情况下，下个记录的 RBA 被移到 REMRA。由于 IGEL 中缺乏变量名，所以不发生数据传送，文件靠左定位于记录的第一项。

现在，让我们做一些部分记录 I/O 试一试。我们就从传送第三个记录的字符串开始，并保持传送以后的文件位置。于是下个数据传送将由整数开始。打入：

```
GET 1,,, A$ ; : PRINT A$ : GOTO 10
```

系统响应：

```
(空行)
-1 $ EOF TEST 171 % EOF RBA
171! NEXT RCD RBA 152 # REMRA
```

在此应注意几件事。由于前一个 GET 操作已使文件靠左定位于第二个记录的第一项，所以通过本例中指定 fp 为空缺的这个 GET 操作的文件定位阶段，文件自动地前进到第三个记录的第一项的开头。于是，读出第三个记录的第一项，并且文件靠左定位于第三个记录的第二项。这里我们已开始处理文件中的最后一个记录。对此，系统是不告诉我们的，但是，通过 LOC (fan)$ 语句已得到了我们可以利用的信息。在普通的顺序数据处理情况下，文件的 EOF 状态是作为读 (GET) 逻辑的一个功能进行测试的，并且在 EOF 状态为真时，把控制传送到程序员指定的数据结束子程序。由于在 NEWDOS/80 中没有为这种子程序的规范作好准备，因此必须在准备把下个记录内容送入内存的 GET 语句之前，直接地测试文件的 EOF 状态，并且，如果发现 EOF 状态是真的，就要进行相应的处理。

注意，LOC (fan)! 值与 EOF 的 RBA 值相同，即使我们只传送了最后一个记录的第一项。这是因为在记录分段的情况下，此功能得到下个记录的 RBA 值。只有在它用于用户分段文件中这样做时，它才得到当前文件位置。如果你返回到第八章，你就会发现，没有方法可从系统取回当前文件位置，也没有方法取得 REMBA。有人或许通过 PEEK 等语句找到一个方法，但是事实上 BASIC 本身不能简单直接地告诉你。

为了显示我们已确实定位于记录的第二项，即整数项，我们将只读出该字段来看一看。打入：

```
GET1,*,,I; : PRINT : GOTO 10
```

系统将响应：

```
 2
-1 $  EOF TEST        171  % EOF RBA
171 ! NEXT RCD RBA    152 # REMRA
```

你注意到上面 GET 语句中“I”的变量类型了吗？它是单精度型的，但是传送给它的文件项是整数。文件项与变量之间的类型变化是允许的，只要它在 BASIC中是允许的。

现在，让我们回过头来使用 REMBA 在数据传送前定位文件，传送整数和单精度项。打入：

```
GET 1,$,,K,J; :  PRINT J; K : GOTO 10
```

系统响应：

```
 4    2
-1 $ EOF TEST    171 % EOF RBA
171 ! NEXT RCD RBA    152 # REMRA
```

REMBA 在前面的 GET 操作开始时就被置成文件的 RBA（即第三个记录第二项的开始字节）。不管已经传送的或跳过的字段数目，总记忆着开始字节的 RBA。另外，LOC 函数没有一个被改变。

为了证明 REMBA 没有被多个文件项的传送改变，让我们传送一个整数和双精度项。打入：

```
GET 1,$,,J,,I; : PRINT I;J : GOTO 10
```

系统响应：

```
 8    2
-1 $ EOF TEST    171 % EOF RBA
171 ! NEXT RCD RBA    152 # REMRA
```

注意，在定位中通过忽略 IGEL 中的一个变量名，这里是一个单精度项，就跳过了该文件项。另外，两个文件项都改变了它们的类型，由于它们被送入了不同类型的变量。

现在，我们要进行一些 RBA 定位，看看它是怎样工作的。打入：

```
GET 1,!0,,A$; : PRINT A$ : GOTO 10
```

系统将响应：

```
ABCDEF
0 $ EOF TEST    171 % EOF RBA
19 ! NEXT RCD RBA    0 # REMRA
```

使用程序员提供的指定RBA，无论它是一个数字（如本例中那样）还是某个变量内容或一个表达式，都使RBA移到下个记录指针，正象使用fp=＃时REMRA被移到那儿一样。从这儿往下操作的顺序就与刚才所说的两个fp（即 $ 和!rba）的操作顺序一样。

让我们试一试别的RBA定位技术。打入：

```
I=152 : GET 1,!$I : GOTO 10
```

系统将响应：

```
 0 $ EOF TEST    171 % EOF RBA
152 !  NEXT RCD RBA
RAD FILE MODE IN 30
```

你能想象出发生了什么事吗？你一定能够想到的！这是因为GET语句使用了fp类型，系统就把REMRA和REMBA都标记为无效，所以在显示文件位置的程序中，由30行的LOC (1)＃函数得到了 “BAD FILE MODE” （文件方式不对）信息。这种GET语句除了设置下个记录指针以外什么事也没有做，也没有发生数据传送。

现在，我们想再试一下，但是这次用“过后的”数据传送。打入：

```
GET 1,!$19 : GET 1,,,A$ : PRINT A$: GOTO 10
```

系统将响应：

```
OUT OF STRING SPACE
```

为什么会发生这一个错误（字符串空间不足)?因为我们开始举例讨论时，曾执行过命令CLEAR 50，而我们现在要传送的字符串有130个字节，当然字符串空间就不够了。不要忘记，NEWDOS/80不改变指向缓冲区的字符串变量的指针，但是把字符串移到内存顶端的BASIC字符串空间，好象已执行了LET语句似的。现在，打入：

```
GET 1,,, (10) A$; : PRINT A$ : GOTO 10
```

系统响应为：

```
* * * * * * * * * *
0 $ EOF TEST    171% EOF RBA
152 ! NEXT RCD RBA    19 ＃ REMRA
```

注意：如前一个GET操作那样输入了同一个文件项。由于发生错误，已把文件区域而不是数据恢复成前一个GET操作的开始时它所处的状态。注意，由于变量名的前缀(len)，只有文件项中120个星号中的头10个被传送给了A$。

以上仅仅是有关我们可以使用的fp值的详细讨论。尚未涉及的一个（即fp=rn）已在第八章中相当详细地讲过。现在应该试试记录的修改了（整个地和部分地修改)。但是，在我们进行修改以前，必须打开文件，以便输入和输出。打入：

```
CLOSE : OPEN "R", 1, "MU/DAT", "MU" : GOTO 10
```

系统将响应：

```
0 $ EOF TEST    171 % EOF RBA
0 ! NEXT RCD RBA
```

这就象仅仅为输入而打开文件以后那样。我们刚才指定的方式允许扩展文件（我们将马上这样做)。如果我们不希望超过文件现有的EOF扩展文件，我们应指定方式"D"。

首先，让我们简单地用一个字段来替代文件中的第一个记录。打入：

```
PUT 1,,, "RECORD REPLACED"; : GOTO 10
```

系统响应为：

```
0 $ EOF TEST    171 % EOF RBA
19 ! NEXT RCD RBA    0 # REMRA
```

注意，下个记录指针正指向第二个记录，就象执行一个 GET 操作那样。

现在，让我们用 3 乘以第三个记录中的双精度值的补数来替代第三个记录中的这个双精度值。打入：

```
I=152 : GET 1, !I,,,,;
GET 1,*,,D#; : PUT 1,$,, -D#*3 : GOTO 10
```

系统将响应：

```
-1 $ EOF TEST    171 % EOF RBA
171 ! NEXT RCD RBA    152 # REMRA
```

另外，系统已准备好处理下一个记录，即使它的位置是在EOF。我们不能从这个文件位置传送任何信息，但是可以把增加的新记录（第四个记录）写入文件。

为了演示这种文件扩展的处理方法，打入：

```
PUT 1,,, "THIS IS THE FOURTH RECORD"; : GOTO 10
```

系统将响应：

```
-1 $ EOF TEST    198 % EOF RBA
198 ! NEXT RCD RBA    171 # REMRA
```

由此，很容易看出该文件已经被扩展了。你应该知道，新的EOF尚未记录到目录的FPDE中。假如这时停电的话，那么我们的这个小小的例文件将向你表明，从我们为了修改而打开它以来，它没有变化。我们可以深信，通过使用等于 & 的 fp 进行 PUT 操作，文件中就有了一个新记录。但这仅仅把缓冲区写入了文件。要修改 FPDE 的 EOF 值，必须执行 CLOSE 或 PUT fan, && 语句。CLOSE 也把修改后的缓冲区写入文件。

现在，让我们回到第二个记录，并且用几个更小的文件项来替代它的一个文件项。我们将用一对 PUT 语句来做这项工作。打入：

```
PUT 1,!19,, "ITEM 1",3.14159*2;
PUT 1,*,, "ITEM 3", 4, 10D2;
PUT 1,*,, "LAST ITEM IN RECORD 2"; : GOTO 10
```

系统将响应：

```
0 $ EOF TEST    198 % EOF RBA
152 ! NEXT RCD RBA    19 # REMRA
```

下个记录指针再次得到我们正在处理的记录后面的一个记录的 RBA，并且，REMRA得到最后处理的记录的 RBA。注意，所有三个 PUT 语句都把项目写入同一个记录。

为了显示记录已经被修改了，打入：

```
GET 1,#,, A$, I, B$, J, K, G$; : PRINT A$,B$,C$, I,J,K : GOTO 10
```

系统将响应：

```
ITEM 1    ITEM 3    LAST ITEM IN RECORD 2
6.28318  4  1000
```

0 $ EOF TEST 198 % EOF RBA

152 ! NEXT RCD RBA 19 # REMRA

这是个极妙的结果，不是吗？如果我们用 fp=＊ 的 GET 语句去读取更多数据，那么我们就会看到 “RECORD OVERFLOW”（记录溢出）错误信息。我们可以给这个特定的记录添加更多的数据（假如我们想这样做的话），只要我们不超过它原先的 131 字节总长度。

现在，剩下的唯一要做的事情是修改磁盘上的 EOF 值。要做这件事情，只要打入：

CLOSE

在此应注意，我们可以使用语句：

PUT 1, &&

来修改目录中的 EOF 而不关闭文件。于是，我们可以继续处理该文件。

让我们再次使用 SUPERZAP 程序来检查此文件，现在，你将在位移量 0，13H,98H 以及 ABH 处找到四个记录的 SOR 字节，接着检查第一个记录。字符串标记字节(8FH) 表明长度为15个字节。字符串的开始位移量加 FH 可得到 11H，在该位移量，你就会发现两个 00H 字节的第一个。这两个字节是填充字节，它们在处理 GET 时被系统跳过。假如我们试图由第一个记录取回两个字符串，就象我们在小小的修改处理以前那样，将得到 “RECORD OVERFLOW”（记录溢出）错误的信息，因为在记录中现在只有一个字符串项。当系统发现要写入记录的数据字节少于开始时记录所具有的字节时，系统就用填充项填满逻辑记录。

在第二个记录中，由位移量 13H 开始，你将发现 SOR 字节，其后是定义 6 个字节字符串的标记字节。从该标记字节起往下到第 7 字节，你就会发现定义单精度数值的标记字节。进一步往下数 5 个字节就到达定义另外 6 个字节长的字符串的标记字节。从这个字节往下 7 个字节是定义一个整数的标记字节。由此往下 3 个字节是双精度数标记字节。从此往下 9 个字节是记录中最后一项的标记字节(95H)，它定义了一个 21 个字节长的字符串。该字符串以后（位移量49H）到位移量 97H 的记录的剩余部分都装填了填充字节。如果必须全部用新的数据来替代记录 2，那么此新记录包括 SOR 字节在内将长达 133 个字节。如果使用了合适的 fp，那么其右边存在的所有内容都被替换了。

剩余的记录应是不说自明的。第三个记录的最初内容与现在内容之间的唯一差别是在位移量 A2H 的双精度数现在已修改成－24。而新的第四个记录开始于ABH,其最后字节在C5H。

修改后的 ″MU/DAT″ 文件内容如下所示：

位移量	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	
字节内容 (记录 1)	70	8F	52	45	43	4F	52	44	20	52	45	50	4C	41	43	45	44	00	00	
注释	SOR	MK	″RECORD REPLACED″																pad	
位移量	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	
字节内容 (记录 2)	70	86	49	54	45	4D	20	31	73	CF	0F	49	83	86	49	54	45	4D	20	
注 释	SOR	MK	″ITEM 1″						MK	6.28318				MK	″ITEM 2″					
位移量	26	27	28	29	2A	2B	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39
字节内容	32	72	04	00	74	00	00	00	00	00	00	7A	8A	95	4C	41	53	54	20	49
注释		MK	4	MK		10D2								MK	″LAST I″					

位移量	3A	3B	3C	3D	3E	3F	40	41	42	43	44	45	46	47	48	49 · · · ·97
字节内容	54	45	4D	20	49	4E	20	52	45	43	4F	52	44	20	32	00 00 · ·00
注释					"ITEM IN RECORD 2"											pad

位移量	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA
字节内容(记录3)	70	80	72	02	00	73	00	00	00	83	74	00	00	00	00	00	00	C0	85
注释	SOR		MK	MK	2		MK		4			MK				−24			

位移量	AB	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD
字节内容(记录4)	70	99	54	48	49	53	20	49	53	20	54	48	45	20	46	4F	55	52	54
注释	SOR	MK							"THIS IS THE FOURTH RECORD"										

位移量	BE	BF	C0	C1	C2	C3	C4	C5	C6
字节内容	48	20	52	45	43	4F	52	44	
注释							EOF		

当然，这一讨论没有显示出MU 文件所能做的一切。但它显示了NEWDOS/80 BASIC文件支持的内部的许多性能。对于那些具有数据库经验的用户来说，部分记录I/O应该是熟悉的，它只是修改记录中单个字段的许多数据库性能之一。假如NEWDOS/80不具备数据库的内部文件项的保密性，那么这就是你必须根据自己的要求在你的系统中建立的东西。但是，由以上讨论，你必须承认NEWDOS/80的性能是非常优良的。

当你第一次进行这些文件处理时，不要被这些可用方法的表面复杂性吓倒。最好的方法是象我们刚才所做的那样，进行练习和实践。通过你的实践，这些概念就变得容易理解和融会贯通于工作之中了。

9.9 MF文件

上面我们已经用MU文件类型进行了一些试验，并对NEWDOS/80的性能增加了一点了解。现在，我们继续用MF文件类型作一些试验。翻到第八章，你将发现MF文件是由带标记的文件项组成的，并且是所有记录具有相同长度的记录分段文件。换言之，它是一个标记项、固定记录长度的文件。记录的长度通过OPEN语句中的lrecl操作数由系统定义。

象MU文件类型那样，MF文件类型可以以记录为基础用记录中的新数据项进行修改。修改的数据与原先的数据在数据类型或长度上不必相同，在修改后的记录中也不必与修改以前一样有相同数目的文件项。在MF文件修改过程中，你必须留心使用的文件位置，就象MU文件那样。修改可以从记录的中间开始，并象由开头那样容易。对于MF文件，你可以使用的fp控制与MU文件一样。在修改标记项文件时，不要忘记这样的事实：从当前文件位置到记录结束的所有字节都要重写，而不管你是否真正希望如此。

我们将用MU文件中使用的同一技术对MF文件进行试验。首先，我们必须建立一个文件，作为我们试验的基础。输入下列BASIC程序，并在今后再需要它的情况下把它存起来。

```
10  OPEN "O", 1, "MF/DAT", "MF", 20
20  PUT 1,,, "STRING1", "STR 2", "STR3";
30  PUT 1,,, "MAXIMUM STRING (19)";
40  I!=4 : I#=8 : I%=2
50  PUT 1,,, I#, I!, I%;
60  PUT 1,,, I#*10, I!*100, I%*1000;
70  CLOSE
```

现在，运行此程序以建立文件 "MF/DAT"。当它完成以后，运行 SUPERZAP 程序，使用 DFS 功能来显示刚建立的文件的扇区 0 。你应注意的第一件事情是，在扇区的开头不存在SOR 字节。这是因为只有 MU 文件使用它来定义记录的开始，这些记录都被认为具有不同的长度，而其他记录分段文件类型都有固定长度的记录，因此系统“知道”每个记录从哪儿开始。该扇区中此文件的四个记录的详细内容如下所示。

"MF/DAT"文件内容：

位移量	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
字节内容 (记录 1)	87	53	54	52	49	4E	47	31	85	53	54	52	20	32	84	53	54	52	33	00
注释	MK	"STRING1"							MK	"STR 2"					MK	"STR3"				pad

位移量	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
字节内容 (记录 2)	99	4D	41	58	49	4D	55	4D	20	53	54	52	49	4E	47	20	28	31	39	29
注释	MK	"MAXIMUM STRING (19)"																		

位移量	28	29	2A	2B	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	3A	3B
字节内容 (记录 3)	74	00	00	00	00	00	00	00	84	73	00	00	00	83	72	02	00	00	00	00
注释	MK	I#=8							MK	I!=4				MK	I%=2			pad		

位移量	3C	3D	3E	3F	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50
字节内容 (记录 4)	74	00	00	00	00	00	00	20	87	73	00	00	48	89	72	D0	07	00	00	00	
注释	MK	I#*10=80								MK	I!*100=400				MK	2000		pad		EOF	

其中，第一字节（位移量00）是描述 7 字节长字符串的标记字节。在位移量 08 是描述 5 字节长字符串的标记字节。在位移量 EH 是描述 4 字节长字符串的标记字节。往下前进到位移量 13H，在此你将发现一个填充字节而不是一个标记字节。请记住，我们在文件中建立的记录是20字节(14H)长。在记录 1 中，我们分别写了长度为 7、5、4 字节的三个文件项，总共用了 19 个字节，因此最后要用一个填充字节来填满此 20 字节的记录。

第二个记录开始于位移量 14H，在此你将发现一个 19 个字节长的字符串标记字节(99H)，这整个记录就只有一个文件项。第三个记录在位移量 28H 开始，你将在 28H,31H 和 36H 等位置发现分别描述双精度项、单精度项和整数项的标记字节。第三个记录的总数据长度为17字节，因此需要 3 个填充字节，这可以在位移量 39H 到 3BH 找到。我们写入的第四个记录（最后一个记录）的数据结构与第三记录相同。它的标记字节位于位移量 3CH,45H和 4AH。

超过4FH范围的数据是不可预测的。其实，在我们建立文件以前，并不管扇区中是什么内容。

返回到BASIC并且恢复原来在MU文件试验时使用的定位显示程序。它应是如下这样：

```
10  PRINT LOC (1) $; "$ EOF TEST      "; LOC (1)%; "% EOF RBA"
20  PRINT LOC (1)!;"!NEXT RCD RBA      ";
30  IF LOC (1)!=0 PRINT ELSE PRINT LOC(1)#; "# REMRA"
```

我们将以MU文件试验中所采用的相同方法使用这个程序，显示在文件位置上GET和PUT操作的结果。我们从头至尾要做的试验与MU文件中所做的不完全相同。它们将涉及两种文件类型之间的主要差别。

首先，我们应打开文件，并检查LOC函数的结果。打入：

```
OPEN "I", 1,"MF/DAT", "MF", 20 : GOTO 10
```

系统将响应：

```
0 $ EOF TEST    80 % EOF RBA
0 ! NEXT RCD RBA
```

除EOF的RBA以外，此结果与MU文件的相同。系统已准备好处理由位移量0开始的记录，即第一个逻辑记录。

现在打入：

```
GET 1,,,, A$, B$; :  PRINT A$, B$ : GOTO 10
```

系统将响应：

```
STR 2    STR3
0 $ EOF TEST    80 % EOF RBA
20  ! NEXT RCD RBA    0 # REMRA
```

注意，我们传送了记录的最后两项，这是因为在GET语句中第一个变量名应该正常出现的地方（第三个逗号后面）为空。

使用REMRA定位，我们可以从当前文件位置退回去，再传送记录的前两项。打入：

```
GET 1,#,,A$,B$; : PRINT A$,B$ : GOTO 10
```

系统响应为：

```
STRING1    STR 2
0 $ EOF TEST    80 % EOF RBA
20 ! NEXT  RCD RBA    0 # REMRA
```

在此不用什么技巧，如MU文件中那样，我们可以继续处理同一个记录。打入：

```
GET 1,*,,C$; : PRINT C$ : GOTO 10
```

系统响应：

```
STR3
0 $ EOF TEST    80 % EOF RBA
20 ! NEXT RCD RBA    0 # REMRA
```

等于"*"的fp值告诉系统从上一个GET或PUT操作完成后系统离开的地方往下继续处理，换言之，从当前文件位置往下继续处理。如果刚才的GET操作要求两个或更多的文

件项，就会发生记录溢出错误，这是由于该记录在此还仅仅含有一个文件项。

现在让我们来处理第四个逻辑记录，而不首先处理第二个或第三个记录。打入：

```
GET 1,!(4-1)*20,,J,K,L; : PRINT J; K; L : GOTO 10
```

系统将响应：

```
80    400    2000
-1 $  EOF TEST  100% EOF RBA
100 ! NEXT RCD RBA    80 # REMRA
```

注意，使用 !rba 类型的 fp 表达式指定了一个等于 60 的值。在此表达式中数字本身代表我们实际想要处理的逻辑记录号(4)，减 1，乘记录长度(20)。MF 文件（或者 FF 文件）中的 ! rba 定位，如你所看到的那样，是十分简单的。就象需要你亲自动手做一些事情那样，请你试一试刚才打入的同一行语句，但使用 fp 的 rn 格式代替 !rba 格式。要这样做时，应把 GET 语句改成：

```
GET 1,4,,J,K,L;
```

现在让我们准备对记录作一些简单的随机修改，并检验其结果。为此通过打入以下语句准备好文件：

```
CLOSE : OPEN "R", 1, "MF/DAT", "MF", 20 : GOTO 10
```

系统响应：

```
0 $ EOF TEST   80 % EOF RBA
0 ! NEXT RCD RBA
```

这完全与打开文件作输入时的结果相同。另外，也完全在意料之中。

首先，让我们替代第一个记录。打入：

```
PUT 1,,,I$; : GOTO 10
```

系统将响应：

```
0 $ EOF TEST   80 % EOF RBA
20 ! NEXT RCD RBA    0 # REMRA
```

此响应表明第一个逻辑记录已处理过了。应知道，即使下个记录的 RBA 显示为数值20，其实当前文件位置是等于 1，因为上述 PUT 操作把记录的整个内容用一个空字符串（仅一个标记字节 80H）和 19 个零字节替代了，然后返回去重新定位文件于空字符串(标记字节)后面的字节处。假如我们用 fp=* 在当前文件位置写数据，那么 PUT 操作的第一个字节（标记字节）将放在文件的第二个字节中。

如做游戏一般，让我们在刚修改的记录上添加两个字段。打入：

```
PUT 1,*,,"2",2; : GOTO 10
```

系统响应：

```
0 $ EOF TEST   80 % EOF RBA
20 ! NEXT RCD RBA    0 # REMRA
```

我们将立即看到这最后修改的结果。但在此以前，让我们在文件的末尾再添加两个记录，打入：

```
PUT 1,!%,,"RCD 5"; : PUT 1,,, "RECORD 6"; : GOTO 10
```

系统响应为：

```
-1 $ EOF TEST    120 % EOF RBA
120 ! NEXT RCD RBA    100 # REMRA
```

这些数字表明文件现在有6个记录。

现在，关闭文件并输入运行SUPERZAP程序，使用DFS功能显示文件的扇区0。文件中的记录分别开始于位移量0，14H，28H，3CH，50H和64H。位于位移量0的标记字节描述了一个空字符串；在位移量1处的标记项描述了一个1字节长的字符串；在位移量3处的标记字节描述了一个整数。注意，该记录的剩余部分（从位移量6开始到13H）已经用填充字节(00H)填充。由于我们只修改了第一个记录，所以第二至第四记录不变。第五和第六记录是新添加的，它们的内容就不必讲述了。你应注意到扇区中第六个记录以外的数据没有被我们小小的修改过程改变。系统忽略扇区的这个区域，因为这是位于和超过文件EOF的空间，因此它不是文件的真正部分。

通过以上MF和MU文件的比较，你现在应知道MF文件一点儿也不难运用。你可以使用!rba定位（如本例所示那样），或者使用记录号本身（fp=rn的定位）来取回各个记录进行修改。

9.10 MI文件

现在我们到了最后一个标记项文件——MI文件类型。它与MU文件类型和MF文件类型的最大差别是：

(1)MI文件不能修改。

(2)MI文件没有系统可识别的记录长度。

这些差别限制了这个文件类型只能用于密集的参照文件，因为它们只能写入或扩展，然后再读出。也可以用随机存取方式得到任何指定的数据组或项，但必须使用!rba定位（或它的逻辑等效值）。

由于你现在已经相当详细地了解了标记项文件，所以我们在此只进行有限的文件存取试验，并且准备强调结构和存取方法上的差别而不是相似性。

首先，恢复我们在建立MF文件时使用过的程序并把它改成如下这样：

```
10  OPEN "O", 1, "MI/DAT", "MI"
20  PUT 1,,, "STRING1", "STR 2", "STR3";
30  PUT 1,,, "MAXIMUM STRING (19)";
40  I!=4 : I#=8 : I%=2
50  PUT 1,,, I#, I!, I%;
60  PUT 1,,, I#*10, I!*100, I%*1000;
70  CLOSE
```

注意，这个例程序中与MF文件中例程序的不同之处只有行10的OPEN语句作了改变，其他各行都相同。

如果你愿意的话，就把此程序存起来，并运行此程序。于是就建立了一个用户分段文件，它含有73个字节的数据，与上节建立的MF文件多少有点不一样（没有用于填满一定记录长度的填充字节）。现在让我们退出BASIC并输入SUPERZAP程序，用DFS功能显示刚建立的MI文件的扇区0。

你将看到，在扇区中没有任何SOR标记字节和填充项。就BASIC来说也不存在什么记录，而只是一串带着标记字节的数据项。文件中的数据及其结构和组织，完全是程序员的职责，一个好的数据设计要求数据项有完全可用的合理的相关的数据结构。

在我们建立的这个文件中，所有的内容都是互不相关的数据项。要顺序地存取它们就需要深切了解：文件中有四个字符串项和六个数字项。要随机地存取它们，则需要知道某文件项标记字节的具体的RBA。否则，最好的情况是，系统产生"BAD FILE DATA"（文件数据有问题）错误，糟的是系统得到不相干的数据。

现在，让我们查看一下SUPERZAP程序显示出的扇区内容（可参考9.9节"MF/DAT"文件的内容)。字符串项的标记字节分别位于位移量0,8H,EH和13H。第一组数据项的标记字节在27H,30H和35H；第二组数据项的标记字节位于38H,41H和46H。一会儿我们将使用这些数（位移量，也就是RBA值）来存取数据。另外，文件EOF的RBA在49H。

这时，我们返回到BASIC READY，装入你存在磁盘上的在MU和MF文件试验中使用的文件位置显示程序。这个程序有助于显示系统缺乏的对MI文件的逻辑记录支持。

文件照例必须打开才能存取。打入：

```
OPEN "I", 1, "MI/DAT", "MI" : GOTO 10
```

系统将响应：

```
0 $ EOF TEST    73 % EOF RBA
0 ! NEXT RCD RBA
```

与其他文件类型中一样，输入方式（"I"）的OPEN操作对系统进行定位，于是，如果使用空缺fp值，则要处理的下一个字节就是文件的第一个字节。

为了显示OPEN操作对于扩展文件时产生的不同定位，打入：

```
CLOSE : OPEN "E", 1, "MI/DAT", "MI" : GOTO 10
```

系统响应为：

```
-1 $ EOF TEST    73 % EOF RBA
73 ! NEXT RCD RBA
BAD FILE MODE IN 30
```

这最后一个信息是由于以下原因：位置显示程序试图在系统刚刚把REMRA标记成无效时显示输出REMRA值，这是OPEN操作产生的结果（因为下个记录的RBA为非零，位置显示程序试图显示REMRA)。

文件现在是在输出方式。为了证明这一点，我们将用三个整数项来扩展文件。打入：

```
PUT 1,,, -1,-2,-3; : GOTO 10
```

系统将响应：

```
-1 $ EOF TEST    82 % EOF RBA
82 ! NEXT RCD RBA    73 # REMRA
```

注意，EOF的RBA在文件中比原来高了9个字节，并且REMRA是原先的EOF RBA值。在MI文件处理中，REMRA总是被置成与REMBA相同的值，它们都等于PUT或GET操作开始数据传送时的文件位置。

现在，让我们回过头来从文件中读取一些数据项。打入：

```
CLOSE : OPEN "R", 1, "MI/DAT", "MI"
```

```
GET 1,!19,, A$; : PRINT A$ : GOTO 10
```

系统将响应：

```
MAXIMUM STRING (19)
0 $ EOF TEST    82 % EOF RBA
39 ! NEXT RCD RBA    19 # REMRA
```

REMRA 反映了 GET 操作开始时的 RBA，而下个记录的 RBA 指向文件中的第一个数字项。如果不指定取代的 fp，则它就是下一个 GET 操作开始传送文件项的位置。为显示这一点，打入：

```
GET 1,,,, J%, K#,I!; : PRINT J% ; K#; I! : GOTO 10
```

系统将响应：

```
4    2    80
0 $ EOF TEST    82 % EOF RBA
65 ! NEXT RCD RBA    39 # REMRA
```

注意，GET 语句中跳过了第一个数字项，并且 IGEL 中所有项目的数字类型再次与传送给它们的文件项的类型不同。标记项文件的固有能力之一是这种数字类型的转换。

为了显示在 MI 文件中 REMRA 和 REMBA 是相同的，我们必须用相应的 fp 值把基本相同的工作重复做两次。首先打入：

```
GET 1, #,,I,J,K; : PRINT I;J;K : GOTO 10
```

然后输入：

```
GET 1,$,,I,J,K; : PRINT I;J;K : GOTO 10
```

在这两种情况下，系统都响应：

```
8    4    2
0 $ EOF TEST    82 % EOF RBA
56 ! NEXT RCD RBA    39 # REMRA
```

这就是我们所要证明的。不要忘记以下事实：这个 REMRA 等于 REMBA 的关系，对于字段项、MI 和 FI 文件，在任何时候都是真的；而对于 MU、MF 和 FF 文件，只有在GET 或 PUT 操作的数据传送开始于逻辑记录的开头时才是真的。

现在，要展示 MI 文件可以在用方式 "R" 打开后进行扩展。打入：

```
PUT 1,!%,,15,-15; : GOTO 10
```

系统将响应：

```
-1 $ EOF TEST    88% EOF RBA
88 ! NEXT RCD RBA    82 # REMRA
```

不出所料，EOF 已往后扩展了 6 个字节。如果使用 fp=（空缺）或*，文件就依然定位在当前文件位置不变，以便继续把数据添加在文件的末尾。

这就是关于我们在 MI 文件中可以进行的试验的详细讨论。你还可以亲自修改一个现有的文件项试一试（你一定会得到 "BAD FILE MODE"——文件方式不对的错误，因为第八章已指出 MI 文件不能修改）。如果在 MI 文件已用某种方式打开并且在 GET 或 PUT 语句中指定了某个 fp 值时，你不相信会发生什么事情，那么在 BASIC 的计算器方式中用一个小小的文件在这种情况下试一试，这是发现到底会发生什么事情的最好的方法。

9.11 FF文件

固定项文件有几个方面与标记项文件不同。首先，它没有每个项目或记录的标记字节，所有项目的描述都是由IGEL而不是由文件进行的。由于这个原因，如果你描述一个20个字节的字符串项，以进行读操作，那么即使原先写入文件的数据是数字，它也完全按照你的描述去读。因此，它也要求写入文件的数字项要以相同的类型去读出，否则文件数据就失去了原有的意义。

第二个主要差别是固定项文件可以用真正的部分记录I/O进行修改。这就是说，修改固定项记录中的一个字段可以不影响周围的任何字段；而在标记项文件中，修改好的字段和记录结束以前的所有其他字段都必须一起写入，否则影响被修改字段以后的记录内容。

第三个重大差别是IGEL中的表达式不能是变量名以外的任何东西，对于字符串项它必须带有长度前缀(len)。这是因为会由表达式得到不明确的项目类型或长度。

固定项文件有两种类型：FF文件，其中所有的记录都有相同长度；FI文件，其中没有BASIC可检测的记录。一会儿我们将只涉及FF类型文件。

如标记项文件的讨论那样，我们先要建立一个FF文件 "FF/DAT"，然后在"计算器方式"用它进行试验。输入以下程序，并按你的愿望存起来。最后运行此程序建立FF文件。

```
10  CLEAR  100
20  OPEN  "O",1,"FF/DAT",  "FF",  20
30  PUT  1,%,40 : GOTO 50
40  (20) I$;
50  LSET  I$ ="ABCDEFIJK"
60  PUT  1,,,  (20)I$;
70  LSET IS="123456789012345678901"
80  PUT  1,,,  (20)I$;
90  I%=2 : I!=4 : I#=8
100  PUT  1,,,  (4)I$, I%, I!, I#;
110  I%=I% 10 : I!=I! 100 : I#=I# 1000
120  PUT 1,,, (4)I$, I%, I!, I#;
130  CLOSE
```

你应该注意到，这个程序与标记项文件中使用的程序有点不同。一是字符串项由变量写入，而不是IGEL本身中的文字。其次，没有使用IGEL表达式把数据放入文件。在给变量I$赋值时，使用LSET而不是LET。这使后面进行的工作保持了I$的长度，I$的长度是行30和行40中进行的伪FIELD操作建立的。如果你的程序可以使用于以下情况：当装填从文件接收数据的变量时，在为文件提供字符串数据的变量右边不进行填充（填入00字节）以达到规定长度，那么就不需要这个伪FIELD操作。也请注意，IGEL中的字符串项都有长度前缀。正是这些前缀实际确定了字符串数据的多少个字节要传送给文件或由文件传送回来，而不是由伪FIELD操作确定（见行100和行120）。记住，文件的字符串项将按需要在右边填充或截尾，以适合长度前缀的要求。

如果我们决定不使用伪FIELD功能，程序应写成：

```
10  CLEAR  1000
20  OPEN "O", 1, "FF/DAT", "FF", 20
50  I$ ="ABCDEFIJK"
60  PUT  1,,,  (20)  IS;
70  I$ ="12345678901234567890"
80  PUT  1,,,  (20)I$ ;
……（以下程序行与上述程序相同）
```

现在，运行 SUPERZAP 程序，使用 DFS 功能检查刚写的 "FF/DAT" 文件的扇区 0。你将看到第一个记录（14H 字节长）中在 9 个传送的数据字节后面已经用11个填充项填满了20个字节（在上述第一个程序中，装填填充项是由于使用了 LSET 语句；而在第二个程序中是由于 PUT 操作的字符串前缀)。第二个记录没有填充项——我们要写的字符串项正好就是20个字节（假如它比 20 个字节长，那么第一个程序中的 LSET 语句会把变量 I$ 的右边截尾；在第二个程序中的 PUT 语句根据 I$ 的前缀把文件项截尾)。第三个记录和第四个记录具有相同的格式：4 个字节的字符串，2 个字节的整数值，4 个字节的单精度值，8 个字节的双精度值和 2 个填充项。再次提醒读者注意，在此 FF 文件的记录中，没有描述记录开头的 SOR 字节，也没有描述文件项类型的标记字节。文件的 EOF 在位移量80(50H) 处。位于或超过这个文件位置的扇区中的字节都没有被这个程序的运行改变，因为这些字节不是文件的组成部分。

"FF/DAT" 文件的详细内容：

位移量	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
字节内容（记录 1）	41	42	43	44	45	46	49	4A	4B	00	00	00	00	00	00	00	00	00	00	00
注释	"ABCDEFIJK"									pad										

位移量	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
字节内容（记录 2）	31	32	33	34	35	36	37	38	39	30	31	32	33	34	35	36	37	38	39	30
注释	"12345678901234567890"																			

位移量	28	29	2A	2B	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	3A	3B
字节内容（记录 3）	31	32	33	34	02	00	00	00	00	83	00	00	00	00	00	00	00	84	00	00
注释	"1234"				I%=2		I!=4				I#=8								pad	

位移量	3C	3D	3E	3F	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50
字节内容（记录 4）	31	32	33	34	14	00	00	00	48	49	00	00	00	00	00	00	7A	8D	00	00	
注释	"1234"				I%=20		I!=400				I#=8000								pad	EOF	

现在，重新装入 MU 文件试验中用过的用于显示 LOC 函数的结果的程序，我们将再次用它来演示如何维护文件位置。

为了说明在文件打开以后的文件位置，打入：

```
OPEN "I",1,"FF/DAT", "FF", 20 : GOTO 10
```

系统将响应：

```
0  $  EOF TEST     80% EOF RBA
0  !  NEXT RCD RBA
```

不出所料，系统已被定位，以便处理文件的下一个记录（第一个记录）。

为了传送第一个记录，打入：

```
GET 1,,, (20) A$ ; : PRINT LEN (A$ ); A$  : GOTO 10
```

系统响应为：

```
20  ABCDEFIJK
0  $  EOF TEST     80% EOF RBA
20  ! NEXT  RCD  RBA     0 #  REMRA
```

正如你所见到的那样，传送了 20 个字节给 IGEL 中命名的变量。我们可以如此容易地传送部分记录，如果我们想进行的话。

正是为了显示部分记录传送是如何完成的，我们假设记录由三个 6 字节的文件项组成，并用独立的 GET 语句分别传送它们。当然，我们必须使用某些特殊的 fp 值来完成这一任务。打入：

```
GET 1,#,, (6) A$ ; : GET 1, *,, (6) B$ ; : GET 1, *,, (6) C$ ;
PRINT LEN(A$ ); A$ :PRINT LEN (B$ );B$ : PRINT LEN (CS);C$ : GOTO 10
```

系统将响应：

```
6  ABCDEF
6  IJK
6
0  $  EOF  TEST     80% EOF  RBA
20 ! NEXT  RCD  RBA     0  #  REMRA
```

这儿我们使用了三个 GET 语句从同一个记录读了三个字段。这显示了部分记录 I/O 的可能性之一。

你可利用的另一个可能性是在记录中跳过一些字节去得到你真正想要的字节的能力。现在我们用第二个记录来做这项工作。打入：

```
GET 1,,, (12)$ , (4)A$ ; : PRINT LEN (A$ ); A$  : GOTO    10
```

系统将响应：

```
4     3456
0  $  EOF  TEST     80% EOF  RBA
40 ! NEXT  RCD  RBA     20 #  REMRA
```

我们跳过的 12 个字节也可以是 6 个整数，跳过它们就象跳过这 12 个字节的字符串那样容易。要强调的是系统既不知道也不管跳过了什么项目或数据类型，只管跳过(len)个字节。

现在，我们将在处理第四个记录时犯一个小小的错误——我们暂时忘记了该记录的开头当初是用一个 4 字节字符串写的。打入：

```
GET  1,4,,I%,I!,I#; : PRINT I%;I!; I#  : GOTO 10
```

系统将响应：

```
12849     0     0
-1 $ EOF TEST     80% EOF  RBA
80 ! NEXT  RCD  RBA     60 # REMRA
```

这当然不是我们原先写入的内容！它指出在 FF（以及 FI）文件中对记录描述必须一致的要求。固定项文件处理与标记项文件不同，在标记项文件中，将会发现和报告这个错误，但在固定项文件的处理中，无论什么都可出现在要传送到命名变量的当前文件位置上，而不进行也不可能进行检查来防止这种错误。在上面进行的处理中，我们用 I% 读取了记录开头的字符串中的前 2 个字符"12"，并把它作为整数值处理，所以显示出数字 12849。而单精度数和双精度数在显示器上显示出 0 值的原因是它们的指数字节（位于位移量 41H 和49H）为 0。

你应注意到目前我们正定位于EOF（或者至少表面上如此）。事实上，就系统来说，当前文件位置是第四个记录的第 15 字节（位移量 4AH）。LOC (fan)! 得到待处理的下一个顺序记录开头的 RBA，即在 fp 等于空缺的情况下要处理的那一个。

正是为了显示我们定位于第 15 字节，打入：

```
FOR I=1 TO 6 : GET 1,*,,(1)A$ ; : PRINT ASC (A$); : NEXT
```

系统将响应：

```
0     0     122     141     0     0
```

作一些十进制到十六进制的转换就能看出，非零字节相应地是原先作为记录 4 写入的双精度数的最高有效尾数字节和指数字节。

现在让我们回过来正确地处理记录 4。打入：

```
GET  1,4,,(4)A$,I%,I!,I#;  :  PRINT  A$;I%;I!;I#  :  GOTO 10
```

系统将响应：

```
1234     20     400     8000
-1 $ EOF  TEST     80 % EOF  RBA
80 ! NEXT  RCD  RBA     60 # REMRA
```

这就是当初写入的内容。当然，你已注意到，第四个记录是用 fp 参数的 rn 格式处理的。在开发固定长度记录文件（FF、MF 和字段项）的索引时，可以使用含有记录号的整数项而不是含有由某个 LOC 函数得到的 RBA 值的单精度项来保存一对字节作为索引。对固定记录长度文件的随机存取，使用 RBA 定位完全与使用 rn 定位一样可靠，并且在检查一个索引项的内容时，或许稍微容易理解一些。

现在让我们关闭此文件，并打开它进行修改的试验。打入：

```
CLOSE : OPEN "R",1, "FF/DAT", "FF", 20 : GOTO 10
```

系统将响应：

```
0 $ EOF TEST     80% EOF RBA
0 ! NEXT RCD RBA
```

我们要研究的是部分记录 I/O。在这一方面，固定项文件的能力真正超过了标记项文件。让我们假设文件中的第一个记录和第二个记录具有相同的数据格式：每一个记录有四个 5 字节长的文件项。现在我们要修改第一个记录中的第二项和第二个记录中的最后一项。打入：

```
A$ ="2ND" : PUT 1,1,,((2-1)*5)$,(5) A$ ; : GOTO 10
```

系统将响应：

```
0 $ EOF TEST 80% EOF RBA
20 ! NEXT RCD RBA 0 # REMRA
```

现在，修改第二个记录。打入：

```
A$ ="LAST" : PUT 1,2,,((4-1)*5)#, (5)A$ ; : GOTO 10
```

系统将响应：

```
0 $ EOF TEST 80 % EOF RBA
40 ! NEXT RCD RBA 20 # REMRA
```

你可能已经注意到，对记录中字段的定位是通过计算要跳过的字节数来完成的。在第二个记录中，对最后的5字节字段的定位不是简单地跳过前面15个字节，而且在处理中把它们清零了。我们立即就能看到这后者的影响。

现在，让我们修改第三个记录和第四个记录中的整数项。打入：

```
I% = -50 : PUT 1,3,,(4)$,I% ; : PUT 1,4,,(4)$,I% ;
GET 1,3,,(4)J$,J%,J!,J# ; : PRINT J$,J% ;J!;J# : GOTO 10
```

系统响应为：

```
1234 -50 4 8
0 $ EOF TEST 80% EOF RBA
60 ! NEXT RCD RBA 40 # REMRA
```

响应的第一行表明我们的修改确实只改变了我们想处理的那个字段，记录中的其他字段都没有改变。在具有相似记录的MF和MU文件中（考虑到标记字节），那怕只是为了修改这么一个整数，也必须在IGEL中指定所有的整数、单精度和双精度项，并且该语句也是不完整的，必须首先读出修改项以后的单精度和双精度数值，以保持它们正确的值，在写回修改后的值时要同时把它们一起写回记录中去，或者使用不同的fp值独立地把它们写回去。可是，在此FF文件中，我们只要跳过我们想要跳过的字节，和只写入被修改的那一个字段。

在"R"方式中，可以扩展任何NEWDOS/80文件。为显示这一特性，打入：

```
PUT 1,6,,J%,J%,J% ; : GOTO 10
```

系统将响应：

```
-1 $ EOF TEST 100% EOF RBA
100 ! NEXT RCD RBA 80 # REMRA
```

在这个例子中，我们整个地跳过了文件的第五个记录。为了保持必要的记录方向，系统在写入我们指定的第六个记录以前，建立和写入了内容全部为0的第五个记录。

现在关闭文件，运行SUPERZAP程序，并象以前那样使用DFS功能去查看文件的第一扇区（扇区0）。你就会看到，两个字符串记录（即记录号为1和2的记录）已经按要求修改好了：第一个记录中，只把位移量05—09的字节改成字符串"2ND"和两个填充字节，其余字节都不改变；第二个记录中，把位移量14H—22H的字节改成00，把23H—27H的字节改成字符串"LAST"和一个填充字节。第三个记录和第四个记录中的整数项2个字节（分别位于2CH—2DH和40H—41H）都改成CEFFH（即-50），其余不变。第五个记录（位于50H—63H）全是00。第六个记录（位于64H—77H）的开头含有三个CEFFH，，其后是14个

00字节。

9.12 FI文件

我们现在要讨论NEWDOS/80的独特的文件类型的最后一个：FI文件。象MI文件一样，它是一个用户分段的文件，并象FF型文件一样，它由固定项而不是标记项构成。与MI文件不一样的是，FI文件可以修改。这一特性使它在应用上比MI型文件稍许强一些。

象我们以前在每种文件上所做的那样，我们将通过运行一个BASIC程序来建立一个文件，然后用这个文件在BASIC的计算器方式下进行试验。为了建立文件"FI/DAT"，输入、存储并运行下列程序：

```
10  OPEN "O", 1, "FI/DAT", "FI"
20  A$="1ST STRING" : B$="STR 2"
30  I%=2 : I!=4 : I#=8
40  PUT 1,,,(15)A$,(6)B$,I%,I!,I#;
50  I%=I%*-1000 : I!=I!*-100 : I#=I#*-10
60  PUT 1,,, (15) B$, (6)A$,I%,I!,I#;
70  CLOSE
```

你应注意的第一件事情是，我们根本没有执行伪FIELD操作(fp=%)，就象我们在FF文件中那样，这是因为它不是绝对需要的。BASIC将根据GET语句的需要分配其中的字符串，并且文件支持将根据需要填充或截尾字符串文件项，使它们符合IGEL项前缀所规定的长度。

要注意的第二件事情是PUT操作都产生相同格式的数据组：一个15字节的字符串，一个6字节的字符串，一个整数，一个单精度数和一个双精度数。在较大的文件中，如此一致的数据格式使它对于FF型文件是适用的，由于所有的数据组都具有相同的长度和结构。

装入以前所有试验中使用的用于显示LOC函数结果的程序，并打入：

```
OPEN "R", 1, "FI/DAT", "FI" : GOTO 10
```

系统将响应：

```
0 $ EOF TEST    70 % EOF RBA
0 ! NEXT RCD RBA
```

不出所料，文件已定位于第一个字节位置，于是，如果指定了fp为空缺或*，那么第一个GET或PUT操作就从文件的第一个字节开始。这是方式"E"以外的所有打开方式的情况，在方式"E"中，文件定位于EOF的RBA。

了解我们写入的这两组数据的数据结构，将使得通过RBA定位来存取第二组数据相当容易。打入：

```
GET  1,!35,,(15)A1$, (6)A2$,I%,I!, I#;
PRINT  A1$, A2$, I%; I!;I# : GOTO 10
```

系统将响应：

```
STR 2    1ST ST    -2000  -400  -80
-1 $ EOF TEST    70  % EOF  RBA
70 ! NEXT RCD RBA    35 # REMRA
```

通过处理第二个数据项组中的所有数据，如 LOC (fan)$ 函数所示那样 已把文件定位于EOF。

FI 文件可以象 MI 文件那样以相同方法进行扩展。让我们来试一试吧！在当前文件位置和新的数据之间要留出10个字节。打入：

```
L=LOC(1)! : J=EXP(1) : PUT 1,,, (10)#,J; : GOTO 10
```

系统响应：

```
-1 $ EOF TEST    84 % EOF RBA
84 ! NEXT RCD RBA    70 # REMRA
```

上面语句中的变量 L 含有我们写入的 10 个填充字节的文件位置，一会儿我们将使用它。注意，在 PUT 语句中，变量 J 没有用类型字符作后缀。J 将具有单精度浮点数的缺项类型，并写入文件 4 个字节。虽然不必象 NEWDOS/80 1.0那样，必须在 IGEL 中使用明确的类型后缀字符，但是，我们极力建议你使用明确的类型后缀。

现在，让我们回过来把一些内容放入我们刚写入的填充区域。我们将明确使用RBA定位来到达该文件区域。打入：

```
A$="ABCDEFG" : PUT 1, ! L, , (4) A$, L!; : GOTO 10
```

系统将响应：

```
0 $ EOF TEST    84 % EOF RBA
78 ! NEXT RCD RBA    70 # REMRA
```

请特别注意最后的 PUT 语句中的间隔。它表示在输入程序时允许有间隔。有空隔时或许使程序变得更容易读些，但是否使用空隔应根据个人的爱好和需要而定。

现在让我们来读取一些已写入文件的数据。打入：

```
GET 1,!58,,I!,I#,(4)B$,K!;
PRINT I!;I#;B$;K! : GOTO 10
```

系统响应：

```
-400    -80    ABCD    70
0 $ EOF TEST    84 % EOF RBA
78 ! NEXT RCD RBA    58 # REMRA
```

由于我们用正确的数据类型去读取位于相应文件位置的数据，所以得到的结果是正确的。无须赘言，假如我们在定位中那怕错开一个字节，那么结果就大不一样了。

为了显示粗心的程序员可能遇到的不幸，让我们错位文件并重复刚才的传送。打入：

```
GET 1,!57,,I!,I#,(4)B$,K!;
PRINT I!;I#;B$;K! : GOTO 10
```

系统将响应：

```
2.36125E+21    2147483648.000008    xABC    6.01858E-36
0 $ EOF TEST    88 % EOF RBA
77 ! NEXT RCD RBA    57 # REMRA
```

真够呛！不是吗？但是系统正是按你的吩咐去干的——因为它是一个 FI 型文件，所以不能为我们提供保护。

如果你对本章讨论的内容已经有了充分的了解，那么你就会对如何巧妙地处理标记项文

件和固定项文件产生清楚的概念。NEWDOS/80 还支持 TRSDOS BASIC 的随机文件（字段项文件）和顺序文件(输出/输入文件),除允许字段项文件可以具有一个不同于 256 的标准长度外，NEWDOS/80 中没有什么改变，所以在此不作进一步的讨论。如果你对此尚不熟悉，请阅读第八章和 TRSDOS BASIC 手册，你将会看到 NEWDOS/80 对老的 TRSDOS 文件类型的支持已经扩充。利用你在本章得到的经验，你就能产生自己的适用于那些扩充的文件处理的试验，以此进一步熟悉它们。

第十章　固定项和标记项文件应用举例

本章的目的是给读者提供一些使用标记项文件和固定项文件的例子，作为第八章和第九章的补充说明，并希望在这些文件的使用方法上对读者有所启发。

本章在引用文件类型时将使用文件类型的缩写。另外也将用到诸如 IGEL, RBA, REMRA, REMBA 等这样的术语。对于它们的定义，请参考第八章和附录 D。

这儿给出的大多数例子都是进行 MU 文件和 FF 文件的处理，因为这两种文件类型是最常用的。而且，我们使这些例子尽可能相似或实用，以使读者容易比较和发现各种文件类型的异同之处。

由于我们的基本目的是说明文件的使用方法，因此我们在这些例子中没有提供子程序来实际使用或产生数据（供文件读出或送往文件）。如果读者想在机器上实际地使用这些例子试一试，则请读者自己提供这些子程序。

在所有这些例子中，每个对应文件项的命名变量都用一个明确的类型符号（$、%、! 或 #）作后缀。这样做就使读者明确了解什么类型的数据被读出或写入文件。我们建议读者在今后的文件处理中也这样做，否则，十分可能出现以下情况：你自以为记得隐含的类型是什么而实际并非如此，于是严重地破坏了文件。在 NEWDOS/80 1.0 中，对于用于固定项文件处理的 IGEL, 要求使用明确的类型符号，但是在 NEWDOS/80 2.0 中就不必如此。例如，在例 7 中，若想不使用明确类型符号的 IGEL, 则可以把行 10 和行 120 改成：

```
10  CLEAR  2000 : DEFSTR  N,S : DEFINT  A,I : DEFSNG F: DEFDBL  D
120  (20) NM, AN, AM!, DT, (15) ST, IG, FP, DP;
```

记住，为了安全可靠起见，我们极力建议在 IGEL 中的变量名要带有类型符号的后缀。

GET 或 PUT 的操作必须按以下两个阶段处理：

1. 文件定位阶段

在这一阶段，根据 GET 或 PUT 语句的第二个参数——文件定位参数 (fp)，进行文件的定位。对于某些类型的定位参数，在这一阶段的末尾，当后续的对该文件区域的定位参数是 # 或 $ 时（见 8.10 节），就把文件位置值 REMRA 和 REMBA 保存起来，供后面继续使用。

2. 数据传送阶段

在这一阶段，在文件与 IGEL 中命名的变量之间进行数据的传送。

对于本章所列举的例子，我们希望读者能化费些时间从例 1 开始顺序地阅读一遍，这是因为后面的例子往往使用前面例子所生成的文件，并且对例程序的解释也是前面解释过的内容在后面就不再重复或只是简单地一带而过。

例1. 把记录顺序地写入 MU 文件

可以把 MU 文件看作为输出/输入文件的变种。MU 文件是最常用的一种文件类型，因为它有许多特点。例如：它比输出/输入文件使用更少的磁盘空间；并在某些限制下可以进

行修改；还可以通过 !rba 定位参数进行检索；而且在写入文件时不需要象输出/输入文件那样用；″,″；字符序列把字符串分隔开。

本例的程序清单如下：

```
10  CLEAR  2000
20  OPEN  "O",  1,  "XXX/DAT : 1",  "MU"
30  GOSUB  10000                    'build  data  for  record
40  IF  RN%=0  THEN  CLOSE : END        'end  of  run
50  PUT  1,,,  NM$,  AN%,  AM!,  DT#,  ST$,  IG%,  FP!,  DP#;
60  GOTO  30
```

为了顺序地输出记录，即把记录顺序写入 MU 文件，首先要打开文件（行 20）：存取方式为″O″——顺序输出，文件类型为″MU″——分段成不等长记录的标记项文件。由于 OPEN 语句中没有规定逻辑记录长度 lrecl，记录最长可达 4095 字节。本例中，文件内各记录的长度完全取决于每个记录中含有的两个字符串（行 50 的 IGEL 中的 NM$ 和 ST$）的大小。文件标识符为″XXX/DAT:1″。建立的 I/O 链使用 1 号文件缓冲区。

PUT 语句中的定位参数是一个空缺，它表示每执行一次 PUT 语句就把下一个顺序记录写入文件。

读者要在行号 10000 处提供为记录产生数据的子程序。如果再没有记录要建立，则在返回主程序前令 RN%=0，于是在主程序行 40 关闭文件，结束运行。否则令 RN%≠0，并把要传送给文件的数据放入 8 个变量：NM$，AN%，AM!，DT#，ST$，IG%，FP! 和 DP#。

在这个例子中，IGEL 包含在 PUT 语句中，并由 8 个表达式组成（在此，都是命名的变量）。与每个记录项有关联的变量或表达式要用逗号相互间隔开。IGEL 用分号结束。

字符串 NM$ 和 ST$ 的每一个的全部内容，连同放在每个字符串前面的一个或两个标记字节被一起送入文件。第二个标记字节只有在字符串的长度为 128 到 255 个字符时才使用，这时，第一标记字节为 71H，第二标记字节为字符串的长度。如果字符串长度小于 128，则只有一个标记字节，它的内容为字符串长度加 80H。

整数 AN% 和 IG% 的每一个在文件中是用三个字节描述的：一个 72H 的标记字节，其后是 2 字节的二进制整数值，其格式与 BASIC 使用的整数格式相同。

每一个单精度浮点数 AM! 和 FP! 在文件中用五个字节描述：一个 73H 标记字节，其后是 4 字节的单精度浮点值，其格式与 BASIC 使用的单精度数格式相同。

每一个双精度浮点数 DT# 和 DP# 在文件中用九个字节描述：一个 74H 标记字节，其后是 8 字节的双精度浮点值，其格式与 BASIC 使用的双精度数格式相同。

根据上述说明的各数据类型的字节数，并考虑记录的 SOR 字节和文件项标记字节，我们可以用 PUT 语句中的 IGEL 来计算最小和最大的记录长度，在这个文件中，最小的记录长度是 37 个字节（两个字符串都为空，所以 1+(1+3+5+9)*2=37），最大记录长度是 549 个字节（两个字符串都有 255 个字节，所以 1+(257+3+5+9)*2=549）。

例2. 从 MU 文件中顺序地读取记录

```
10  CLEAR  2000
20  OPEN  "I",  1,  "XXX/DAT : 1",  "MU"
```

```
30  IF  EOF(1)  THEN  CLOSE : END
40  GET  1,,, NM$, AN%, AM!, DT#, ST$, IG%, FP!, DP#;
50  GOSUB  10000                'process  the  record's  data
60  GOTO  30
```

这个例子是例 1 的相反处理过程。它与例 1 使用相同的 IGEL 和相同名称的变量。这时，OPEN 语句中的存取方式为 "I"——顺序输入。文件的数据记录（就是例 1 所建立的）将在行40被连续地读出并赋给相应的变量。读者应在行 10000 提供子程序对比数据进行所需要的数据处理。

在 GET 语句中的文件定位参数是空缺，这意味着每执行一次 GET 语句就读取文件中的下一个顺序记录。

当下个记录的文件位置正好在文件的 EOF 时，EOF (1) 函数就得到真状态 (−1)，于是关闭文件并结束执行（行 30 语句）。

例3. 顺序地读取和修改 MU 文件的记录

```
10  CLEAR  2000
20  OPEN  "R",  1,  "XXX/DAT:1",  "MU"
30  IF  EOF(1)  THEN  CLOSE : END
40  GET 1,, 120          'read the next sequential record
50  GOSUB  10000            'update  the  record's  data
60  PUT  1,#, 120        'rewrite the record back to the file
70  GOTO  30
120  NM$, AN%, AM!, DT#, ST$, IG%, FP!, DP#;
```

在这个例子中，使用例 1 中建立的同一个 MU 文件，但以 "R" 方式打开，以进行顺序的输入和输出操作。在 NEWDOS/80 2.0 中，文件的存取方式 "R" 除了可以象 TRSDOS 中那样进行随机存取外，也可以进行顺序存取，但不使用 PRINT/INPUT 语句而使用 GET/PUT 语句。在本例中，由行 40 的 GET 语句把记录顺序地从文件读入 BASIC 变量，再通过读者在语句行 10000 提供的子程序修改（或不修改）这些变量。由该子程序返回，行 60 的 PUT 语句把记录写回文件，于是完成了修改文件记录的工作。

在 PUT 语句中的文件定位参数是字符#。对于本例，它在每次执行 PUT 语句之初，将文件重新定位于刚才由 GET 语句读出的那个记录的开头，使此 PUT 操作修改同一个记录。

GET 和 PUT 语句都使用相同的位于文本行 120 的 IGEL。这个 IGEL 与例 1 和例 2 中使用的相同，只是它不包含在 GET 或 PUT 语句之中，而在单独的一个文本行中，因此在 GET 和 PUT 语句中应含有此行号作为它们的第三参数（即语句格式中的 IGELSN)。

如果 PUT 语句发现记录的新长度超过例 1 中原先分配给该记录的长度，则宣布错误。这种情况仅仅发生在以下这种时候：记录中的字符串项所使用的空间的总和超过了字符串原先占用的空间与原先记录中含有的任何空的空间的总和（使用 (len)# 功能可以在记录中插入空字符串，见 8.4 节之 3)。对于数字值可以放心地进行修改，因为它们总是占用相同数量的文件空间。因此，如果在 MU文件中修改一个字符串项，则字符串的最终长度不能增加。

例4. 读入一个 MU 文件，进行内部排序，然后再写回 MU 文件

```
10  CLEAR  10000 : DEFINT  I
15  DIM  NM$(200), AN%(200), AM!(200), DT#(200)
17  DIM ST$(200), IG%(200), FP! (200), DP #(200), IX%(200)
20  IX=0 : OPEN "I", 1, "XXX/DAT:1", "MU"
30  IF  EOF(1)  THEN  80
40  IX=IX+1 : IF  IX>200  THEN PRINT  "TOO MANY  RECORDS" : END
50  GET  1,,60 : GOTO 30
60  NM$(IX), AN%(IX), AM!(IX), DT#(IX),          'IGEL  lst  line
70  ST$(IX), IG%(IX), FP!(IX), DP#(IX);          'IGEL  last  line
80  IF  IX=0  THEN  PRINT  "EMPTY  FILE" : END
90  CMD"O", IX, *IX%(1), AM!(1), NM$(1)
100  CLOSE : OPEN"O", 1, "XXX/DAT:1", "MU"
110  IY=IX : FOR IZ=1  TO  IY
120  IX=IX%(IZ) : PUT  1,,  60
130  NEXT  IZ : CLOSE : END
```

把 MU 文件 XXX/DAT 的 1 到 200 个记录读入八个数组。

然后用 IX% 数组作为整型间接数组，利用 BASIC 的数组排序功能在行 90 间接地把记录排序。它用 AM! 值对 IX% 进行排序，排序的次序是 AM! 的上升序列（如果行 90 的排序数组前冠以负号，则为降序）。在排序过程中建立的整数数组 IX%，按要求的排序次序排列，它顺序地含有索引其他数组的索引值。

应注意，排序不改变记录数组 AM! 和 NM$ 中的任何内容。在排序过程中，首先预置 IX% 数组，使每个连续的元素指向 AM! 数组的连续元素，然后前后移动 IX% 数组中的元素，使符合排序的次序（AM! 的升序），于是数组 AM! 和 NM$ 就间接地排序好了。

还应进一步注意，虽然文件记录共有八个数组，但排序只考虑其中的两个，即行90中为 CMD"O" 语句提供排序数据的 AM! 和 NM$。

在排序完成以后，文件的记录以排序好的次序写回文件。这是在行110—130的循环中进行的，其中以数组 IX% 作为索引，把 AM! 和 NM$ 按 IX% 元素规定的次序写回文件。由于使用相同的文件存放排序好的记录，因此，用户务必保持原始文件的备份副本，以免在输出排序好的记录的过程中发生错误。

这个例子还说明了 IGEL 可以含有数组变量名，并且 IGEL 可以跨越多个文本行（本例为行 60 和 70 两行），当 IGEL 有多个文本行时，其最后一行必须用分号结尾，其他各行用逗号结尾。

例5. 把记录顺序地写入 FF 文件

FF 文件是分成等长记录的固定项文件，即文件的记录长度是一定的（由 OPEN 语句的 LRECL——逻辑记录长度规定），记录中的文件项也是一定的（包括类型、长度），所有的文件项都由 IGEL 定义。可以把 FF 文件看作字段项文件（TRSDOS 随机文件）的变种。虽然有

些用户想使用 8.11 节中讲述的伪 FIELD 功能，但在 FF 文件中不能使用 FIELD语句。即使用户已经通过伪 FIELD 功能把字符串设置成指定的长度，在 FF 文件处理中也不使用 LSET 和 RSET 来设置组成记录的变量（用户可能想用 LSET 或 RSET 语句来保持字符串变量的长度)。对于数字变量，决不能 使用LSET 和 RSET 语句。在 FF 文件中，也不使用 MKD$，MKI$，MKS$，CVD，CVI 和 CVS 等功能。

IGEL 中的每个字符串变量必须用文件中字符串项将具有的长度作前缀，并且不管 PUT 时字符串变量中的字符个数，文件中对应的字符串项将在右边截尾或填充空格，使它达到所需的字符个数。但在 RUT 过程中，字符串变量不改变，只有文件项被改变。

```
10  CLEAR  2000
20  OPEN  "O", 1, "YYY/DAT : 1", "FF", 63
30  GOSUB  10000              'build  data  for  record
40  IF  RN%=0  THEN  CLOSE : END          'all  done
50  PUT  1,,, (20) NM$, AN%, AM!, DT#, (15) ST$, IG%, FP!,  DP#;
60  GOTO  30
```

在行 20 的 OPEN 语句中，我们以顺序输出方式 "O" 打开文件，规定文件类型为 "FF"，逻辑记录长度 (LRECL) 为 63,以便把记录长度为 63 字节的每个记录顺序地写入文件名为 "YYY/DAT" 的 FF 文件。

在行 50 的 PUT 语句中，文件定位参数空缺，它表示每次执行 PUT 都把下一个 顺序记录写入文件。

读者要在行 10000 提供为记录产生数据的子程序。如果再没有记录要生成了，令 RN%=0, 否则令 RN%≠0, 并且把要传送给文件的数据装入 8 个变量：NM$， AN%，AM!，DT#，ST$，IG%，FP! 和 DP#。

IGEL 直接包含在 PUT 语句中，并且由八个命名的变量组成。这些与每个记录项有关联的变量名之间要用逗号分隔开。IGEL 要用分号结束。

字符串 NM$ 和 ST$ 的每一个，在文件中严格地按照 IGEL 中变量的前缀所指定个数的字符来描述。本例中，每执行一次 PUT 语句,NM$ 的当前内容就被送入文件。如果 NM$ 字符串多于 20 个字符，那么在右边多出的字符就被文件项丢弃，而 NM$ 不变。如果 NM$ 字符串少于 20 个字符，那么就在文件项的右边，而不是在 NM$ 的右边填充空格，使文件项长达 20 个字符。在限制为 15 个字符的与 ST$ 字符串有关联的文件项中也采用上述方法，使文件项保持 15 个字符。

整数 AN% 和 IG% 的每一个，在文件中用 2 个字节描述，其格式与 BASIC 使用的整数格式相同。

单精度浮点数 AM! 和 FP! 的每一个，在文件中用 4 个字节描述，其格式与 BASIC 使用的单精度数格式相同。

双精度浮点数 DT# 和 DP# 的每一个，在文件中用 8 个字节描述，其格式与 BASIC 使用的双精度数格式相同。

例6. 从 FF 文件顺序地读记录

```
10  CLEAR  2000
```

```
20  OPEN "I", 1, "YYY/DAT:1", "FF", 63
30  IF EOF(1) THEN CLOSE : END
40  GET 1,,, (20) NM$, AN%, AM! DT#, (15) ST$, IG%, FP!, DP#;
50  GOSUB 10000              'process the record's data
60  GOTO 30
```

这个例子是例5的逆处理过程，它与例5使用相同的IGEL和变量名，并使用例5建立的FF文件，从其中读取数据记录。与例5不同的是这儿用"I"方式打开文件，因此文件的数据记录在行40被顺序地读出并赋给相应的变量。读者应在行10000提供处理数据的子程序，对读出的数据进行必要的处理。

在GET语句中，文件定位参数是空参数，它意味着每执行一次GET就读出文件中的下一个顺序记录。

在每个记录读出以后，NM$含有一个20个字符的字符串，而ST$含有一个15个字符的字符串。

例7. 顺序地读出和修改FF文件的记录

```
10  CLEAR 2000
20  OPEN "R", 1, "YYY/DAT : 1", "FF", 63
30  IF EOF(1) THEN CLOSE : END
40  GET 1,, 120              'read the next sequential record
50  GOSUB 10000              'update the record's data
60  PUT 1,#,120              'rewrite the record back to the file
70  GOTO 30
120 (20)NM$, AN%, AM!, DT#, (15) ST$, IG%, FP!, DP#;
```

这儿使用例5中建立的同一个文件，并以"R"方式打开此FF文件，以进行顺序的输入和输出操作。在行40，记录被顺序地从文件读入BASIC变量，然后，读者可在行10 000提供的子程序中修改（或不修改）这些变量。由子程序返回，记录被写回文件（行60的PUT语句）。读者在本例与例3比较时，将不难发现两者是基本相似的，差别仅在于，本例为FF文件，因此必须在OPEN语句中规定LRECL，在IGEL中要用len前缀规定字符串项的长度。

PUT语句的文件定位参数是字符#。本例中，它使得在每次执行PUT之初，把文件往回重新定位于由GET读取的文件记录的开头（更确切的说法是定位于REMRA），因此，可使PUT操作重写该记录。

GET和PUT语句都使用开始于行120的IGEL。

在每次执行GET以后，NM$含有一个20个字符的字符串，而ST$含有一个15个字符的字符串。在用户提供的处理（即子程序10 000）中，字符串之一或两者的长度都可以改变。在PUT过程中，对应NM$的文件项又被置成20个字符长——按需要在右边填充空格或截尾。这相同的概念也用于对应ST$的15个字符的文件项。记住，NM$和ST$本身不被PUT操作改变，改变的只是它们对应的文件项。

例8. 随机地读取和随意地修改 FF 文件记录

```
10  CLEAR  2000
20  OPEN  "D",  1,  "YYY/DAT : 1",  "FF",  63
30  GOSUB  10000          'determine  which  record  to  read
40  IF  RN%=0  THEN  CLOSE : END      'end  if  no  more
50  GET  1,  RN%,  120          'read  that  record
60  GOSUB  15000          'optionally  update  the  record's  data
70  IF  RN%<>0  THEN  PUT  1,RN%,120     'if  required,  write  the  record
80  GOTO  30
120  (20) NM$, AN%, AM!, DT#, (15) ST$, IG%, FP!, DP#;
```

这个例子与例 7 相似，只是本例进行的记录读取是随机的，并且用户可以对是否修改记录进行选择。

这儿使用例 5 建立的文件。对于每个记录进行如下处理：

1) 用户在行 10 000 提供的子程序确定下一次要查看哪一个记录。返回时，RN% 含有所需的记录号，如果 RN%=0，则结束运行。

2) 使用 RN%（记录号）作为文件定位参数来指定想要存取的记录。这种用记录号随机存取文件记录的概念和方法与 TRSDOS 的随机文件是一样的。

3) 用户在行15000提供的子程序用于查看记录的数据，并随意地改变一个或几个与该记录有关的变量。在从这个子程序退出时，如果不修改记录，则把 RN% 置为 0，否则不改变 RN%。

4) 如果 RN% 不为 0，则行 70 的 PUT 语句使用 RN% 作为文件定位参数把该记录写入文件（修改了文件记录）。注意，该 PUT 语句的文件定位参数也可以使用字符#，其效果相同。

注意，在本例中 OPEN 语句使用 "D" 而不是 "R" 作为它的第一个参数（即存取方式）。也可以使用 "R"，虽然两者都是随机存取方式，但是在 PUT 操作中，当记录号超过文件的 EOF 时，"R" 方式允许扩展文件，而 "D" 方式不允许扩展文件，这就是两者的差别。因此，如果由于某种原因在 PUT 操作之前，RN% 被改变成超出文件范围的记录号时，"D" 方式能防止文件被扩展。

例9. 顺序地把记录写入 MU 文件并顺序地把数据写入作为 MU 文件的索引的 FF 文件

很多情况下，用户会有一个很大的文件，它的每一个记录含有各种长度的字符串，并且不希望对字符串进行字段项文件或固定项文件中所进行的填充或截尾，还希望仍然能够随机地存取文件，以及在有限的范围内能够修改该文件。使用 MU 文件作为主文件，并使用 FF 文件作为检索主文件的索引文件，用户就能达到这个目的。

```
10  CLEAR  2000
20  DIM  AN%(4000), RB!(4000)          'two  arrays  to  hold  index  data
30  OPEN  "O",  1,  "XXX/DAT:1",  "MU"          'open  the  main  data  file
40  RC%=0
```

```
50 GOSUB 1000          'create next record's data
60 IF RN%=0 THEN 105          'done with main file
70 RC%=RC%+1 : IF RC% 4000>THEN PRINT "FILE TOO LARGE":
GOTO 105
80 RB!(RC%)=LOC(1)!          'save RBA of next record
90 PUT 1,,, NM$, AN% (RC%), AM!, DT#, ST$, IG%, FP!, DP#;
100 GOTO 50
105 CLOSE
110 IF RC%=0 THEN PRINT "NO DATA RECORDS" : END
120 CMD"O", RC%, AN% (1), RB! (1)          'sort index data
130 OPEN "O", 1, "YYY/DAT:1", "FF", 6          'open index file
140 FOR X=1 TO RC%
150 PUT 1,,, AN% (X), RB!(X);          'write index record
160 NEXT X : CLOSE : END
```

这个例子也可编程为交错地把主文件与索引文件的一个记录分别写入各自对应的文件中去，每个文件一次写入一个记录。但是，由于这两个文件都在同一个驱动器内的盘片上，所以驱动器臂将不停地往返运动，以便交错地给两个文件写入各自的每一个记录。因此，在本例中，为避免上述情况，在写入主数据文件记录的时候，先把索引文件的数据存入数组，等主文件全部写完以后（共 4000 个记录），一起写入索引文件。

在这个例子中，数组 AN% 用于保存帐目号码（假设此记录是有关财务帐目的记录），并且对每个主数据文件记录，此帐号是唯一的。

本程序的主要处理过程如下：

1） 对于主数据文件

用户应在行 10 000 提供子程序，以产生文件的记录的数据。如果再没有主数据文件记录要建立，则令 RN%=0，否则，令 RN%≠0，并通过把数值存入相应的变量中，包括把帐号存入它的数组中，来建立记录的数据。

主数据文件中放置该记录处的 RBA，是由行 80 的 LOC(1)!函数确定的，并把它存入RBA 数组 RB!。

然后在行 90 把该记录写入主数据文件，并重复上述过程，直至写完整个主文件。

2） 对于索引文件

把 AN% 和 RB! 两个数组进行直接排序，数组实际上都要按照排序的次序进行排列，这个排序的次序是帐号的上升序列。注意，虽然 RBA 的上升序列是第二排序准则，但是，因为帐号是唯一的，所以永远不检查 RBA 值。

索引文件也是按照帐号的上升序列，把数组作为索引记录写入索引文件来建立的。

若把上述文本行 80 和 90 写成如下形式，则获得完全相同的结果。

```
80 PUT 1,,, NM$, AN% (RC%), AM!, DT#, ST$, IG%, FP!, DP#;
90 RB!(RC%)=LOC(1)#          'RBA where the record was placed
```

例10. 随机地读取和随意地修改一个被索引的 MU 文件的记录

```
10  CLEAR  2000
20  DIM  AN%(4000),  RB!(4000)
30  RC%=0 : OPEN "I", 1, "YYY/DAT:1", "FF", 6     'open index file
50  IF  EOF (1) THEN 100
60  RC%=RC%+1
70  IF RC%>4000 THEN PRINT "INDEX TOO LARGE" : CLOSE : END
80  GET 1,,, AN% (RC%), RB! (RC%);          'read index data into arrays
90  GOTO 50
100  CLOSE : IF  RC%=0  THEN  PRINT "NO ACCOUNTS" : END
110  OPEN "D", 1, "XXX/DAT : 1", "MU"          'open main data file
120  GOSUB  10000          'determine  which  account  record  wanted
130  IF  RN%=0  THEN  END          'if  required,  end  the run
140  FOR  X=1  TO  RC%
150  IF  RN%=AN%(X)  THEN  170          'search index for account #r match
160  NEXT  X : PRINT  "BAD  ACCOUNT  NUMBER" : GOTO 120
170  GET  1,!RB!(X),  300          'read  the  account  record
180  IF  AN%<>RN%  THEN  PRINT  "BAD DATA FILE" : CLOSE : END
185  GOSUB  15000          'display  data,  receive  back  updates
190  IF  RN%<>0  THEN  PUT  1,#,300          'if  required,  rewrite record
200  GOTO  120
300  NM$, AN%, AM!, DT#, ST$, IG%, FP!, DP#;
```

这个例程序假设有这样一个应用，它基本上是为了给用户显示读取的数据，以及在某些情况下，从用户那里接收修改信息，以便改变已经存放在主数据文件中的信息。

在这个例子中要使用例 9 中建立的两个文件。首先打开索引文件，把它的内容读入两个数组 AN% 和 RB!（程序行 30—90），以便在用户指定帐号时用它们去检索主数据文件。

对每一个记录的处理如下：

用户在行 10 000 提供的子程序请求用户确定要索引的主数据文件记录 的帐 号。在由该子程序返回时，RN% 含有此帐号，如果是 0，则表示用户要求结束运行。

在行 140—160，搜索帐号数组 AN%，寻找与用户指定帐号相符合的帐号。如果没有找到，就显示错误信息。

找到符合的帐号以后，就利用由 RB! 数组得到的帐目记录的 RBA 值作文件 定位参数，从主数据文件读取帐目记录（行 170）。然后把用户指定的帐号 RN% 与由文件得到的 AN% 值（即文件记录中的帐号）进行比较，如果不相等，则显示错误信息并结束运行（行180）。

用户应在行 15000 提供处理主文件记录数据的子程序，即我们开始时假设的应用：显示帐目数据，如果需要，则修改数据记录中需要修改的某些字段等等。如果没有修改记录，用户应在返回主程序前令 RN=0。

在返回主程序后，如果 RN%≠0，则表示这个数据记录已被修改了，于是把该记录重新写入文件（行 190）。PUT 语句使用了文件定位参数#，与前述一些例子一样，它把文件重新定位于该记录的开头（换言之，#使文件又定位于 REMRA）。

例11. 把不同类型的记录写入MU文件

在日常的应用中，用户可能要使用不同类型的记录。通常，各种记录类型应存入不同的文件，但是，如果一种类型的记录对应于一种或多种其他记录类型的一个或多个记录时，一般来说，用户总是想把所有这些有关联的记录一起存入一个文件内，以便使存取它们所需要的时间减到最小。MU、MI和FI 文件完全允许这种记录类型的混合，而字段项、MF和FF文件只有在记录都是相同长度时才能这样做。

保险公司的保险帐目或许是这种记录类型混合的一个典型例子。例如，它可以有一个帐目主记录、一个或多个分别涉及帐目的记录、一个或多个付款金额的记录以及一个或多个赔偿要求的记录，而这四个记录类型的每一个都具有不同的格式。

在下面的例程序中，使用三个不同的记录类型：主记录和两个辅助类型的记录。这三个记录只是一些假想，它们与现实中的任何事物之间都没有关系。我们在这个例子中要做的一切，是要说明MU文件怎样能含有多种类型的记录。

下面的例子中，每一组记录由一个类型1的记录和数个类型2、类型3记录（即每一组记录中可以有多个辅助类型记录，或者一个辅助记录也没有）的混合体组成。

```
10  CLEAR  2000
20  OPEN  "O", 1, "XXX/DAT:1", "MU"
30  GOSUB  10000          'create  type 1 record's data
40  IF  RN%=0  THEN  CLOSE : END          'all  done
50  PUT  1,,,"1", NM$, AN%, AM!, DT#, ST$, IG%, FP!, DP#;      'write
type  1  record
60  GOSUB  11000          'create  type  2  or  3 record's data
65  IF  X$<>  "2"  THEN  90
70  PUT  1,,, "2", SA$, SB$, LN$, PD!;     'if type 2 record, write it
80  GOTO  60
90  IF  X$<>  "3"  THEN  30
100  PUT  1,,,"3", SJ$, DF#, IP%, IA%, FG!;          'if type 3 record, write it
110  GOTO  60
```

程序行30—50用于建立类型1记录，读者应在行10000提供产生类型1记录数据的子程序，如果再没有记录要建立，读者应在子程序中令RN%=0，于是返回主程序后结束运行。否则令RN%≠0，于是返回主程序后把类型1记录的数据写入文件。对于每一组记录，有一个也只有一个类型1记录。

读者在行11000提供的子程序用于建立类型2或类型3记录的数据，这两个类型的记录无论谁先谁后都可以。从该子程序返回时，X$含有记录的类型标志（"2"或者"3"），如果既不是"2"也不是"3"，则表示该记录序列的结束。类型2和类型3记录，在文件中是混合在一起跟在有关联的类型1记录的后边。对于一个给定的类型1记录，可以有多个类型2和类型3记录，也可以一个都没有，因为它们不是必需的。

程序中每个PUT语句含有它们各自的IGEL。注意，在这些IGEL的每一个中，第一项是一个表达式（本例中实际为常数"1"，"2"等）而不是命名的变量。它也可以是含有记录

类型字符的变量名，但是，我们用表达式代替变量名是为了说明用于写标记项文件的IGEL，可以含有表达式和命名的变量。如BASIC中表达式的含义一样，表达式包括了常数和变量，因此，这就是为什么叫它IGEL（项目组表达式表）而不叫IGVL（项目组变量表）。

例12. 从一个含有多种记录类型的MU文件中顺序地读取和随意地修改记录

这儿使用了例11建立的文件。在本例中，我们将说明标记项文件的部分记录读的特性（也可用于固定项文件）。我们将读出类型1记录的前三项、第五项和第七项，跳过类型2记录并完全处理（包括随意修改）类型3记录。

```
10  CLEAR  2000
20  OPEN  "D",  1,  "XXX/DAT : 1",  "MU"
30  IF  EOF(1)  THEN  CLOSE : END
40  GET  1,,,  RT$;          'read  record  type  character
50  IF  RTS<>  "1"  THEN  PRINT  "BAD  RECORD  TYPE" : CLOSE:END
60  GET  1,*,,NM$, AN%,, DT#,,IG%;        'read selected type 1 record items
70  IF  EOF(1)  THEN  CLOSE : END
80  GET  1,,,  RT$;          'read  next  record  type  character
90  IF  RT$ ="2"  THEN  70          'bypass  type  2  records
100  IF  RT$ <>  "3"  THEN  50
110  GET  1,*,200          'read  in  rest  of  type  3  record
120  GOSUB  11000          'process  type  3  record's  data
130  IF  RN%<>0  THEN  PUT  1,$,200          'if  required,rewrite  type  3
record
140  GOTO  70
200  SJ$,  DF#,  IP%,  IA%,  FG!;
```

在行40和80的GET语句，只读出下个记录的一项(即第一项，文件记录的类型标志)，并且把文件靠左定位于该记录的第二项。REMRA和REMBA都指向该记录的第一项。

在行60和110的GET语句，从行40或80的GET操作离开的地方开始继续读取记录的文件项。行60和110的GET操作在输入数据以前不前进到下个记录。然而，在输入数据以前，把REMRA置成指向该记录第二项的位置。因此，如果执行行130的PUT语句，PUT语句的重新定位参数把文件重新定位于指向该记录第二项的REMBA。REMRA不受行60或行110的GET语句的改变。

行60的GET操作读取类型1记录的第二、第三、第五和第七项，并使文件定位于第八项处。第四项和第六项被跳过，这是因为在IGEL的这两个文件项位置上出现连续的两个逗号。其后接着执行的行80的GET操作将跳过类型1记录的剩余项（第八项和第九项）。

类型2记录中的剩余项是在执行行90以后通过下一次执行行80的GET语句跳过的。

行110的GET语句读取类型3记录的剩余部分，即第二项到第六项。

读者在行11000提供的子程序可以使用类型3记录的全部数据，以及由类型1记录中抽出的部分数据，对类型3记录进行读者需要的任何处理。如果类型3记录被修改了，则在返回主程序前置RN%为非零值，否则令它等于零。

由此子程序返回后，在行 130 进行类型 3 记录是否修改过的判别，如果 RN% <>0，则把类型 3 记录重新写回文件。注意，只有第二项到最后一项被写回到文件中。第一项不改变，这是由于行 130PUT 语句所进行的文件定位是对 REMBA 位置定位，此 REMBA 位置是行 110 的 GET 操作过程中，在它的定位完成以后，而任何项目输入以前所在的文件位置。

记住，在修改 MU 和 MF 文件的记录时，一旦要把一个文件项写回到该记录中，那么记录中该文件项后面的所有项目也必须写回（如果这些项目仍然是记录的一部分的话），因为每一个只修改 MU 文件记录部分文件项的 PUT 操作，都要用空字节去填充写入记录的最后一项后边的所有字节（若有的话）。

对于记录分段文件，EOF 函数把下个记录的位置与文件的 EOF 进行比较。对于上面的类型 3 记录的处理，行 130 的 PUT 操作离开处的位置与下个记录的位置是相同的。对于类型 1 和类型 2 记录，GET 语句使文件分别定位于第八项和第二项。在这些情况下，EOF函数计算下个记录的位置，并用这个值与文件的 EOF 进行比较。

例13. 把记录顺序地写入 MF 文件（固定记录长度的标记项文件）

在用户的应用中，可能需要对含有字符串的文件进行完全修改的能力。由于MU文件在字符串延长时不能保证记录修改的成功，所以用户必须用字段项文件、FF 文件或者最终使用 MF 文件。字段项文件和 FF 文件的优缺点我们已经讨论过了，所以，我们在这里将只涉及 FF 和 MF 文件的优缺点。

MF 文件的优点是不用空格去填充字符串项使项目填满到它所允许的最大长度。每个字符串项用它所需数目的字符写入，最多但不超过它所允许的最大长度。然后，在记录的末尾，如果存在没有用的记录空间，就用空字节填充，而在读 MF 文件时，程序是永远不查看这些空字节的。虽然多数时候，在由文件得到的字符串项与另一个字符串的比较中，填充字节不会给用户带来麻烦，但是有时候它产生一些实际的问题，例如需要额外的磁盘空间。

MF 文件与 FF 文件相比较，它的缺点是需要使用更多的磁盘空间，这是因为它包含文件项的标记字节。在本例中，虽然除了 MF 文件不填充字符串以外，两者含有相同的数据，但是 MF 文件的记录大小仍然要比例 5 中相应的 FF 文件大 13%。

```
10  CLEAR  2000
20  OPEN  "O",1,"XXX/DAT:1",  "MF",  71
30  GOSUB  10000          'create  data  for  record
40  IF  RN%=0  THEN  CLOSE : END          'no  more  records
50  PUT  1,,,(20) NM$, AN%, AM!, DT#, (15) ST$, IG%, FP!, DP#;       'write
record
60  GOTO  30
```

在行 20 以顺序输出方式打开文件，规定文件类型为"MF"，每个记录的长度为 71 个字节。根据行 50 的 IGEL，这 71 个字节分别供长度为 21,3,5,9,16,3,5 和 9 字节的文件项使用（记住，标记项文件的每个项目都以一个标记字节开始）。

读者在行 10 000 提供的子程序用于建立 MF 文件记录的数据，如果再没有记录要建立，则令 RN%=0，否则令 RN%≠0，并把新记录的数据装入 8 个变量：NM$，AN%，AM!，DT#，ST$，IG%，FP! 和 DP#。

文件项在磁盘上的表示与例 1 中对 MU 文件的描述相同，只是 MF 文件不使用SOR 项，并且字符串都限制为最大的字符数目，即对于 NM$ 为 20 而对于 ST$ 为 15。如果在写入文件项时，字符串变量的长度大于文件项允许的最大值，那么就不把右边超出的字符传送给文件项。

严格地说，在上面行 50 使用的 IGEL 中，字符串表达式用最大字符串长度值作前缀不是一个必要条件。在例 1 行 50 中的 IGEL 就是这样做的。但是，对于任何一个字符串项，不指定最大字符串长度值时，就不能保证对记录的全修改能力。

由子程序返回，根据 PUT 语句中的 IGEL，把记录顺序写入 MF 文件。

例14. 随机地读取和随意地修改 MF 文件的记录

```
10  CLEAR  2000
20  OPEN  "D",1,"XXX/DAT:1",  "MF",  71
30  GOSUB  10000          'determine  which  record  to  read
40  IF  RN%=0  THEN  CLOSE : END          'end  if  no  more
45  IF  LOF(1)<RN%  THEN  PRINT  "BAD  RECORD  #" : GOTO  30
'check  if  the  record  is  within  the  file
50  GET  1,  RN%,  120      .    'read  that  record
60  GOSUB  15000          'optionally  update  the  record's  data  variables
70  IF  RN%<>0  THEN  PUT  1, #,120          'if required, rewrite the record
80  GOTO  30
120  (20)NM$, AN%, AM!, DT#, (15) ST$, IG%, FP!, DP#;
```

打开例 13 建立的 MF 文件作随机读写。OPEN 语句的第一个参数是 "D"，这种存取方式可以防止文件的扩展。

读者在行 10 000 提供的子程序中确定下次处理哪一个记录。从该子程序退出时，RN% 应含有读者所需记录的记录号，只是在 RN%=0 时，才结束运行。

然后读取记录。字符串 NM$ 的最终长度是 0 到 20 个字符，而ST$ 为 0 到15个字符，这取决于哪一个相应的文件项实际送入其中。记住，在写入项目时不用空格进行填充，并在 GET 时也不进行。

读者在行 15000 提供的子程序中可以随意地修改变量 NM$,AN%, AM!, DT#, ST$, IG%, FP! 和 DP# 中的数据。如果不修改此记录，令RN%=0；否则保持 RN% 不变，这表示子程序改变了记录中的某些变量或全部变量内容。

从这个子程序返回，如果RN%≠0，就把该记录写回文件。如果一个或多个字符串变量有了新的长度，则相应的文件项就采用此新长度。

请注意行 45 的 LOF 函数的应用，这个函数可得到通过指定 I/O 链（即指定文件区）进行存取的文件的最大记录号。在本例中，它用于检查所需要的记录是否在文件之中。

例15. 顺序写入 MI 文件

MI 文件和 FI 文件的内容是一长串项目。如果用户自己把项目逻辑上分组成记录，则 BASIC 对此毫不了解，这是因为在 OPEN 语句中没有象字段项文件、MF 和 FF 文件那样指

定记录长度；文件中也没有记录标记，例如 MU 文件的 SOR（记录开始）字节和输出/输入文件的 EOL（行结束）字节。由于对用户的 MI 和 FI 文件的可能的逻辑记录分段一无所知，BASIC 就不能象例 12 中行 80 的 GET 语句那样，跳过类型 1 或类型 2 记录的剩余部分，自动地前进到下一个记录。

我们将使用例 11 的程序，但要作一项修改，以产生一个由三个记录类型组成的 MI文件（记住，BASIC 对 MI 和 FI 文件中的记录一无所知）。只要把例 11 的程序行 20 改成：

```
20 OPEN "O", 1, "XXX/DAT : 1", "MU"
```

其他的程序行都不需改变。于是我们就能生成一个下面例 16 中使用的 MI 文件。

例16. 顺序地读 MI 文件

这儿使用例 15 中建立的MI 文件。虽然用户知道此文件含有三种类型的记录，但BASIC 不知道。因此，要前进到下一个记录，程序必须完整地读出前一个记录，虽然它不必在一个 GET 语句中进行此全部工作。

MI 文件不能修改。这个限制是因为不可能处理长度变化的字符串。

```
10 CLEAR 2000
20 OPEN "I",1,"XXX/DAT:1", "MI"
30 IF EOF(1) THEN CLOSE : END          'exit if empty file
40 GET 1,,, RT$;          'read record type character
50 IF RT$ <> "1" THEN PRINT "BAD RECORD TYPE" : CLOSE :END
60 GET 1,,, NM$, AN%, AM!, DT#, ST$, IG%, FP!, DP#;      'read type
1 record
70 IF EOF(1) THEN CLOSE : END
80 GET 1,,, RTS;          'read next record type character
90 IF RT$ ="2" THEN 150、
100 IF RT$ <>"3" THEN 50
110 GET 1,*,,SJ$, DF#, IP%, IA%, FG!;          'read type 3 record
120 GOSUB 11000          'process using type 1 and 3 record data
140 GOTO 70
150 GET 1,,,SA$,SB$,LN$,FD!; : GOTO 70
```

记住，虽然上述程序中的注释论及记录，但是文件逻辑分段成记录只有用户自己知道，而 BASIC 不知道。

注意，上面行 60 中使用了一个空定位参数，而在例 12 中，行 60 的该处使用的定位参数是＊号。在例12中使用＊号是因为文件定位要停留在它所处的位置而不前进到下个记录。然而，由于对 MI 和 FI 文件，BASIC 对用户进行的记录分段一无所知，所以空定位参数和＊号的定位参数的作用完全相同，它使文件定位于它目前所处位置。因此，在上面的行40、60、80、110 和 150 中，空缺的定位参数和＊可互换地使用。可是还要记住，对于 MU、MF 和 FF 文件处理，空缺的意义和＊号的意义之间是有差别的（见第九章）。

例17. 把记录顺序地写入 FI 文件，并在该文件的末尾顺序地写索引记录，以便检索主记录

FI 文件是一个非常灵活的文件。当允许使用不同长度的记录时，它给用户提供了 FF 文件的能力。记住，就象 MI 文件似的，BASIC 对用户自己进行的 FI 文件的记录分段一无所知。可是对于用户来说，FI 文件可以是一个记录种类齐全的文件。本例和例 18 将使用由五种不同逻辑记录类型组成的 FI 文件：例 11,12,15 和 16 中使用的三种记录类型（一个主记录和两个辅助记录）、例 9 和例 10 中 FF 文件使用的索引记录、以及本例文件中独有的另一个 2 字节的记录类型（索引记录的数目）。

在这个例子中，我们假设记录类型 1 的文件项 AN% 是一个帐号，而且该帐号对于每个类型 1 记录是唯一的。本例的处理步骤是：首先写文件使它含有上述的所有前三种类型的记录。每个类型 1 记录的 RBA 和帐号被存入两入 BASIC 数组。在写完全部数据记录以后，再写一个表示数据记录结束的一个字节的记录。接着，把这两个数组按帐号的上升序列进行排序。然后把索引记录写入文件。最后，把索引记录的数目写入文件，并关闭文件结束运行。

这个例子是例 9 和例 11 的混合，并且那里的大多数注释都用于此处，不同的是这儿处理的是一个 FI 文件，而不是 MU 文件和 FF 文件。

```
10  CLEAR  2000
20  DIM  AN%  (4000), RB! (4000)        'arrays  for  index  data
30  OPEN  "O",1, "XXX/DAT:1", "FI"  'open  the  combined  data/index  file
40  RC%=0
50  RT$="1" : GOSUB  10000          'create  next  type  1  record
60  IF  RN%=0  THEN  170          'done  with  main  data
70  RC%=RC%+1 : IF  RC%>4000  THEN  PRINT  "FILE  TOO  LARGE" :
GOTO  170
80  RB! (RC%)=LOC(1)!        'RBA  where  type  1  record  will  be  stored
90  PUT  1,,,(1)RT$, (20)NM$, AN% (RC%), AM!, DT#, (15)ST$, IG%, FP!,
DP#;      'write  type  1  record
100  GOSUB  15000       'create  type  2  or  3  record
110  IF  RT$="2"  THEN  150
120  IF  RT$<>"3"  THEN  50
130  PUT1,,,(1)RT$, (40) SJ$, DF#, IP%, IA%, FG!;        'write type 3 record
140  GOTO  100
150  PUT  1,,,(1)RT$, (3)SA$, (32)SB$, (14)LN$, PD!;    'write type 2 record
160  GOTO  100
170  IF  RC%=0  THEN  PRINT  "NO  DATA  RECORDS" : END
175  RT$="0" : PUT  1,,,(1)RT$;          'flag  end  of  main  data
180  CMD  "O", RC%, AN% (1), RB! (1) .       'sort  index  data
190  FOR  X=1  TO  RC%
200  PUT  1,,,AN%(X),  RB!(X);          'write  index  record
210  NEXT  X
220  PUT  1,,,RC%;          'write  number  of  index  records
230  CLOSE : END
```

索引记录与数据记录写在一起的优点是两者只使用一个文件，于是避免了保持数据和索引文件在复制备份副本时互相协调一致的问题。在例 9 和例10中，我们也可以只使用一个文件，即把索引存放在 MU 文件的末尾。

由于文件是固定项类型的，所以 IGEL 中命名的所有字符串变量都必须用文件项具有的长度作前缀。于是，在字符串数据被送入文件项时，将在其右边进行截尾或填充空格。

就象例 9 那样，若把上述文本行 80 和 90 写成如下形式，则可获得完全相同的结果：

```
80  PUT  1,,,(20)NM$, AN% (RC%), AM!, DT#, (15) ST$, IG%, FP!, DP#;
90  RB!(RC%)=LOC (1)#          'RBA  where the record was placed
```

注意，在行 175 写入了一个 1 字节的标志主数据结束的数据记录。这个分隔符是例18需要的。

例18. 随机地读取和随意地修改被索引 FI 文件的数据记录

这儿使用例 17 中建立的 FI 文件。读取此文件的最后两个字节，以确定索引记录（和类型 1 记录）的数目。然后把索引记录读入两个数组。接着，从文件中读取所选择的数据记录组，并且随意地修改类型 3 记录的 DF# 和 IA% 项，再送回文件。

这个例子是例 10 和例 12 的混合，只是索引和数据都包含在同一个FI文件中，并且只修改类型 3 记录的两项，而其他的差别如下所述。

```
10  CLEAR  2000
20  DIM  AN% (4000), RB! (4000)          'two arrays for index data
30  OPEN "D", 1, "XXX/DAT:1", "FI"
40  X!=LOC(1)%-2          'compute  RBA  of  file's  last  2  bytes
50  GET  1,!X!,,RC%;          'read  in  count  of  index  records
60  GET  1,!$(X!-6*RC%)          'position  file  to  lst  index  record
70  FOR  X=1  TO  RC%          'read  index  data  into  the  two  arrays
80  GET  1,,,AN%(X),  RB!(X);
90  NEXT  X
100  GOSUB  10000          'determine  witch  account  to  process
110  IF  RN%=0  THEN  CLOSE : END          'run  is  completed
120  FOR  X=1  TO  RC%          'search  account  #  array  for  match
130  IF  RN%=AN%(X)  THEN  150
140  NEXT  X : PRINT "BAD  ACCOUNT  NUMBER" : GOTO  100
150  GET  1,  !RB!(X),,(1)RT$,(20)NM$,AN%,AM!,DT#,(15)$,IG%,(12)$;
'read  type  1  record
160  IF  RT$<>"1"  THEN  PRINT "BAD  INDEX" : CLOSE  :  END
170  IF  RN%<>AN%  THEN  PRINT "BAD  FILE  DATA" : CLOSE:
END
180  GET  1,,,(1)RT$;          'read  next  record's  type  character
190  IF  RT$="3"  THEN  230
195  IF  RT$="1"  OR  RT$="0"  THEN  100          'test  end  of  account
```

```
200 IF RT$ <>"2" THEN PRINT "BAD FILE DATA" :CLOSE:END
210 GET 1,*,, (3)SA$, (32)SB$, (14)LN$, PD!;   'bypass type 2 record
220 GOTO 180
230 GET 1,,, (40)SJ$,DF#,(2)$,IA%,FG!;      'read type 3 record
240 GOSUB 11000          'process type 1 and 3 record data
250 IF RN%<>0 THEN PUT 1,$,,(40)$,DF#,(2)$,IA%,(4)$;
'update type 3 record
260 GOTO 180
```

请注意行 40 中 LOC(1)%函数的应用，LOC 函数得到文件 EOF 的 RBA。在此用它计算文件倒数第二个字节的 RBA，由于文件的最后两个字节含有文件中索引记录的数目和文件中类型 1 记录的数目这两者的整型计数值。这个整数值在行50被读入 RC%。

在行 60，使用这个索引记录的计数，计算第一个索引记录的 RBA，并把文件定位于第一个索引记录的开头。注意，此计算是在 GET 语句的文件定位参数中正确地完成的。只要此计算本身不引用文件区域就可以这样做。也请注意，文件定位参数是属于!$ rba 型的（见 8.8节），这表示此 GET 操作仅用于文件定位，而不进行数据传送，因此 IGEL 或 IGELSN 是不允许出现在这种 GET 或 PUT 语句中的。

用户在行 10000 提供的子程序应确定需要处理的记录组的帐号。如果再没有帐目要读入，令 RN%＝0，否则令 RN% 等于帐号。

在行 150，文件使用与索引数组中的帐号有关联的 RBA 进行定位，然后读取类型 1 记录。因为这儿假设不必知道在对应 ST$、FP! 和 DP# （见例 17 的行 90）的文件项中是什么，因此，可以跳过这些项目。但是，由于这是固定项文件，所以必须使用(len)$ 格式（见 8.4 节)告诉 BASIC 要跳过的文件字节数。(15)$ 使得跳过通常应读入字符串 ST$ 的15 个文件字节，(12)$ 使得跳过通常应读入 FP! 的 4 个字节和应读入 DP# 的 8 个字节。最后，如果把这个 IGEL 用于 FF 文件处理中，那么(12)$ 表达式就应从 IGEL 中丢弃，因为在 IGEL 中它后面没有别的需要读入数据的表达式了，并且下一个带有空缺的定位参数的 GET 操作将使文件前进到下一个记录，跳过了这些应跳过的字节。而在 FI 文件的处理中，用户必须自己计算所有的记录空间，因为 BASIC 对他的记录结构一无所知，因此需要此(12)$。

在行 210，虽然我们想做的一切只是跳过类型 2 记录，但是，我们仍然必须自己来进行文件定位，因为 BASIC 不知道这个记录在什么地方结束。用以下语句也可获得相同的文件位置的推进量：

```
210 GET 1,,,(53)$     'bypass rest of type 2 record
```

用户在行 11000 提供的子程序用于处理从类型 1 和类型 3 记录得到的数据，本例程序中规定只能改变类型 3 记录中的 DF# 和 IA% 两项（见行 250)。如果不修改类型 3 记录，则在返回主程序前，令 RN%＝0，否则改变 DF# 和 IA%之一或两者，并令 RN%≠0。

在行 250 的 PUT 语句，把文件重新定位在行 230 的 GET 操作所记忆的 REMBA 位置。然后修改类型 3 记录。注意，它只替换文件中的两项，它们对应于 DF# 和 IA%（请参阅例 12,与写类型 3 记录的行 130 进行比较)。其他的项目都被跳过并且不改变。在 IGEL 中使用 (len)$ 或 (len)#表达式时，用户一定要计算真正的字节数。

在行 150 和 250 的 IGEL 中，使用 (len)$ 表达式的目的只是为了给读者一个(len)$ 应用

的例子。如果用户不想使用 (len)$，而宁愿象程序行 210 那样使用正规的 IGEL，那么可把这两行改成下列程序行，其效果是完全一样的：

```
150  GET  1,*,,(1)RT$,(20)NM$,AN%,AM!,DT#,(15)ST$,IG%,FP!,DP#;
250  IF  RN%<>0  THEN  PUT  1,$,,(40)SJ$,DF#,IP%,IA%,FG!;
```

附录A　错误代码和信息

A.1　DOS　错误代码和信息

表 A.1 是 NEWDOS/80　2.0 版本的 DOS 错误信息及其对应代码表。当把错误代码放入寄存器 A 中，然后执行 CALL　4409H 或 JP　4409H 时，就可获得相应的错误信息。如果错误代码没有定义，则显示“UNKNOW　ERROR　CODE”（未知的错误代码）信息。在系统程序中，SYS4/SYS 是 DOS 错误信息显示模块。表 A.1 按错误代码的大小次序，同时用十进制和十六进制的形式列出。

表A.1　DOS 错 误 代 码

错误代码 Dec.	错误代码 Hex.	错　误　信　息
00	00	没有错误
01	01	文件数据有问题
02	02	在读过程中寻道错误
03	03	在读过程中丢失数据
04	04	在读过程中奇偶错误
05	05	在读过程中数据记录没有找到
06	06	企图读已封锁的或已删除的记录
07	07	企图读系统记录
08	08	设备尚未就绪
09	09	没有定义的错误代码
10	0A	在写过程中寻道错误
11	0B	在写过程中丢失数据
12	0C	在写过程中奇偶错误
13	0D	在写过程中数据记录没有找到
14	0E	在磁盘驱动器上写错误
15	0F	写保护着的盘片
16	10	设备不能使用
17	11	目录读错误
18	12	目录写错误
19	13	非法的文件名
20	14	磁道号太大
21	15	DOS-CALL下的非法功能
22	16	没有定义的错误代码
23	17	没有定义的错误代码
24	18	文件不在目录中

续 表

错误代码 Dec.	Hex.	错误信息
25	19	拒绝文件存取
26	1A	目录空间已满
27	1B	磁盘空间已满
28	1C	遇到了文件的结束
29	1D	超过了文件的结束
30	1E	目录已满，不能再增添文件
31	1F	程序没有找到
32	20	非法的或遗漏了驱动器号
33	21	没有设备可以使用
34	22	装入文件格式错误
35	23	存储器错误
36	24	企图装入只读存储器
37	25	企图非法存取保护着的文件
38	26	文件没有打开
39	27	在系统盘片上非法地初始置数
40	28	非法的磁盘磁道数
41	29	非法的逻辑文件号
42	2A	非法的DOS功能
43	2B	链接过程中的非法功能
44	2C	目录数据有问题
45	2D	FCB数据有问题
46	2E	系统程序没有找到
47	2F	参数有问题
48	30	文件标识符有问题
49	31	错误的磁盘记录类型
50	32	引导读错误
51	33	DOS 致命错误
52	34	非法的关键字/分隔符/结束符
53	35	文件已经存在
54	36	命令太长
55	37	拒绝磁盘存取
56	38	非法的 MINI-DOS 功能
57	39	用户/程序/参数要求操作结束
58	3A	数据比较中发现不一致
59	3B	内存不够
60	3C	不兼容的驱动器或磁盘
61	3D	文件为 ASE=N 属性，不能扩展文件
62	3E	不能通过读来扩展文件

A.2 磁盘BASIC错误代码和信息

在磁盘 BASIC 中，除了用到标准的 ROM BASIC LEVEL II错误代码以外（请参阅 计

声编译,《微计算机实用手册》，海洋出版社，1982年7月，第80页)，还使用以下磁盘BASIC错误代码。

表A.2 磁盘BASIC错误代码

代码	错误信息
51	字段溢出
52	内部错误
53	文件号有问题
54	文件没有找到
55	文件方式有问题
56	文件已经打开
58	DOS错误
59	文件已经存在
62	磁盘满了
63	输入超过末尾
64	记录号有问题
65	文件名有问题
66	方式失配
67	文件中的直接语句
68	文件太多
69	磁盘写保护
70	拒绝文件存取
71	顺序号溢出
72	记录溢出
73	非法扩展文件
75	前边显示过的错误
76	不能处理行0
77	文件类型有问题
78	IGEL 句法错误
79	IGEL 项句法错误
80	错的/非法的/遗漏了IGEL 项前缀
82	记录长度有问题
83	语句使用两个文件名
84	文件定位参量有问题

在系统程序中，SYS13/SYS是显示磁盘BASIC和ROM BASIC错误信息的模块。通常，只有在DOS需要用它显示错误信息时才将此模块装入内存储中。如果一个错误代码没有相应的错误信息，则显示“UNPRINTABLE ERROR”（不可显示的错误)。

附录 B　不兼容性及其处理

B.1　前言

和以前的 NEWDOS 版本一样，NEWDOS/80　2.0 也不可避免地存在一些错误，并且还与其他的 DOS 包括 NEWDOS 的早期版本之间存在着不兼容性。为此，Apparat 公司不定期地发表修补(Zap)通报，以改正错误、消除不兼容性和提供进一步的扩充。

NEWDOS/80 是一个不同于 TRSDOS, VTOS, LDOS, DOSPLUS 和其他操作系统的 DOS, 因此许多用户程序不作某些修改在 NEWDOS/80 上是不能运行的。对于某个特定程序能否在 NEWDOS/80 上运行，最好的方法是用 NEWDOS/80 试一试，当然，在这种试验中，你不应使用有价值的文件数据，以免破坏这些有用的数据。

但是，为减少混淆、挫折和不浪费时间，希望用户做到：

1．　阅读和弄懂现有的 NEWDOS/80 资料。

2．　要保证 NEWDOS/80 使用的程序语言是正确的，编制的程序是正确的。

3．　必须保证在你的 NEWDOS/80 系统或用户程序中，已经实施了所有非标准的必须进行的修补。

4．　在各种条件下（若有可能的话）反复验证导致 NEWDOS/80 错误或不兼容性的原因，然后再与其他用户商讨或向 Apparat 公司报告。

B.2　目录的 FPDE 中 EOF 字段的不兼容性

从 TRS-80 Ⅲ型的 TRSDOS 1.3 开始，TRSDOS 就使用 RBA（相对字节地址）作为目录 FPDE 中 EOF 字段的格式，并作为 FCB 中 EOF 和 NEXT 字段的格式（请参阅 5.7 节和 5.9 节）。

TRSDOS 的 FPDE 的 EOF 字段改成 RBA 格式虽然是一个正确的措施，但是随之产生一个严重的问题，即使得 TRS-80 Ⅲ型的 TRSDOS 1.1 和 1.2 版磁盘在 TRSDOS 1.3 上不能直接读出，反之亦然。而且，允许与 TRS-80 Ⅲ型 TRSDOS 磁盘之间进行文件复制的 NEWDOS/80 COPY 命令（见 2.14 节）的功能就被限制于 TRS-80 Ⅲ型的 TRSDOS 1.3 磁盘。

在设计 NEWDOS 时，曾经想把目录 FPDE 的 EOF 字段也设置成 RBA 格式。但是，这样做的话就会使 NEWDOS 磁盘与所有的已存在的 TRSDOS 磁盘不相兼容，并严重地降低 NEWDOS 的通用性。因为实际上只有很少几个程序去读写 FPDE 的 EOF 字段，并且，把它改成 RBA 格式也只是为了消除 FCB 处理中产生的混乱。权衡利弊得失，最后 Apparat 公司决定在 NEWDOS 中保持目录 FPDE 的 EOF 字段不变，即仍然采用 TRSDOS 的老格式。

由 NEWDOS 和老的 TRSDOS 使用的 FPDE 的 EOF 格式至 RBA 格式的转换过程，以及相反的转换过程是十分简单的。其规则是：

1）　由 NEWDOS 和老的 TRSDOS 格式转换成 RBA 格式：如果 3 字节值的低位字节不等于零，则从 3 字节值中减去 256（或者从高位的 2 字节值中减去 1）。

2) 由 RBA 格式转换成 NEWDOS 和老的 TRSDOS 格式：如果 3 字节 RBA 值的低位字节不等于零，则把 3 字节 RBA 值加 256（或者高位的 2 字节值加 1）。

B.3 TRS-80 I 型上 NEWDOS/80 2.0 版与 1.0 版的不兼容性

1. 工作于 TRS-80 I 型的 NEWDOS/80 1.0 上的大多数程序将可以在 TRS-80 I 型 NEWDOS/80 2.0 上正常工作。

2. 在 NEWDOS/80 2.0 上，BREAK 键的允许/禁止，再也不能通过 4369H 的位 4 进行控制。虽然用户程序仍然可以设置这一位，但是 DOS 忽略它不管。见 2.8 节。

3. FCB 的变化（见5.9 节）

(1)在 2.0 版本中，已放弃使用 FCB 第 1 字节的位 2（表示磁道和扇区操作)。

(2)已对 FCB 第 2 字节的位 3 和第 3 字节的位 7 至位 5 作了新的规定。

(3)FCB 的第 17 至第 32 字节已被重新定义。

4. 目录的变化（见 5.6,5.7 和 5.8 节）

(1)GAT 扇区现在考虑的是 gr. 组 (lump) 而不是磁道。GAT 扇区中 00—BFH 范围内的每个字节现在对应于一个 gr. 组而不是一个磁道，并且，现在使用每个 gr. 组有多少 gr. 而不是每磁道有多少 gr.。FPDE 和 FXDE 中的每个范围元的第 1 字节现在是 gr. 组的编号而不是磁道号。磁盘第一扇区（引导扇区）的第 3 字节现在是一个 gr. 组编号而不是磁道号。相应的 GPL (Granul Per Lump——每 gr. 组的 gr. 数）值是在 PDRIVE 命令中指定的。所有的1.0 版本磁盘和引导扇区的第 3 字节在 2.0 版本中都是直接可使用的，反之亦然，但使用中请多加小心。

(2)FPDE 第 2 字节的位 7 、位 6 和位 5 已被重新定义。

(3)gr. 分配表 (GAT) 现在可以随意地使用 GAT 扇区的前 192 个字节。如果磁盘的 gr. 组数目大于 96(60H)，则 gr. 的分配可以延伸到 gr. 存在表（封锁表）中，并取消了此表。

5. 不能够再用 4315H 去设置 DEBUG 的允许/禁止状态。虽然用户程序可以继续设置这一单元，但是 DOS 对它忽略不管。

6. 不能够由按下 BREAK 键来进入 DEBUG，而只能使用 '123' 联键操作（即同时按此三键）才可能进入 DEBUG（见 4.1 节)。

7. PDRIVE 命令已作了很大变动。请仔细研究 2.37 节。在 NEWDOS/80 2.0 上必须使用以下的 PDRIVE 命令来读写已存在的 NEWDOS/80 1.0 版本的盘片。当要制备一个在 1.0 版本上可读的盘片时也必须使用这些 PDRIVE 特征说明。

(1)PDRIVE, dn1, dn2, TI=A, TC=35, SPT=10, TSR=3, GPL=2, DDSL=17, DDGA=2

是用于标准的 5 英寸、单密度、单面磁盘的说明。对于 40、77 或 80 磁道的驱动器，要相应设置 TC 参数。

(2)PDRIVE, dn1, dn2, TI=A, TD=C, TC=80, SPT=20, TSR=3, GPL=4, DDSL=17, DDGA=2

这个 PDRIVE 特征说明用于 5 英寸、单密度、双面的磁盘。对于 35、40 或 77 磁道的驱动器，则要相应设置 TC 参数。

(3)PDRIVE, dn1, dn2, TI=BH, TD=B, TC=77, SPT=15, TSR=3, GPL=3, DDSL=17,

DDGA=2

是用于使用 OMIKRON 接口的 8 英寸、单密度、单面磁盘的说明。对于这种磁盘，NEWDOS/80 2.0 可以管理到 SPT=17（即每个磁道多达 17 个扇区的磁盘）；由此，你可以改造你的现有的盘片，以得到额外的 12%的空间。

（4）PDRIVE, dn1, dn2, TI=BH, TD=D, TC=77, SPT=30, TSR=3, GPL=6, DDSL=17, DDGA=2

是用于使用 OMIKRON 接口的 8 英寸、双面双密度磁盘的说明。在这种情况下，NEWDOS/80 2.0 可以管理每磁道 34 扇区的磁盘（即 SPT=34）；由此，你可以改造你的现有的盘片，以得到额外的 12%的空间。

（5）PDRIVE, dn1, dn2, TI=CK, TD=E, TC=34, SPT=18, TSR=3, GPL=2, DDSL=17, DDGA=2

是用于使用 PERCOM 接口的 5 英寸、单面双密度磁盘的说明。对于 40，77 和 80 磁道的驱动器，要把 TC 分别设置成 39,76 和 79。如果使用 LNW 接口，则令 TI=EK；如果这样不能工作，请用 TI=CK 试一试。

（6）注意！在 NEWDOS/80 1.0 版本上使用的 5 英寸、双面、双密度磁盘在 2.0 版本上不能直接使用。这些盘片上的文件必须在使用 NEWDOS/80 1.0 时，把它们复制到 NEWDOS/80 2.0 可以使用的双面单密度磁盘或单面双密度磁盘上。然后，通过刚复制成的磁盘把这些盘片上的文件再复制到 2.0 版本的双面双密度盘片上。

8. 在 NEWDOS/80 2.0 版本中，对于有双密度修正的 PERCOM 和 LNW 接口，可以支持 5 英寸双密度盘片。

9. SYSTEM 命令已进行了很大的扩充。请仔细研究 2.46 节。

（1）废弃了选用参数 AH 和 AK，而增加了选用参数 AT，AU…AZ，BA，BB 直到BN,但其中不包括 BL。

（2）选用参数 BN 确定 NEWDOS/80 是写 TRS-80 Ⅰ型 TRSDOS 可读出的单密度目录扇区还是写 TRS-80 Ⅲ型 NEWDOS/80 可读出的单密度目录扇区。只允许其中之一，而不允许两者兼备。

（3）选用参数 BJ 允许增加 NEWDOS/80 磁盘延迟定时循环，于是在磁盘 I/O 期间使得加速的 CPU 模块能正常工作。NEWDOS/80 能管理加速的 CPU 模块，但是它不能管理低于标准的 1.772 兆赫速度的 CPU 模块。

10. COPY 命令已作了相当大的变动。请认真研究 2.14 节。

（1）即使系统盘片没有装配着，或者所有的三个盘片可能都使用同一个驱动器，CBF 选用参数仍然可以工作。

（2）如果你使用有 CBF参数的格式6 COPY 命令把 NEWDOS/80 2.0 版本的系统复制到另一个盘片上，则必须指定 FMT 选用参数。若不这样做，那末 BOOT/SYS 和 DIR/SYS 的信息就可能是错误的。如果你只是复制一个或几个系统文件到一个已经有系统的盘片上去（已经有系统的意思是，它能够在支持引导的驱动器上正确地引导系统），那末就不必指定 FMT。这个情况的说明过去没有包含在 CBF 的资料中，而现在已经放入了资料内。

（3）COPY 命令允许在 NEWDOS/80 2.0 磁盘与 TRS-80 Ⅲ型 TRSDOS 1.3磁盘之间，或与提供相应 PDRIVE 特征说明（见 PDRIVE 命令 TI 参数M标志）的更高级的磁盘之间往

返复制文件。

11. 以前位于403EH单元的DOS系统ID（标识符）现在已移到了4427H。在NEWDOS/80 1.0中，403EH中放有80(50H)或128(80H)值。在NEWDOS/80 2.0中，4427H单元放有标识NEWDOS/80 2.0的值130（82H），而在442BH单元中放有值01（TRS-80 Ⅰ型）或03（TRS-80 Ⅲ型）。

12. NEWDOS/80 1.0的模块，包括所有的系统模块、BASIC模块以及系统盘片上支持的所有别的程序，没有一个可以在NEWDOS/80 2.0上使用。因此，在NEWDOS/80 1.0系统磁盘上的用户文件都必须复制到NEWDOS/80 2.0系统磁盘上，而不要复制任何老的1.0版本的系统模块。对于只有一个驱动器的用户来说，这是一个很艰巨的工作，但是，即使是配有多个驱动器的用户，也必须改造几个系统磁盘。对于每一个这样的磁盘，你可以使用以下的方法来复制文件。

（1）用一个修补后的NEWDOS/80系统盘片的副本作为系统和源磁盘，使用带有FMT选用参数的格式5或格式6的COPY命令，再复制一个副本。

（2）删除你不想保留的NEWDOS/80 2.0文件。你可以在上面的COPY命令（如果是格式6 COPY）中使用ILF参数来进行这项工作。为此要建立一个ILF文件，首先用NEWDOS/80 2.0系统盘上提供的NWD80V2/ILF文件，并用CHAINBLD/BAS或SCRIPSIT程序删除其中不想要的文件的内容。记住，请用一个不同的文件名存储此时得到的文件，这就是你要在COPY命令的ILF参数中引用的文件名。

（3）再用获得的磁盘作为目标盘，而用老的NEWDOS/80 1.0版本磁盘作为源磁盘，执行带有NFMT和XLF=NWD80V2/XLF:0参数的格式6 COPY命令。这就把你的所有文件从NEWDOS/80 1.0版本磁盘复制到NEWDOS/80 2.0版本的磁盘上，但是不复制任何NEWDOS/80 1.0的文件，因为它们都不包括在XLF文件中，在NEWDOS/80 2.0磁盘上包含的NWD80V2/XLF文件就是作这个用的，你可以通过SCRIPSIT或CHAINBLD/BAS程序来查看这个文件。

（4）如果你想把所得到的NEWDOS/80 2.0版本的系统磁盘（目前该磁盘上已有你的文件）也复制回老的NEWDOS/80 1.0版本磁盘上，则你应该使用格式5或格式6的COPY命令并带有FMT选用参数进行这项工作。这就把NEWDOS/80 2.0的系统和你的文件装回到具有老的NEWDOS/80 1.0的磁盘上。

B.4 TRS-80 Ⅰ型上的NEWDOS/80 1.0与TRS-80 Ⅲ型上的NEWDOS/80 2.0之间的不兼容性

1. 上面B.3节的大部分内容在此都适用，请在阅读这一节以前阅读一下B.3节。本节只涉及TRS-80 Ⅲ型的有关事项。

2. 多数经过修补的在NEWDOS/80 1.0下工作的用户程序，通过以下修改就可以在TRS-80 Ⅲ型NEWDOS/80 2.0下工作：

（1）在TRS-80 Ⅲ型NEWDOS/80 2.0上，必须放弃对4300H—43FFH范围内任何字节的访问，应改成不同的相应单元。这一区域现在是系统的扇区缓冲区，而不是NEWDOS/80 1.0所使用的4200H—42FFH区域。

（2）在TRS-80 Ⅲ型NEWDOS/80 2.0中，必须完全放弃使用4315H单元去设定

DEBUG 的工作状态。

(3)TRS-80 Ⅰ型 NEWDOS/80 1.0 中用于允许/禁止 BREAK 键功能的 4312H 字节，在 TRS-80 Ⅲ型 NEWDOS/80 2.0 中已经移到 4478H。所以不再使用 4369H 的位 4 来进行设置。

(4)在 TRS-80 Ⅲ型 NEWDOS/80 2.0 中，HIMEM 单元已由 4049H—404AH 移到了 4411H—4412H。

(5)在 TRS-80 Ⅲ型 NEWDOS/80 2.0 中，CLOCK 单元已由 4041H—4043H 移到了 4217H—4219H。

(6)在 TRS-80 Ⅲ型 NEWDOS/80 2.0 中，DATE 单元已由 4044H—4046H 移到了 421AH—421CH。

(7)在 TRS-80 Ⅲ型 NEWDOS/80 2.0 中，25ms 的 1 字节循环计数器已由 4040H 移到 441FH。即使中断实际上是每 1/30 秒或 1/25 秒发生一次，用户定时中断程序仍然在 25ms 基础上周期地增量。

(8)在 TRS-80 Ⅲ型 NEWDOS/80 2.0 中，用于将一个定时中断程序插入 NEWDOS/80 队列的 4410H 矢量已改成447BH（见 3.8 节）。

(9)在 TRS-80 Ⅲ型 NEWDOS/80 2.0 中，从 4318H 开始的 DOS 命令缓冲区已被改成从 4225H 开始。

3. TRS-80 Ⅲ型 NEWDOS/80 2.0 磁盘目录采用 TRS-80 Ⅰ型 NEWDOS/80 2.0 的格式，而与 TRS-80 Ⅲ型 TRSDOS 磁盘不相兼容。

4. TRS-80 Ⅲ型 NEWDOS/80 2.0 的 FCB 格式是与 TRS-80 Ⅰ型 NEWDOS/80 2.0 相同的，而与 TRS-80 Ⅲ型 TRSDOS 的 FCB 格式不兼容。

5. 在 TRS-80 Ⅲ型 NEWDOS/80 2.0 上，必须用以下的 PDRIVE 特征说明去读写现有的 NEWDOS/80 1.0 磁盘。当制备一个在 NEWDOS/80 1.0 上可读出的磁盘时，也必须使用此 PDRIVE 特征说明。

(1)PDRIVE, dn1, dn2, TI=AK, TD=E, TC=39, SPT=18, TSR=3, GPL=2, DDSL=17, DDGA=2

是用于 5 英寸、单面、双密度、40 磁道的磁盘的 PDRIVE 特征说明。对于 35、77 或 80 磁道，则令 TC 分别为 34、76 和 79。

(2)PDRIVE, dn1, dn2, TI=A, TD=A, TC=80, SPT=10, TSR=3, GPL=2, DDSL=17, DDGA=2

是一个 5 英寸、单面、单密度磁盘的特征说明。对于 35、40 或 77 磁道驱动器，则要相应设置 TC。

(3)PDRIVE, dn1, dn2, TI=A, TD=C, TC=80, SPT=20, TSR=3, GPL=4, DDSL=17, DDGA=2

是一个 5 英寸、双面、单密度、80磁道磁盘的特征说明。对于 35、40和77磁道驱动器，则要相应设置TC。

(4)注意！在 NEWDOS/80 1.0 上使用的 5 英寸、单面、双密度磁盘不能直接用于 TRS-80 Ⅲ型微计算机上。见 B.3 节之 7。

B.5 NEWDOS/80 2.0 与 TRS-80 I 型 TRSDOS 2.3 之间的不兼容性

1. NEWDOS/80 总是保持 FCB 的 NEXT 字段为 RBA 格式。TRSDOS 2.3 在低位字节等于 0 的任何时候，或者当前的写位置在一个已经改变但尚未修改的缓冲区中的任何时候，保持 NEXT 字段为一个 RBA 值。在其他多数情况下，TRSDOS 将保持 NEXT 字段等于 RBA 加 256。在很多情况下，混淆正好是由于不了解 NEXT 字段的真正意义而引起的。

2. NEWDOS/80 总是保持 FCB 的 EOF 字段为 RBA 格式，并且，它对于写到文件中去的每一个字节都要修改 FCB 的 EOF 字段（如果 EOF 确实要改变的话）。TRSDOS 2.3 虽然在写一个字节或记录的过程中连续修改低位字节，但只是在实际写扇区时，才真正修改了 EOF。因此，如果当前记录引起 EOF 的改变，则 EOF 有两个可能的值，这取决于当前扇区是正在等待着写数据还是写了当前扇区。通常，如果低位字节等于 0，则 TRSDOS 的 FCB的 EOF 值是一个 RBA 值；如果低位字节不等于 0，则为 RBA 加 256。

3. 在 TRSDOS 中，DEBUG 功能的允许或禁止仍然由设置 4315H 单元来进行。在 TRS-80 I 型 NEWDOS/80 中不用这个单元设置 DEBUG 的功能，若用户设置了则忽略它，而在 TRS-80 Ⅲ型 NEWDOS/80 中是不允许用 4315H 单元来设置 DEBUG 状态的。

4. 定时子程序的启动和禁止在两个系统中是以不同的方法进行的（对于 NEWDOS/80 使用的方法，请参阅 3.8 节和 3.9 节）。

5. TRS-80 I 型 TRSDOS 和 NEWDOS/80 基本上都使用同样的目录格式，只是 TRSDOS 仍被限制于 35 磁道的磁盘和一个 2 个 gr. 的目录，而 NEWDOS/80 使用了一些以前不用的字节和位。

6. 下表是第三章中定义的子程序表，对于 NEWDOS/80 2.0 和 TRS-80 I 型 TRSDOS 2.3,它们是共同的。在两个系统中，每个子程序执行的功能几乎相同。第三章以外的子程序，或者在 TRS-80 I 型 TRSDOS 中不使用，或者定义为不同的功能。这些共同的子程序为：

0013H, 001BH, 402DH, 4030H, 4400H, 4405H, 4409H, 440DH, 441CH,
4420H, 4424H, 4428H, 442CH, 4430H, 4433H, 4436H, 4439H, 443CH,
443FH, 4442H, 4445H, 4448H, 4467H, 446AH, 446DH, 4470H, 4473H

B.6 NEWDOS/80 2.0 与 TRS-80 III型 TRSDOS 1.3 之间的不兼容性

1. TRS-80 Ⅲ型 TRSDOS 磁盘与 NEWDOS/80 2.0 磁盘是完全不兼容的。用 TRS-80 Ⅲ型 TRSDOS 的转换程序，可以处理 5 英寸、单密度、单面、35磁道并且有一个开始于第 17gr. 组的 2 个 gr. 的目录的磁盘。文件也可以在 NEWDOS/80 2.0 和 TRS-80 Ⅲ型 TRSDOS 1.3之间来回复制，或者和更高级的磁盘之间来回复制，但要对此更高级的磁盘提供 PDRIVE 特征说明，并且对 TRS-80 Ⅲ型 TRSDOS 磁盘要设置 TI 标志M。

2. TRS-80 Ⅲ型 TRSDOS 1.3 已经在 FCB 的 NEXT 和 EOF 字段以及目录的 EOF 字段中使用了 RBA 值。由于对 FCB 处理的这种改变，NEWDOS/80与TRSDOS之间在 TRS-80 Ⅲ型上的兼容性较前有所改善。

3. TRS-80 Ⅲ型 TRSDOS 使用一个 50 字节的 FCB, 而 NEWDOS/80 2.0 仍使用老的 32 字节格式的 FCB。NEWDOS/80 可以使用此 50 字节 FCB 区域，但是 TRSDOS 将弄乱 32

字节 FCB 后边的 18 个字节。用户应了解两个系统的 FCB 的详细情况及它们之间的差别。

4. 用于允许或禁止 BREAK 键的字节，在 TRS-80 Ⅲ型 TRSDOS 中是在 42AEH，而在 TRS-80 Ⅲ型 NEWDOS/80 中是在 4478H，TRS-80 Ⅰ型的NEWDOS/80 中是在 4312H。如果该字节等于 0C9H，则允许 BREAK 键起作用；如果此字节等于 0C3H，则禁止 BREAK 键起作用。

5. 下表是第三章中定义的子程序表，对于 NEWDOS/80 2.0 和 TRS-80 Ⅲ型TRSDOS，它们是共同的。在这两个系统中，每个子程序执行的功能几乎相同。除了第三章中定义的子程序以外，或者在 TRS-80 Ⅲ型 TRSDOS 中不使用，或者定义为不同的功能。共同的子程序是：

0013H, 001BH, 402DH, 4030H, 4409H, 440DH, 441CH, 4420H, 4424H, 4428H, 442CH, 4430H, 4433H, 4436H, 4439H, 443FH, 4442H, 4445H, 4448H

6. 对于 NEWDOS/80 中提供的 BASIC 中的 CMD 功能与 TRS-80 Ⅲ型 TRSDOS 提供的 BASIC 中的 CMD 功能之间的比较，请参考 7.13 节。

7. 在这两种系统中，路径的处理有些不同，但只要按规定去用就可以了。在 NEWDOS/80 中，不执行 DUAL 命令。

B.7 其他说明

1. 可能有极少数的用户编制了一些系统子程序，并且，这些系统子程序要用 DOS 的系统程序装入程序来装入，那末这些用户应该知道，NEWDOS/80 2.0 使用系统的 FPDE 端口 (slots) 已经用到 SYS21/SYS。而 NEWDOS/21 和 TRSDOS 被限制于只有 14 个系统程序可用系统程序装入程序来装入，NEWDOS/80 允许有 30 个 FDE 端口，它们的分配仍延续老的 TRSDOS 建立的同一次序。启动处于与系统模块有关的这些目录位置中的子程序的代码，要放在 A 寄存器中送到系统中去。这个代码必须大于 1FH，并且是 uuubbsss 的 8 位 (bit——二进制位) 格式，其中：

sss+2=目录中含有 FDE 的相对扇区

bb×32(20H)=扇区中到 FDE 的偏移量

uuu=一个大于 0 的用户定义代码

今后提供给用户的 NEWDOS 将使用 SYS22/SYS 以上的一些系统程序，所以用户应从 SYS29/SYS 开始往下使用。

2. 所有 NEWDOS/80 支持的程序都使用 TRS-80 Ⅰ型 4049H—404AH 单元 (TRS-80 Ⅲ型为 4411H—4412H 单元) 中的 HIMEM 高端内存值作为内存储器的高限。

3. (只限 TRS-80 Ⅰ型) 在打开电源、复位或跳转到单元 0 的过程中，控制被传送到 ROM。为确定是否有磁盘控制器，ROM 程序测试单元 37ECH 的内容，即磁盘控制器状态字节。如果此值为 00H 或 FFH，则 ROM 确定系统为一个无磁盘系统，并进行无磁盘的 LEVEL Ⅱ BASIC 初始化。然而，00H 是一个有效的磁盘控制状态，它意味着控制器没有状态和驱动器已准备就绪 (灯亮)。为避免这个不想要的进入非磁盘 BASIC 的入口，请等到就绪灯灭了以后再按复位键。

4. 为了在分配额外的文件空间给文件时提高磁盘操作速度，NEWDOS/80 一次分配多

达 4 个 gr.。但是，这存在一个缺点。如果同一时期在相同的盘片上打开两个以上的新文件，则很可能用完文件空间，而关闭所有的文件将发现此盘片上还有空间，因为 CLOSE 释放了已经分配给文件但尚未使用的额外的 gr.。

5. 当 NEWDOS/80 移动磁盘臂的时候，对于最大磁道号现在还没有进行任何检查。如果磁道号超出驱动器的物理界限，则驱动器臂将敲击制动器，发出哐哐的声响，磁道号超出驱动器的实际磁道数多少就敲击多少次。因为 DOS 想多次进行 I/O，因此，在 I/O 宣告出错以前，这种状况会长达一分钟之久。当发生这种敲击时，为缩短敲击的时间，只要打开驱动器门，并等到驱动器停止旋转或宣告出错为止，然后关闭驱动器门。

6. BASIC 单步功能 (CMD″F=SS″) 是不允许那些与时间有关的功能进行工作的，例如 INKEY$ 循环。在执行 INKEY$ 的情况下，如果用户给 INKEY$ 函数输入一个非空键，同时输入使 BASIC 单步进行的 ENTER 键，则输入给 INKEY$ 的键被忽略，因为在 ENTER 键以前就已看过 INKEY$。同样，在 32 字符的显示方式中，单步显示也不工作。

7. FORMAT 命令的修正。参量 PFST 的 Y 和 N 是互相排斥的。

8. COPY 命令的修正。如果使用格式 6 的 COPY（带选用参数 CBF）命令把 NEWDOS/80 系统复制成一个新的系统盘片，则必须指定 FMT 参量，以便给系统文件分配所需的目录 FPDE、在与目录有关的需要位置中给系统文件分配磁盘空间、并且使系统文件具有放在文件 BOOT/SYS 中的相应信息。在建立一个 PDRIVE 特征说明与源盘的PDRIVE特征说明不同的系统盘的时候，都必须使用这种格式的 COPY 命令。

附录C　TRS-80 I 型NEWDOS/80 2.0的错误更正

本附录介绍怎样实施修补，来更正 NEWDOS/80 的系统模块和实用程序模块中的错误或消除不兼容性。

用户应注意，一些模块有多种版本，其修补随每个版本而异。所以，在实施一个修补以前，要首先用核对的方法来确定待修补模块的版本，再根据其版本实施相应的修补。如果一个模块的版本与修补通报中列出的那些版本都不符合，请不要贸然进行修补。

C.1　NEWDOS/80 修补通报的格式

在 NEWDOS/80 中，修补是用 6.1 节中讨论的 SUPERZAP 程序人工地进行的。用户应认真学习 6.1 节，并一定要学会 SUPERZAP 程序的使用方法。虽然使用 SUPERZAP 程序进行修补，手续很麻烦，但是，每个用户都有可能要使用 SUPERZAP 去修补损坏的磁盘文件，而且，当这种事故发生时，如果用户具有丰富的使用 SUPERZAP 的经验，那末被损坏的磁盘文件就能顺利地正确地得以修复。

NEWDOS/80 的修补通报是连续编号的，并注明该修补付诸应用的日期。一个修补或者是必须进行的或者是随意选用的，也可能是针对一个 NEWDOS/80 模块的（即 NEWDOS/80 主系统盘片上的一个文件）或者是针对非 NEWDOS/80 模块的。如果它是针对一个 NEWDOS/80 模块的必须进行的修补，并且你的 NEWDOS/80 系统盘片所注明的日期晚于该修补通报的日期，那末在你的系统盘片上通常（但不总是）已经作过这个修补了。

每个修补通报都有一个简短的理由说明，并说明它是必须进行的还是可以选用的。接着给出一个或多个修补区域，每个区域由以下三部分组成：

1. 修补区域的第一字节在磁盘上的位置。这个位置由三个参数组成，格式如下：
   filespec1, relsector, relbyte

其中：

(1)filespec1　给出要修补的文件的文件名或文件名/扩展符。

(2)relsector　是修补区域在文件中的相对扇区编号。relsector 以十进制表示。

(3)relbyte　是修补区的第一字节在扇区中的相对位置。relbyte 以十六进制表示，但是不用字母 H 作后缀。

例如：

DIR/SYS, 2,20　表示要修补的文件是 DIR/SYS, 修补区位于文件的相对扇区 2，修补区的第一字节位于该扇区的相对字节 20H。

EDTASM/CMD, 20,F6　表示要修补的文件是 EDTASM/CMD, 修补区的第一字节是文件的相对扇区 20H 的相对字节 F6H。

YOURFILE,0,88　表示待修补文件的文件名是 YOURFILE, 修补区的第一字节位于该文件的相对扇区 0 的相对字节 88H。

2. 修补区域的旧内容。每个字节将表示成两个十六进制数字，并且为了易于阅读，这些字节将至少用一个空格分隔开。如果在一个十六进制数字的位置上出现的是一个“-”号，则表示在它被修补以前，不知道也可以不必理会该数字位置上的内容是什么。

3. 要修补到该区域中去的新内容。其表示格式与上述旧内容的格式相同。

注意，修补区域的内容的第一字节和最后字节，在实际的修补中，大多数是不必修改的。给出这些首尾字节只是为了帮助用户去寻找相应的区域（无论在修补之前或之后），给用户提供更多的检验字节，以免发生差错。然而，在所有的修补通报中，并不一律遵循这种方法。如果当前修补区域连接另一个修补区域，或延伸到另一个修补区域，或者，如果修补区域开始于或结束于一个扇区的边界，或者修补区超出了一个扇区的边界，则通常不使用这种给出首尾字节的方法。

C.2 修补过程

要实施一个修补，请按以下步骤进行：

1. 首先为被修补的盘片至少复制一个备份的副本。在此之前，千万不要进行修补！

2. 执行 DOS 命令 SUPERZAP，即执行 SUPERZAP 程序。

3. 装配好含有待修补文件的盘片。

4. 输入 SUPERZAP 的功能代码 DFS。

5. 输入文件的 filespec（文件标识符），即修补区位置的第一个参数，它包括：

（1）文件名或文件名/扩展符。如果该文件已被重新命名，则使用新的文件名。

（2）存取通行字（如果需要的话）。

（3）驱动器编号。

例如：MYFILE/CMD. PASSWARD: 1

6. 输入修补区位置的第二个参数，即文件中的相对扇区号。

7. 当该扇区被显示出来以后（见下面的步骤 14），在显示器上寻找该修补区域，并验证其旧内容是否是修补通报中给出的内容。如果不是，则核对一下你想实施的修补是否已经完成了。因为有可能已经作过修补了，若是如此，则跳过当前的修补区并进入下一个修补区继续进行。

8. 当确信是旧内容时，打入 MOD××，但不要按 ENTER 键。×× 就是修补区位置的第三个参数，即扇区中相对字节编号。

9. 光标此时应出现在相对字节××的第一个十六进制数字上。如果光标没有出现，则重新打入 MOD××。如果光标出现在错误的数字位置上，请核对一下你是否确实输入了正确的位置参数。注意！当出现光标时，SUPERZAP 是在修改（重写）方式中，这时一定要留神你所按下的键。在修改方式中，向左、向右、向上、向下的箭头以及空格键都可以用于移动光标。

10. 要修改光标所在位置的十六进制数字，请直接打入新的十六进制数字（即按下相应的 0—9 或 A—F 键）。之后，光标会自动地进到下一个十六进制数字上。

11. 打入所有的新的内容，即重复步骤 10。

12. 如果对这些修改不满意，按 Q 键就可取消这些修改并返回到显示方式。

13. 当完成了这些修改并准备在盘片上修改它们时，请按 ENTER 键。然后，按Y键，

并且在显示出提示信息时，再按 ENTER 键。于是，SUPERZAP修改盘片上该扇区的内容，退出修改方式返回到显示方式。

14 当处于扇区显示方式（无光标）时：

（1）如果你想显示同一个文件的另一个扇区，请按K 键，再转到第 6 步继续往下进行。

（2）如果你想进入另一个文件继续修补，请按 J 键，再转到步骤 5 继续往下进行。

（3）如果你想返回到显示 SUPERZAP 程序的功能清单，请按 X 键。

（4）如果在这同一扇区中存在另一个修补区域，则转到第 7 步继续进行修补。

C.3 修补的初始装入

当你第一次收到 NEWDOS/80 系统盘片时，首先复制几个 NEWDOS/80 系统盘片的备份，然后才能按照 C.6 节给出的修补通报，对其中列出的 NEWDOS/80 的模块逐一进行修补。

对NEWDOS/80 模块的必须进行的修补虽然大多数可能已经装入了你的系统盘片，但是你仍然必须对此进行核对。这可以使用 SUPERZAP 程序进行测试，看一下最高编号的必须进行的对 NEWDOS/80 模块的修补是否已经装入。如果它已经装入，那末就可以认为所有低编号的对 NEWDOS/80 模块的必须进行的修补都已经装入了。但是，对于 NEWDOS/80 的供选用的修补和非 NEWDOS/80 程序的任何修补就不是这种情况。如果这一最高编号的必须进行的 NEWDOS/80 模块的修补还没有实施，则检查下一个较小编号的这种修补，直到你遇到一个已经装入的修补为止。然后由该修补开始（不包括该修补）按着修补编号的次序，由小到大逐一实施较高编号的必须进行的 NEWDOS/80 模块的修补。较高编号的修补也可能覆盖已经实施过的较低编号修补的一个区域再次进行修补。因此，在实施所有的较低编号的必须进行的 NEWDOS/80 模块的修补以前，决不要实施较高编号的必须进行的 NEWDOS/80 模块的修补。

与实施必须进行的 NEWDOS/80 模块的修补一样，你必须对那些准备在 NEWDOS/80 中使用的非 NEWDOS/80 模块实施必须进行的修补（如果有的话）。同时，还应该至少阅读一遍供选用的修补，由此，一旦在需要时能马上想到它们，并根据它们去进行修补。

C.4 修补的后续装入

当你收到一个由 Apparat 公司寄来的新的修补通报时，你就应该对 NEWDOS/80 模块和那些正在 NEWDOS/80 中使用的非 NEWDOS/80 模块实施新的必须进行的修补。还应再次从头至尾读一下新的供选用的修补。对于其中已经进行的修补则没有必要再重新阅读，因为修补是极少再作修改的，而且，如果它们真的被修改了，那末，通常会在后续的修补中指导你去修改它们。

记住，你的 NEWDOS/80 系统盘片也可能已经实施了一些较新的必须进行的NEWDOS/80 模块的修补，因此，与 C.3 节所述那样，请检查最高编号的新的修补，直到遇见已经装入的修补为止。然后，以上升的修补编号次序，开始装入更高编号的新的修补。

C.5 修补的复制

所有的用户一般都有许多 NEWDOS/80 的副本，并且单驱动器用户被迫在他们使用的

NEWDOS/80 的每一个盘片上都保持着 NEWDOS/80 系统。一旦新的修补被正确地装入一个 NEWDOS/80 的副本上，并且对这些修补已经核对完毕，那末用户现在就面临这样一个任务：或者修补所有的其他盘片，或者把修补过的文件复制到别的盘片上。使用带有 ILF 和 DFO 参数（DFO参数将在下面定义，因为它没有在 COPY 命令中进行定义）的格式 6 的 COPY 命令（带 CBF 参数），就可以进行修补的复制。我们使用以下例子作为这一复制过程的说明。

假设用最新的修补改变了模块SYS0/SYS, SYS2/SYS, SYS17/SYS, SYS14/SYS, BASIC/CMD 以及 DIRCHECK/CMD。修补是在一个 NEWDOS/80 的副本上实施的，并且对 NEWDOS/80 进行了核对，证实此修补是正确的。在本例的以下叙述中，这个盘片被称为修补后的盘片。

建立一个 ILF 文件（它就象一个链文件），它含有以下记录：

```
SYS0/SYS
SYS2/SYS
SYS17/SYS
SYS14/SYS
BASIC/CMD
DIRCHECK/CMD
```

这个文件被命名为 ZAPNAMES/ILF，并存放在修补后的盘片上。接着，建立一个含有以下两个命令之一的链文件：

```
COPY,0,0,,NFMT,DFO,CBF,ILF=ZAPNAMES/ILF :1   用于单驱动器系统
COPY,0,1,,NFMT,DFO,CBF,ILF=ZAPNAMES/ILF :1   用于双驱动器系统
```

这个文件被命名为 ZAPDUP/JCL，也存放在修补后的盘片上。这两个文件都可以用 CHAINBLD（见 6.6 节）程序或 SCRIPSIT 程序来建立。

修补后的盘片将被看作系统盘片和源盘片，并装配在 0 号驱动器上。要接受修补后的模块的 NEWDOS/80 盘片将被看作目标盘片，并且在双驱动器系统的情况下，把它装配在 1 号驱动器上。

然后，对每一个要接受修补后的模块的 NEWDOS/80 盘片（即目标盘片），执行 DOS命令：

```
DO,  ZAPDUP
```

这个 DO 命令将使得含在 ZAPDUP/JCL :0 中的 COPY 命令得以执行。由于该 COPY 命令指定一个 ILF 文件，所以只复制该 ILF 文件中列出的那些文件。并且，由于指定了 DFO 选用参数，因此只有那些预先已经存在于目标盘和源盘上的六个文件被复制。例如，如果 DIRCHECK/CMD 没有预先在目标盘上，则不把它复制到目标盘上。

在复制过程中，单驱动器系统的用户将不得不进行大量的反复的盘片装配工作。因此，最好在修补后的盘片（源盘片）上打一个特殊记号，以区别于其他的盘片。

双驱动器系统的用户则轻松得多，在每次进行盘片复制中，只要装配两次。

由于 DFO 选用参数在 COPY 命令中没有加以定义，所以在这儿进行定义。DFO 表示：只有已经同时存在于目标盘和源盘上的文件才被复制到目标盘上。

C.6 修补通报

＊＊＊＊· ZAP 001 ＊＊ 08/04/81 ＊＊V2M1 ＊＊＊

这是对 Microsoft 公司的 EDIT/CMD 的必须进行的修补，使它能够在 NEWDOS/80管理下运行。对于 TRSDOS 和 NEWDOS, 你必须分别保持这一文件的独立的副本，由于一个系统上的模块与另一个系统上的模块是不兼容的，虽然它们管理的文件是兼容的。这些改变是由于 FCB 的 NEXT 和 EOF 字段在 NEWDOS 中与 TRSDOS 中定义不同。

EDIT/CMD, 38, E8

把 7E E6 7E 7E 21 FE FF CA 58 78 EE 80 87 DA 5D 78

改成7E 60 69 D9 01 30 01 2B 29 07 30 01 23 18 04 78

EDIT/CMD, 32, A2

把· 78 E6 7F C2 28 72 78 EE

改成78 D6 01 30 08 18 04 EE

＊＊＊ ZAP 002 ＊＊ 08/04/81 ＊＊ V2M1 ＊＊＊

这是对 PENCIL/CMD 的必须进行的修补，以禁止 PENCIL 改变一个 DOS 单元。因为 PENCIL 对 DOS 的这个改变在 NEWDOS/80 和 NEWDOS/21 中都是不必要的。

PENCIL/CMD, 00, AE

把 F3 32 9B 46 C3

改成F3 00 00 00 C3

另外，以下两个对 PENCIL/CMD 的修补将使得：

(1)允许 PENCIL/CMD 正确地读目录。

(2)在 PENCIL/CMD 下能使用 MINI-DOS。

PENCIL/CMD, 05, 60

把 58 23 22 改成 58 00 22

PENCIL/CMD, 00, 61

把 54 22 16 40 21

改成54 00 00 00 21

这后一个修补阻止 PENCIL/CMD 使用它自己的键盘子程序，因此，能够发现 NEWDOS/80 中 'DFG' 调用 MINI-DOS 的缺省返回。

＊＊＊ ZAP 003 ＊＊ 08/04/81 ＊＊ V2M1 ＊＊＊

这是对 Radio Shack 的 SCRIPSIT/UC 以及 SCRIPSIT/LC 程序的必须进行的修补，以便在 NEWDOS/80 (和 NEWDOS/21) 下运行它们。用户应保持 NEWDOS 和 TRSDOS 的独立的副本，因为每个系统的程序与另一个系统是不兼容的，虽然文件操作是兼容的。其中前三个修补是必须的，因为 FCB 的 NEXT 和 EOF 字段保持不同的方法。如果你正在运行NEWDOS, 并且文件装载不足一个扇区，则检查一下是否确实实施了这些修补。第四个修补使 4049H—404AH 中 DOS 的 HIMEM 地址值被用作为 SCRIPSIT 程序的最高内存储器。第五个修补再次允许中断，于是可以由 SCRIPSIT 程序调用 MINI-DOS。

SCRIPSIT/UC, 11, 75 和 SCRIPSIT/LC, 11, 75

把 47 00 CD 6E 7A 4F

改成47 00 3A B9 7C 4F

SCRIPSIT/UC, 11, FB 和 SCRIPSIT/LC, 11, FB

把 B7 C4 EF 5D 79

改成B7 32 B6 7C C4

SCRIPSIT/UC, 12, 00 和 SCRIPSIT/LC, 12, 00

把 32 B9 7C 11 改成 EF 5D 00 11

SCRIPSIT/UC, 00, 63 和 SCRIPSIT/LC, 00, 63

把 7C 21 FF 00 25 7E 2E 77 AE 20 F9 22

改成7C 2A 49 40 00 00 00 00 00 00 22

SCRIPSIT/UC, 00, C3 和 SCRIPSIT/LC, 00, C3

把 57 F3 ED 改成 57 00 ED

＊＊＊ ZAP 004 ＊＊ 08/04/81 ＊＊ V2M1 ＊＊＊

这是一个供选用的修补，用于增加打开驱动器电源时的延时，使它满一秒钟，老的DOS中就需要使用这么长的延时。NEWDOS/80 在选中开电源时只要等待 1/2 秒钟，就可以使驱动器达到所需的旋转速度。如果用户认为用长一点的延时驱动器会工作得更好，就请应用这个修补。

SYS0/SYS, 04, C3

把 06 80 CB 改成 06 FF CB

＊＊＊ ZAP 005 ＊＊ 08/04/81 ＊＊ V2M1 ＊＊＊

这是对 APL80/CMD 的必须进行的修补，使它能够运行于 NEWDOS/80。因为 APL80/CMD 要访问一个 DOS 的内部子程序，而它在 NEWDOS/80 中所处的地址与 TRSDOS 和 NEWDOS/21 中的不同。

APL80/CMD, 12, 74

把 21 A2 4B 改成 21 B4 4B

APL80/CMD, 14, 52

把 21 A2 4B 改成 21 B4 4B

＊＊＊ ZAP 006 ＊＊ 08/04/81 ＊＊ V2M1 ＊＊＊

这是对 EDTASM/CMD 的选用的修补，用于禁止 EDTASM 的键盘输入程序进行小写至大写的转换。EDTASM 中的大多数功能将仍然要求大写，但注释和括在单引号中的操作数（用于 DEFM 和 DEFB）将接受小写。

EDTASM/CMD, 28, E1

把 FE 61 D8 改成 FE 80 D8

＊＊＊ ZAP 007 ＊＊ 08/04/81 ＊＊ V2M1 ＊＊＊

本修补通报不是实际的修补，而是向用户提供的一些信息。

1. 已为 NEWDOS/80 模块保留了一些供修补用的区域。Apparat 公司在资料中没有提到它们是不想让用户使用它们。因此要告诫用户，Apparat 将优先占用这些区域，并且在它的修补检验中将要求这些区域都为零。使用这些区域进行非 Apparat 公司提供的修补的用户，应小心地保持他们所做工作的记录，以便在将来当它们与 NEWDOS/80 修补发生冲突时可以进行对照检查。

2. DOS 命令 ATTRIB（见 2.3 节）有一个附加的选用参数 LRN=×××，在 2.3 节中没有说明。LRN=××× 指定文件中记录的新的逻辑记录长度，其中 ××× 是一个 1 到 256 之间的整数。现在，只有 DIR 命令使用记录长度，但是，如果用户在处理中采用的文件具有一个给定的记录长度，那末某些用户可以使用 DIR 命令来观察它。

3. MINI-DOS（见 4.2 节）不能在 DOS-CALL（见 4.4 节）中使用。当 DOS 处于 DOS-CALL 时，如果按下 DFG 三键，则忽略此联键操作的作用。

4. DOS 命令 ROUTE（见 2.43 节）已经作了修正，如果 ROUTE 命令中没有指定参数，则只显示那些已经存在的路径。

5. DOS 命令 COPY（见 2.14 节）格式 6（带 CBF 参数）有一个新的选用参数 DFO。如果在命令中指定了 DFO，那末只有文件已预先同时存在于目标盘和源盘上时，才复制这些文件的内容。参数 DFO 与 FMT 是互相排斥的。

6. DOS 命令 FORMAT（见 2.22 节）有一个新的选用参数 RWF。如果在命令中指定 RWF，则不理会格式化过程中的所有错误，并且每个磁道都被格式化一次，而不论它们实际上是否完成了格式化。使用 RWF 参数往往是在用户希望擦去一个已损坏的盘片上的信息而又没有消磁器的时候。参数 RWF 与 KDN, KDD, DDSL, DDGA 以及 PFST 是互相排斥的。

*** ZAP 008 ** 08/04/81 ** V2M1 ***

这是对 PROFILE 的 INIT 模块的必须进行的修补，以改正NEWDOS/80送回的一个不同的错误代码，并且在从 4420H 调用返回时使 NZ 错误状态优先于 C 或 NC 状态。进行这个修改以后，修补过的模块仍可运行于 TRSDOS 中。这个修补是用于三个不同版本的同一模块的。

INIT, 00, 54

把 44 38 09 28 0F FE 1A 28 13 C3 3F 52 CD

改成44 20 19 00 00 00 00 00 00 00 30 08 CD

如果检验以上内容发现不符合，则用下面的一个位置或另一个位置试一试。

INIT, 00, 63 或 INIT, 00, 64

把 44 38 11 28 07 FE 1A 28 13 C3 47 52 CD

改成44 20 19 00 00 00 00 00 00 00 38 08 CD

对那些感兴趣的用户和某些 PROFILE 模块，下面的选用修补将重新允许中断，于是可以在 PROFILE 和 NEWDOS/80 下使用'123'和'DFG'联键操作。

PROFILE/CMD, 11, 45

把 F3 CD 改成 00 CD

*** ZAP 009 ** 08/04/81 ** V2M1 ***

这是对 Racet 的 Infinite Basic IBLOAD/CMD 模块的必须进行的修补，使它能在 NEWDOS/80 的管理下运行，就象在 TRSDOS 中运行一样。

IBLOAD/CMD, 08, 15

把 15 21 B6 79 11 F4 79 CD 36 44 28

改成15 11 F4 79 CD 13 00 32 B6 79 28

*** ZAP 010 ** 08/04/81 ** V2M1 ***

这是对 Racet 的DSM模块DSMB/CMD的必须进行的修补，以适应NEWDOS/80 的要求。因为 NEWDOS/80 要求文件标识符的类型（即文件名扩展符）最多是 3 个字符，并且要求用一个终止符来结束文件标识符。这个修正只适用于文件标识符 DSMC/CMDPATCH，由此，如果一个用户要用一个不同的文件标识符对此进行修正，那末必须在该文件标识符的末尾应用这三个字节。

DSMB/CMD, 03, C2

把 44 50 41 改成 44 03 41

＊＊＊ ZAP 011 ＊＊ 08/04/81 ＊＊ V2M1 ＊＊＊

这是一个必须进行的修补，以便允许 VISICALC 在 NEWDOS/80 管理下运行。所得到的修补过的模块将不能在 TRSDOS 或 NEWDOS/21 管理下运行，所以应该为这两个系统保持两个不同的版本。这一修补要影响到 NEWDOS/80 对用户的 25ms 中断程序（见 3.8 节和 3.9节）的处理。

VC/CMD, 03, 2B

把 09 3E 00 21 20 03 22 51 9E C9 79

改成09 18 BB 00 00 08 08 C3 EF 9B 79

VC/CMD, 75, 15

把 11 28 9C 22

改成11 1E 55 22

VC/CMD, 75, 24

把 C9 3E 03 C3 13 44 CD 4E 53 F5 CD 16 9C 28 0E 3E

改成C9 11 1E 55 C3 13 44 CD 4E 53 CD 16 9C C8 00 3E

以下是一个对 VISICALC 的必须进行的修补，以适应 NEWDOS/80 与 TRSDOS 在送回的一个错误代码上的差异，这个错误代码是引起 VISICALC 的目录搜索挂起来的错误代码，如果四个驱动器中的任何一个都没有联接在系统上或者虽然联接在系统上但没有装上盘片，就会得到此错误代码。这个修补后的模块与 TRSDOS 不兼容。

VC/CMD, 73, 01

把 C9 FE 18 20

改成C9 37 C9 20

＊＊＊ ZAP 012 ＊＊ 08/04/81 ＊＊ V2M1 ＊＊＊

这是一个供选用的修补，用于增加或减小双密度盘片格式化的灵敏度。根据你使用的接口、驱动器和盘片的可靠性，可以选择以下三个字节模式中的一个。字节模式越灵敏，则边缘盘片不能格式化的可能性就越大，格式化成功的可能性就越小，最终该盘片将不能使用。字节模式越不灵敏，则边缘盘片不能格式化的可能性就越小，格式化成功的可能性就越大，最终该盘片就能付诸使用。

字节模式为：

1. E5 E5

格式化过程中灵敏度最小。这是单密度盘片的标准模式。

2. 5B 5B

格式化过程中灵敏度中等。这是 TRS-80 Ⅲ型 TRSDOS 的模式。

3. 6D B6

格式化过程中灵敏度最大。这个模式将极其严格地进行磁盘格式化，因此，如果接口、驱动器和磁盘不是优质产品，则磁盘的30%或30%以上将不能格式化。

用户应根据平时格式故障相对于磁盘故障的频率，确定使用以上三个2字节模式中的哪一个，并把这两个字节插入到以下位置中。首先要检查三个字节模式中的哪一个已经装入这些位置中。每个位置要接受两个字节，每个位置的前导字节是F5,用以核对每个位置。要插入字节模式的位置是：

SYS6/SYS, 31, D9

SYS6/SYS, 31, F3

＊＊＊ ZAP 013 ＊＊ 08/03/81 ＊＊ V2M1 ＊＊＊

这是一个供选用的修补，它使得COPY命令使用TRS-80 Ⅰ型盘片时，不用TRSDOS 1.3的目录格式，而改用TRSDOS 1.2或更早版本的目录格式。进行这个修补以后，如果用户想使COPY命令再处理TRSDOS 1.3格式的TRS-80 Ⅰ型的磁盘时，则必须改回这个修补以前的内容。

SYS6/SYS, 14, 75

把 01 00 7B 改成 01 01 7B

SYS6/SYS, 14, C8

把 01 00 40 4E 01 13 00

改成01 13 40 4E 01 00 00

SYS6/SYS, 20, EA

把 61 C8 5E 改成 61 C9 5E

＊＊＊ ZAP 014 ＊＊ 08/08/81 ＊＊ V2M1 ＊＊＊

这是一个必须进行的修补，以改正BASIC中的一个错误，否则在使用简写的命令A, D, E和L时发生SYSTAX（句法）错误。BASIC中的这个错误恰好是在NEWDOS/80 2.0投放市场以前在改正另一个更明显的错误时引入的。

SYS18/SYS, 02, 31

把 2C D7 28 06 FE 3D 20 F9 18 23 E1

改成2C CD CF 65 28 27 FE 0D 20 F7 E1

BASIC/CMD, 14, D8

把 00 00 00 00 00

改成23 7E FE 3D C9

＊＊＊ ZAP 015 ＊＊ 08/18/81 ＊＊ V2M1 ＊＊＊

这是一个必须进行的修补，用于改正格式5的COPY命令中的错误。如果目录的起始gr. 组编号与PDRIVE特征说明中的不一致，即使目录是正确的，这个错误也使得BOOT扇区不接受此目录。这个错误可在DIR命令中由DIRECTORY READ ERROR(目录读错误)表明。

SYS6/SYS, 05, 96

把 64 01 00 01 18

改成64 CD 80 5C 18

SYS6/SYS, 15, 46

把 00 00 00 00 00 00 00 00 00 00 00 00 00 00

改成01 00 01 B7 C0 13 13 1A 32 BC 64 1B 1B C9

*** ZAP 016 ** 08/18/81 ** V2M1 ***

这个修补用于改正PDRIVE命令中的错误，这个错误使得PDRIVE中指定的第二个驱动器编号大于9时，产生SYSTEM PROGRAM NOT FOUND（系统程序没找到）错误。

SYS16/SYS, 00, 6A

把 C3 1A 52 CB

改成C3 4B 50 CB

*** ZAP 017 ** 08/18/81 ** V2M1 ***

这是一个必须进行的修补，以改正DOS命令COPY和FORMAT中的错误，否则命令将拒绝其中的选用参数DDSL。

SYS6/SYS, 01, FA

把 CB 70 20 改成 CB 52 20

附录D 词 汇 表

alpha 或 alpha character 字母

指英文字母，通常在涉及字符集 A—Z 和 a—z 时使用这一术语。

alphanumeric 字母数字

指英文字母和数字，通常在涉及字符集 A—Z, a—z 和 0—9 时使用。

bit 比特、位或二进制位

主存储器或磁盘存储器的最小可存取单元。一个比特所具有的值是 0 （表示复位）或 1 （表示置位）。四个连续的比特构成的一组就是通称的一个十六进制数字。连续的八个比特一组就是通称的字节。本书中，当在一个字节内定位一个比特时，通常约定位 7 是最左边的一位，位 0 是最右边的一位。在字节内比特的次序是从左到右对应位 7 到位 0 。这个概念在十六进制数字中也是如此，从左到右为位 3 到位 0 。

boot 引导

见 reset/power-on

BOOT/SYS

NEWDOS/80 使用的每个盘片上必须有两个控制文件，BOOT/SYS 是其中之一。见 5.1 节。

buffer 缓冲区

是主存储器内的一个区域，用来存放由磁盘读入的一个扇区的内容，或者存放将要写入磁盘的一个扇区的新内容。为此目的，每一个打开的 FCB 有一个 256 字节的缓冲区。字节方式的磁盘 I/O，例如用于输出/输入、标记项、固定项和字段项（如果记录长度小于 256）等文件，其实际的对缓冲区的读写操作是在必要时由磁盘扇区读写完成的，而不是在执行每一个 GET 或 PUT 语句，或 PRINT 或 INPUT 语句时完成的。

byte 字节

主存储器或磁盘存储器的最小可寻址单元。一个字节由 8 个比特构成。当给出一个字节的值时，通常表示成两个十六进制数字。在 NEWDOS/80 的资料中，尽管字符的含义有更大的局限性，但字节和字符这两个字通常可以交换使用。

chaining 链接

在 NEWDOS/80 中，把由磁盘文件（即通称的链文件）给出键盘输入字符的过程称为链接。见 4.3 节。

character 字符

在 NEWDOS/80 资料中可与字节交换使用，但字符通常用来代表含有可打印值（例如字母、数字或符号等）的字节。

close 关闭

在磁盘 I/O 中，关闭一个 FCB 或文件区的意思是解除由 OPEN 功能建立的程序与磁盘

文件之间的 I/O 链。

DEC 目录项代码 (Directory Entry Code)

DEC 是 1 字节的代码，用来确定一个特定的 FDE，并被 DOS 用来快速地在目录中定位该 FDE。当 FCB 是打开的时候，它的第 8 字节中含有该文件的 FPDE 的 DEC。每个 FXDE 的第二字节中放有同一文件的前一个 FDE 的 DEC，而每个第 31 字节等于 255 (0FFH) 的 FPDE 或 FXDE，在它的第32 字节中含有该文件的下一个FXDE的DEC。这个 8 比特的 DEC 的格式为：

rrrsssss

其中：

sssss＋2＝目录中含有 FDE 的扇区的相对扇区号。

rrr×32(20H)＝扇区中 FDE 的相对字节地址。

DIR/SYS

它是 NEWDOS/80 使用的每个盘片上必须有的两个控制文件之一。DIR/SYS 含有磁盘的目录。见 5.1 节和 5.6 节。

directory 目录

在 DOS 中，目录是指 DIR/SYS 文件的内容，该文件必须出现在 NEWDOS/80 使用的每个磁盘上。目录含有确定所有文件的控制信息，以及盘片上所有空间的空闲或已分配状态。如果目录被损坏或破坏了，那么磁盘上的其他信息对用户来说，通常就再也不能使用了。见 5.1 节和 5.6 节。

DOS 磁盘操作系统 (Disk Operating System)

虽然许多程序员都有足够的能力写一个直接与磁盘通讯的程序，但他们几乎总是愿意让另一个程序或程序的集合作为用户程序和程序使用的磁盘之间的一个媒介。这个媒介通常叫做 DOS，它可以大大简化用户程序与它使用的文件之间的 I/O。通常，如同在 NEWDOS 以及 TRSDOS 中那样，DOS 功能进行了很大的扩充，以致 DOS 成了计算机中的主要控制程序，并具有磁盘 I/O 控制以外的各种有用的功能，以完成对命令，即所谓 DOS 命令（第二章中讲述）的响应，或者对 DOS 调用（第三章中讲述）的响应。在 NEWDOS/80 中，DOS 工作于 4000H—51FFH 主存储区域，它的某些功能要使用 5200—6FFFH 区域，而实用程序 ASPOOL（假脱机程序）要一直用到内存储器的最高区。

DOS-CALL 操作系统调用

系指 DOS 状态，该状态是在用户程序调用 4419H 处的 DOS 子程序（见3.11和4.3节）去执行 DOS 命令或用户程序时进入的，可以有多级 DOS-CALL 状态。

DOS command 或 doscmd 操作系统命令

这是指第二章中讲述的 DOS 功能之一。DOS 命令的执行可以通过键盘打入或由当前执行程序的调用（见 DOS-CALL）来进行。

EOF 文件结束 (End Of File)

是一个文件的终点。有些文件有一个或多个指定的 EOF 字节，它标志文件的结束（例如，汇编程序源文件使用 1AH，BASIC 程序的非 ASCII 文本使用三个连续的 0 字节），但是，多数文件不指定 EOF 而完全依赖 FCB 或 FPDE 的 EOF 来标识文件的终点。如果文件是空的，则 EOF 值等于 0；如果文件有 1324 字节，则以 RBA（相对字节地址）表示的 EOF 值

是1324。在NEWDOS的FCB中，EOF是一个3字节的RBA值，它等于文件的最后字节的RBA值加1。存放在文件的FPDE中的EOF值不是RBA格式。请参阅5.7节（FPDE的字节4，21和22）和5.9节（FCB的字节9，13和14）。

EOL 行结束 (End Of Line)

是一行的终点。对于输入数据或一个命令来说，这通常是ENTER字符(0DH)。对于BASIC文本，是用一个0字节来结束一行。如果一行上设有一个明确的EOL字符，那么EOL意味着是这一行的最后一个字符的后面的字节。

EOM 信息结束 (End Of Message)

是一个信息的终点。EOM字符的代码是03。当信息的终点不是行的终点时，EOM用于结束信息。在遇到信息的终点时，不显示或不打印EOM字符，显示器或打印机也不前进一格。

EOR 记录结束 (End Of Record)

是记录的终点。EOR也以文件中的相对字节地址表示，它是该记录的最后字节的RBA加1。

EOS 语句结束 (End Of Statement)

是语句的终点。对于BASIC文本，用一个冒号结束语句。

extent element 范围元

它是FPDE或FXDE中的一个2字节的控制元素，它在分配给文件的磁盘存储区上指定1到32个gr.的连续区域。见5.7节FPDE的第23—30字节。

a n 文件区编号 (file area number)

fan是一个求得整数值（范围为1—15）的BASIC表达式，它指定哪一个文件区域被用于当前的BASIC功能。

FCB 文件控制块 (File Control Block)

FCB是一个含有关于文件信息的数据区，它控制一个程序和一个磁盘文件之间的I/O链。该I/O链由OPEN功能建立，由CLOSE功能解除，并供所有的磁盘I/O功能，包括GET, PUT, PRINT, INPUT, LOC等使用。FCB含有NEXT字段、EOF字段、缓冲区地址、保密信息和记录长度等信息。见5.9节。

FDE 文件目录项 (File Directory Entry)

在NEWDOS中，目录文件DIR/SYS的每一个扇区，除开始两个以外，都被分成八个32字节的控制区，此控制区被称作FDE。FDE或者是空闲的（可用于分配）或者用作为FPDE或FXDE。请参阅5.6节。

FF file FF文件

NEWDOS/80 BASIC中一个分成等长记录的BASIC固定项文件。

FI file FI文件

一个BASIC固定项文件，它不进行记录分段。

file或disk file或diskette file 文件或磁盘文件

文件是磁盘上的数据的集合。文件可以含有磁盘控制信息(如BOOT/SYS和DIR/SYS)、可执行的机器语言程序（如SYS0/SYS, BASIC/CMD和SUPERZAP/CMD)、BASIC程序（如CHAINTST/BAS）或用户的数据（如邮单、工资、库存物资)。所有文件的控制数据都

包含在文件 DIR/SYS 之中，每个文件都分配有一个 PFDE 和几个（或者一个都没有）FXDE。一个文件必须完整地存放在一个盘片上。磁盘空间是根据需要以 gr. 为单位分配给文件的。

filearea 文件区

文件区是一个 BASIC 的系统存储区域，它含有控制信息、一个 FCB 和一个 256 字节的缓冲区。文件区在磁盘文件操作期间被用于维护文件与 BASIC 程序之间的 I/O 链。这个 I/O 链是由 OPEN 语句建立的，供 PRINT, INPUT, GET, PUT, FIELD, EOF, LOF 和 LOC 等语句或函数使用，并用 CLOSE 语句解除。当对同一个文件打开两个以上的文件区时，各文件区互不影响。一个 BASIC 程序在任何一个时期最多可以打开 15 个文件区。对于 BASIC程序，实际可用的文件区数目是在启动 BASIC 时指定的(见 7.2 节)，其缺项值为 3 。

field item file 字段项文件

这是 NEWDOS/80 中使用的一个名称，在 TRSDOS 磁盘 BASIC 中，它被称为随机文件，因为在 NEWDOS/80 BASIC 中，所有的三种类型的文件：字段项、固定项和标记项文件都可以随机地或顺序地，或随机和顺序混合使用。字段项文件和固定项文件基本上是相同类型的文件，其主要差别在于两者所使用的 I/O 链的形式。对于字段项文件，文件项的定义完全通过 FIELD 语句完成。字段项文件总是分段成相同长度的记录，其长度可以为 1—256 字节。

file item 文件项

文件项是长度大于或等于 0 字节的文件存储单位，它含有一个数值或字符串。

filespec 文件标识符

NEWDOS/80 中使用的这一术语指的是文件名、文件名扩展符(或称后缀)、通行字（或称口令）和驱动器编号四者的组合。它用在一个 DOS 命令、BASIC 语句或未打开的 FCB 中指定一个文件。在此组成文件标识符的四个元素中，只有文件名是必需的，其他可以省略，省略时则取它们的缺项值。对于文件标识符的全定义及格式，请参考 2.1 节。

fixed item file 固定项文件

固定项文件和字段项文件基本上是相同类型的文件。其差别在于字段项或固定项在文件输入/输出中使用的 I/O 链的形式。在固定项文件处理中，文件项的定义完全取决于 GET或 PUT 语句中使用的 IGEL。有两种形式的固定项文件：FI 和 FF。见 8.4 节。

format 格式

除了 format 一字的许多其他定义以外，它也是用于对一个 NEWDOS/80 准备使用的空白盘片进行处理的一个命令。这个处理从磁性上把盘片划分成磁道，同时进一步把磁道再细分成 256 字节的扇区。根据驱动器的类型，盘片将含有 35、40、77 或 80 个磁道，同时根据驱动器类型和记录密度，每个磁道将含有 10,17,18 或 26 个扇区。

fp 文件定位 (file positioning)

fp 是 GET 或 PUT 语句的第二参数。fp 指定在文件定位状态期间要完成的文件定位。如果要进行数据传送，则文件定位状态要先于 GET 或 PUT 语句中的数据传送状态。见 8.4 节。

FPDE 文件主目录项 (File Primary Directory Entry)

关于 FPDE 的详细说明请参阅 5.7 节。在建立文件的时候，就在磁盘目录中建立一个

FPDE。如果一个文件在磁盘上是存在的，那么在目录中总有一个对应它的 FPDE。FPDE含有文件名、扩展符、通行字、保护级、EOF、前四个范围元和其他信息。当文件被删除时，FPDE 和任何有关的 FXDE 就都被解除。

FXDE　文件扩展目录项 (File Extended Directory Entry)

关于 FXDE 的详细说明请见 5.8 节。任何时候，文件的磁盘空间所需的范围元的数目超过四个时，就在目录中建立一个或多个 FXDE，以容纳额外的范围元，但每个 FXDE 最多有四个范围元。如果文件有 FXDE，则通过FPDE访问它们。在文件的磁盘空间需要量改变时，FXDE就按需要建立或解除。当一个文件被删除时，所有的与此文件有关的FXDE就都被解除。

GAT　gr. 分配表 (Granule　Allocation Table)

GAT 是目录部分的第一个扇区(即所谓 GAT 扇区)，它含有的信息是磁盘上每个 gr. 的空闲或分配的状态。见 5.6 节。

granule　本书中简写为 gr.

gr. 是可以分配给文件的或可以从文件中去除的最小磁盘存储单位(在有些书籍或资料中 gr. 被译成区或簇)。当一个文件需要磁盘空间时，就给它分配一个或几个 gr.。对于 NEWDOS/80，一个 gr. 由 5 个扇区，即 1280 个字节组成。

hash code　散列代码

DOS 中使用的散列代码是文件名和扩展符的 1 字节的编码，用于在 OPEN 过程中，在目录中快速寻找文件的 FPDE。散列代码存储在 HIT 扇区中。见 5.2 节。

Hexadecimal 或 hex　十六进制

一个使用 16 个数字的计数系统，而一般的十进制系统是使用 10 个数字。十六进制中使用的数字为 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E 和 F。使用十六进制的理由是，十六进制数是表示 4 个相邻比特的值的最容易的方法。下表规定了十六进制数字与 4 个比特之间的关系：

10 进 制	16 进 制	二 进 制
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100

10 进 制	16 进 制	二 进 制
13	D	1101
14	E	1110
15	F	1111

磁盘、文件或内存储器地址以及它们的内容在计算机领域中广泛使用十六进制表示法。虽然某些用户一点不学习十六进制也能理解它，但我们希望对此不了解的用户要学习一些入门知识，至少要能看得懂 SUPERZAP 和 DEBUG 的输出内容。在本书及 NEWDOS/80 资料中，一个十六进制值是用后缀字母 H 表示的（例如，13＝0DH, 256＝100H），除非另有说明。

HIMEM (High MEMory)

它是指（1）最高可用内存单元的地址。（2）存储 HIMEM 值的 2 字节内存区（TRS-80 Ⅰ型为 4049H—404AH, TRS-80 Ⅲ型为 4411H—4412H 单元）。（3）一个 DOS 命令的名称(见 2.5 节)。

HIMEM 以上的内存或者是不存在的，或者是为其他用途而保留的。用户在编写 Z-80 代码的程序时应注意此 HIMEM 值。

HIT 散列码索引表 (Hash code Index Table)

是目录部分的第二个扇区（即所谓 HIT 扇区），它含有磁盘上所有文件的散列代码。在打开文件的过程中，DOS 不是搜索整个目录去寻找文件的 FPDE，而是根据文件的文件名和扩展符来计算散列代码，在 HIT 扇区中把它找出来，然后根据它在 HIT 扇区中的位置直接找到含有 FPDE 的扇区。请参阅 5.6 节。

I/O 输入/输出 (Input/Output)

I/O link 或 I/O path I/O 链或 I/O 途径

在磁盘文件和内存之间，实际的磁盘输入/输出是通过 I/O 链完成的，此 I/O 链是由 OPEN 语句建立的，并用 CLOSE 语句解除，它供 GET, PUT, PRINT, INPUT, EOF 等语句或函数使用。在打开 I/O 链时，该 I/O 链的控制信息包含在 FCB 或文件区（它包含有 FCB）中。对于同一个文件，可以在同一时期打开多个 I/O 链，而每个 I/O 链对另一个 I/O 链是一无所知的。一个 I/O 链记得文件中它正在操作的位置，因此多个 I/O 链可以在同一时期对同一个文件进行操作。但请记住，一定要十分小心，因为每个 I/O 链对其他的动作一无所知。

IGEL 项组表达式表 (Item Group Expression List)

IGEL 是一个 BASIC 表达式表，在进行固定项或标记项文件处理时，用于 GET 或 PUT 语句中对应于一组文件项。见 8.4 节。

IGEL expression IGEL 表达式

一个 IGEL 表达式（通常是但不总是一个 BASIC 表达式）是对应一个文件项的 IGEL 的一部分。对于固定项文件或标记项文件，GET 或 PUT 语句中处理的每一个文件项，在IGEL 中都有一个对应的 IGEL 表达式。见 8.2 节。

IGELSN IGEL 序号 (IGEL Sequence Number)

这是一个 BASIC 文本行的行号。使用此行号的文本行含有当前 GET 或 PUT语句要处理的第一行或唯一的一行 IGEL。如果使用 IGELSN，那末它应是 GET 或 PUT 语句的第三参

量。在固定项或标记项文件的处理中，当两个以上的 GET 语句和 PUT 语句使用同一个 IGEL，而 GET 或 PUT 语句本身并不含有 IGEL 的时候，通常就要在 GET 和 PUT 语句中使用 IGELSN。

item group　项组

项组是一组文件项（可以有几个文件项或一个文件项也没有）。在 BASIC 中，一个项组是一个单独的 INPUT, PRINT, GET 或 PUT 等语句处理的一组文件项，它通常等价于逻辑记录。

len　长度

见 8.7 节和词汇 LRECL。它是 BASIC 的 OPEN 语句中的一个参量，它或者指定标准记录长度，或者指定最大记录长度。

logical record　逻辑记录

一组意义上相关的文件项。虽然文件数据在磁盘上是以物理次序排列成扇区的，但是程序员通常处理的数据是以逻辑相关来分组的，而不是以物理相关分组的。因此，当数据由文件中读出或写入文件时，通常是以逻辑记录为单位进行的。

LRECL　逻辑记录长度 (Logical RECord Length)

这是文件记录的以字节为单位的标准长度或最大长度。对于非 BASIC 文件，LRECL 的值为 0—255（0 代表 256），并存储在 FPDE 的第 4 字节（虽然从来不用）以及 FCB 的第10字节中。在 BASIC 中，LRECL 等效于 len（见 8.7 节）。

lump　gr. 组

它是 gr. 的一个集合，我们称它为 gr. 组。每个 gr. 组可以有 2—8 个 gr.。GET 扇区中前 192 个字节的每一个含有一个 gr. 组的空间分配信息或封锁信息，其中，根据每个 gr. 组的 gr. 数目，字节中的每一比特表示该 gr. 组的一个 gr. 的状态：不用、或者说明已分配/空闲、或者不存在/存在。在 NEWDOS/80 2.0 中造出这个定义是为了避免使用磁道和柱面等术语。见 5.6 节和 5.7 节第 23—30 字节。

marked item file　标记项文件

标记项文件是一个用前缀标记字节来识别每个文件项的长度和类型的文件。标记项文件与输出/输入、字段项或固定项文件有着明显的差别。标记项文件有三个子类型：MI、MU和 MF。见 8.6 节。

MF file　MF 文件

一个标记项文件，它分段成全部等长的记录。

MI file　MI 文件

一个不分段成记录的标记项文件。

MU file　MU 文件

一个标记项文件，它分段成不等长的记录。

null　空或空缺

参数或表达式的缺省。当用逗号分隔参数时，连着的两个逗号就表示一个空缺，即其间缺省了一个参数。

null character　空字符

一个值为 0 的字符或字节。

null string 空字符串

一个长度为 0 个字符的字符串或表达式。

open 打开

在磁盘 I/O 中，打开 FCB 或文件区是在程序和数据文件之间建立一个 I/O 链，并使用 FCB 或文件区（它含有一个 FCB）来保存这个 I/O 链的控制数据。虽然平常都习惯地称作打开一个文件，但是更正确应称作打开一个 FCB 或文件区，因为在磁盘文件中没有什么东西表示打开或关闭状态，或者表示为它而打开的 I/O 链的数目，由于对一个给定文件，可以同时打开几个 FCB 或文件区。由 OPEN 功能建立的 I/O 链要一直保持到由 CLOSE 功能把它解除为止。这个 I/O 链确定文件进行 I/O 的类型以及在文件中的什么地方进行。因此，如果对同一个文件建立不同说明的 I/O 链，那末就可以使用同一个文件数据，由每一个不同的 I/O 链进行不同的解释。

partial record I/O 部分记录 I/O

这是指在部分的逻辑记录中而不是在整个逻辑记录中进行 I/O。在 BASIC 中，对于标记项文件和固定项文件，GET 和 PUT 语句可以用这种方式进行操作，虽然它们通常是以全记录 I/O 方式进行操作的。

patch 修补

请参阅词汇 ZAP。

power-on/reset 通电/复位

见 reset/power-on。

print/input file 输出/输入文件

用 PRINT 语句写入和用 INPUT 语句读出的磁盘文件。

record segmented file 记录分段文件

一种可以用 BASIC 分成逻辑记录的文件。这种文件类型有：字段项、MF、FF 和 MU。

REMBA 记忆字节地址 (REMembered Byte Address)

请参阅 8.10 节。

REMRA 记忆记录地址 (REMembered Record Address)

请参阅 8.10 节。

RBA 相对字节地址 (Relative Byte Address)

一种在文件、记录、控制块等中间寻址的方法，其中，寻址是从 0 而不是从 1 开始的。单元（指文件、记录、控制块等）的第一个字节的 RBA 为 0，单元中的第 n 个字节的 RBA 为 n−1。在 NEWDOS 中，RBA 用于表示 FCB 中的EOF字段和 NEXT 字段。这种在 FCB 中使用 RBA 的方法就是 NEWDOS 和老版 TRSDOS 之间的主要差别。在 BASIC 中，RBA用于文件定位(见 8.4 节)，其中，fp=!rba、!$rba 或 !# rba,而 rba定义为一个 BASIC 表达式，其值为 0—16 777 215 之间的数，并表示从文件起始位置开始的相对字节位置。

reset/power-on 复位/通电

也称作引导 (boot)。这是指计算机自动执行初始化，它发生在按下计算机的复位按钮或给计算机通电的时候。而实际上，当你给计算机通电时是决不应该把盘片放在任何驱动器中的，一般应在通电以后再把系统盘片放入 0 号驱动器并按复位按钮。在大多数情况下，NEWDOS/80 处理通电后的复位与其他任何时候的复位是一样的，但也有一些差别，最明显

的是通电后的复位要求用户进行日期和时间的设置。

在复位/通电过程中，ROM 的引导程序收到来自计算机硬件复位逻辑的控制，并把装配在 0 号驱动器上的盘片的第一扇区读入 DOS 系统缓冲区（TRS-80 Ⅰ型为 4200—42FFH；TRS-80 Ⅲ型为 4300—43FFH)。这 256 个字节含有 NEWDOS 的引导程序，它接受来自计算机 ROM 的控制，然后把 NEWDOS/80 的内存储器常驻模块 SYS0/SYS 读入内存，接着把执行控制传送给 DOS 复盖区中的 SYS0/SYS的初始化程序。NEWDOS/80 使用当前的 SYSTEM 和 PDRIVE 的规定进行初始化。在完成初始化以后，显示 "NEWDOS/80 READY"，或者 DOS 开始执行由 DOS 命令指定的 AUTO 功能。请参阅 2.4 节。

sector 扇区

在 NEWDOS/80 中，磁盘上数据的存储实际上是以所谓扇区，即 256 字节为一组进行的。实际的磁盘读和写是用全扇区进行的，通常是一次读写一个扇区。

SOR 记录的开始 (Start of Record)

是记录的开始字节。所有的 MU 文件的记录都使用 SOR 项，即一个 70H 字节开始。

track 磁道

这是在盘片旋转一周期间，磁盘驱动器读写头通过的磁盘存储器的单位。一个盘片在格式化过程中在磁性上被划分成一个个同心圆环的磁道（TRS-80 Ⅰ型的标准是 35 磁道，TRS-80 Ⅲ型为 40 磁道)。格式化也把每个磁道在磁性上分成一个个 256 字节的扇区，它们随后将含有各种类型的数据。

user segmented file 用户分段文件

一种不能由 BASIC 分成逻辑记录的文件。这些文件类型有 FI 和 MI。如果这些文件类型被分段成记录，则它的完成完全依赖于程序员而不是 BASIC。

whole record I/O 全记录 I/O

全记录 I/O 是在单个的 INPUT、PRINT、GET 和 PUT 语句执行过程中读/写整个逻辑记录。这是这些语句的正常过程。请参考部分记录 I/O。

zap 修补

修补是指改变磁盘上的数据或程序的目的代码(而不经重新编译)。zap 的另一个意义是指 Apparat 公司为了修正错误或消除不兼容性等目的而不定期发布的修改通报。请参阅附录 C。