软盘加密与解密新技术

范修维 著

清华大学出版社

软盘加密与解密新技术

范修维　著

清华大学出版社

(京)新登字 158 号

常驻程式之威力——磁片保护、破解新技术(繁体字版)
软盘加密与解密新技术(简体字版)

范修维 著

图书在版编目(CIP)数据

软盘加密与解密新技术/范修维著。—北京:清华大学出版社,1994
ISBN 7-302-01474-4

Ⅰ.软… Ⅱ.范… Ⅲ.①磁盘-保密编码②保密编码-磁盘
Ⅳ.TP333.3

中国版本图书馆 CIP 数据核字(94)第 01148 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)
印刷者:北京密云胶印厂
发行者:新华书店总店北京科技发行所
开 本:787×1092 1/16 印张:29.25 字数:687 千字
版 次:1994 年 6 月第 1 版 1995 年 3 月第 3 次印刷
书 号:ISBN 7-302-01474-4/TP·582
印 数:11001—16000
定 价:39.00 元

序

早在 APPLE 电脑盛行时，磁盘保护技巧可谓千变万化，手法也层出不穷，举凡半磁道、1/4 磁道、螺旋磁道等千奇百怪，无奇不有。但道高一尺、魔高一丈，仍旧有拷贝软件可用来解密并拷贝，例如，Locksmith，nibble away 等。但在 16 位电脑盛行的今日，为何没有这种万能的拷贝软件，难道是解密技术不够？还是加密保护手法再度翻新？于是，它便吸引我走入软盘破密与加密保护的研究中。

在研究过程中经常感到孤掌难鸣，每当遇到困扰想找人研究时，得到的回答往往是“这方面我没研究”“这种东西太难”，于是只好一本又一本地找寻相关书籍，一次又一次地进行实验。皇天不负苦心人，终于让我找寻出一些端倪。

其实，软盘加密及破密并不难，只是缺少了一些提示与灵感而已。我相信，一定有很多人对这方面的研究亦有兴趣，而他们可能亦遭遇同样的问题。这也就是激起我撰写本书之动机。

在此，我要感谢廖鸿图先生，他促成我开始撰写本书，同样地，亦感谢儒林书局的杨镜月先生及陈淑时小姐，没有他们的协助，本书无法如此顺利得以付梓。感谢好友邵致平、柯博昌、邱孟佑、陈启荣，他们在我研究的过程中提出了不少问题，因此让我得到更多宝贵的经验。最后，我衷心的感谢三舅陈顺钦先生，以及我最敬爱的爸妈，没有他们，今天我也不会走入计算机科学的领域里。当然，还有我的女朋友陈汶华小姐，在撰写本书的期间，她不断地给我鼓励与支持，她——是我最大的精神支柱。

范修维

目　录

第一章　驱动器工作基本概念

1-0　引　　言

相信各位读者一定有过这类经验：好不容易存够了钱去买一套软件（或 GAME），为了预防在意外的情形下损坏磁片（即软盘）而急欲制作一份备份，但却因为该磁片设有防拷保护，无法备份，只好摇头叹息，并祈求上天保佑让这张磁片的寿命能维持长一点；或者，可能您已经有了硬盘，想将该软件拷贝至硬盘，以便今后之应用，但不幸的是，每次要执行时都要检查 key disk 而感到非常不便，甚至因 key disk 损坏而使得该软件无法执行，所有的投资就此报销。这时，或许您会抱怨为何磁片要设这些保护（加密）；或许您亦尝试着去解除这些保护但却因功力不够而放弃；也或许您会激动地想去买台拷贝机，但又因价钱太高而裹足不前；也或您也为此去购买了几套专门解密的软件，但也因其成效不理想而灰心……。不过，请别失望，当您购买了本书之后，您就等于拥有了一套强而有力的工具：本书不但介绍有关对磁片保护的原理，亦讲述破解（即解密）之道，并且教您编写解密的程序。另外，本书并公开一份完整的程序，专门对付磁片保护，这是历经长期的研究而完成的，试用至今尚无任何保护程序可以躲过其破解，威力可谓凶猛之至！

虽然一般消费者非常讨厌磁片之保护，但尚有一些程序设计师却为了防止非法拷贝而伤透脑筋。甚至，为了保护程序而成立了专门的俱乐部（例如 Crazy Club），专门研究如何防拷，因此保护程序就愈来愈刁钻，愈来愈古怪，让别人无法追踪（Trace）其程序。但保护再严，仍旧有人破解，尤其对于国内的软件而言，其保护功力均不甚深厚，再加上本书程序之公开后更无所遁形。不过请别着急，本书亦提供几种保护方法，可以对付本书所谈之破解程序，使您之心血结晶不致遭到剽窃！

读到这里，或许会有读者会问，我所说的保护与破解是否有所矛盾？本来，保护与破解就是一场永远都打不完的头脑战争，保护再深的软件终究有其破解之道，破解再强的程序终究会有更强的保护程序来防止，就此展开的破解保护之战永不平息。要强调的是，保护与破解并非完全对立的，而是一体两面，想要破解必先要了解如何保护，而想要做到保护也必先了解破解之法才可想出因应之道。保护与破解没有所谓真正的赢家，完全是看谁所下的工夫深而定。

为此，本书提供了有关保护及破解的基本常识，并以一些实例解说，最后还提供了完整的程序供读者参考。

在此，笔者还要强调一点，此破解程序虽然好用，但也请尊重别人的知识产权。本破解程序就像一台影印机一般，只是为您提供应用上的方便，而不是向您提供做非法拷贝的工具，请读者务必牢记。

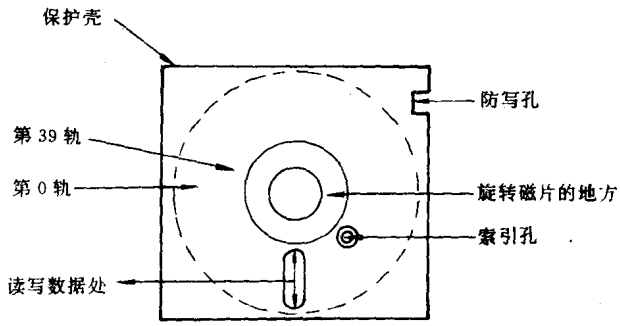
1-1 磁片的构造与磁盘的运行

本书并不打算介绍一些深奥、难懂又难以应用的知识，而是向您介绍在研究加密保护及破解的过程中会用到的知识。另外，由于目前的磁片保护均为360KB之磁片，故本书即针对此种磁片作为说明。

我们都知道，刚买来的磁片是一张空白的磁片，必须经过格式化之后，才成为双面可读可写的磁片：每面40磁道，每磁道有9个扇区，每个扇区可存放512 bytes的数据。

不知读者是否曾想过这个问题：格式化磁片究竟是什么样的动作呢？我们要存放一批数据又是如何进行的呢？为何一般的磁片可用disk copy拷贝，而设有保护的磁片却无法拷贝呢？磁片上存放的数据不都是"0"或"1"吗？若我们令驱动器读一张磁片，读到"0"就写"0"，读到"1"就写"1"，如此磁片不就可以拷贝了吗？但为何却又行不通？问题究竟出在哪里？

要解答上述的问题，我们必须先从磁道，扇区的形成加以探讨。我们拿出一张360KB的磁片来观察：



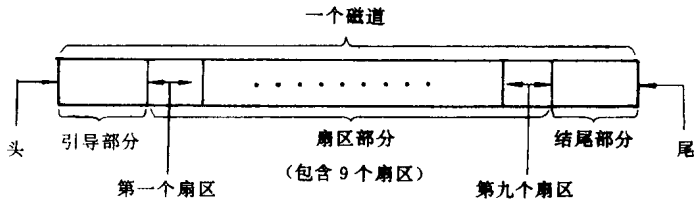
用手旋转磁片，可在磁片读写处的右方看到有一小洞，继续旋转磁片，可在该小洞发现磁片本身亦有一个更小的洞，此即为索引孔。驱动器无论在作读写或格式化的动作时，均由此索引孔作为起始处。

正常的格式化软盘之动作，是经过下列步骤：

1. 驱动器开始运转。
2. 由索引孔检测，是否到了读写的起始处。
3. 由磁头写下一堆引导信号。
4. 写下扇区识别字段的信号。
5. 写下数据字段的信号。
6. 在4、5两个动作完成9次后即格式化了9个扇区。
7. 将该磁道剩余的部分以无用的数据填满。

经过上述几个步骤后即完成了一个磁道的格式化，我们亦可经由上述步骤得知一个磁道内所包含的数据及所划分成的部分可分为：

1. 引导部分；
2. 扇区部分；
3. 结尾部分。



其中，扇区部分包含了 9 个扇区。

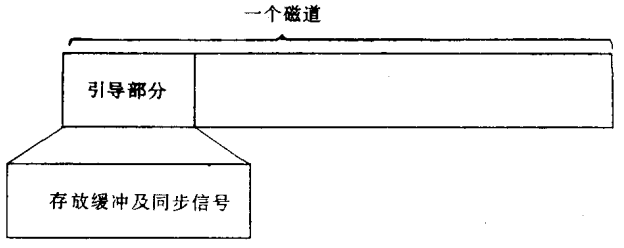
为了对磁道的内容能有更深入的了解，以下我们针对各个部分再作进一步的说明。

一、引导部分

此部分是由磁片作格式化动作时，由磁盘所写下的一些信号，它是在驱动器检测到索引孔后开始写，大约有 150 bytes 长，其主要作用为读写数据之缓冲及同步。

在 PC 上驱动器的读写均以检测索引孔作为起始，但通常在检测到索引孔后无法立即作读与写的动作，这时就需要有一段缓冲区域，使磁盘在读写数据时不致读错数据或写错地方。有了缓冲区域之后当然需要有同步区域，使驱动器知道从哪里开始才是真正可供读写的位置。

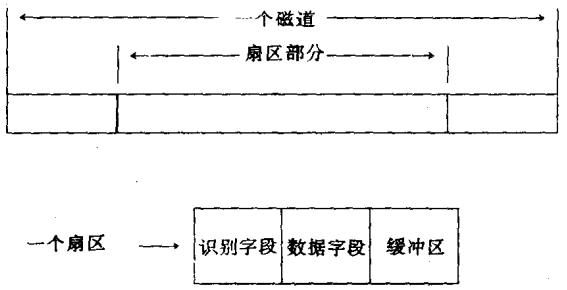
关于引导信号的内容在此就不再深究，因为它对我们目前的保护及破解技术并没有影响，读者若有兴趣可自行参考一些相关书籍。



二、扇区部分

此部分包含了 9 个扇区，而每个扇区又可再细分三部分：

1. 识别字段；
2. 数据字段；
3. 缓冲字段。



在识别字段中，主要存放了四个参数：C、H、R、N。

C：磁柱面(Cylinder)，亦可称为磁道(Track)

H：磁头(Head)

R：记录区(Record)，亦可称为扇区(Sector)

N：数据数目(Number)

这四个参数标明了该扇区的识别号码，当驱动器在读写数据时就由此识别字段来辨别是否为该数据存取的位置。此识别字段的内容只能由磁片格式化的时候写上去，往后即无法更改，除非重新格式化磁片。在正常的格式化后，其C、H、R、N之内容及意义分别如下：

C：存放该磁道所在的编号，其值为0到39。

例如：在第0道的所有扇区识别字段其C值均为0，第1道所有的扇区识别字段其C值均为1，依此类推。

H：存放0或1。

例如：在正面的所有扇区(不管是哪一磁道)识别字段之H值均为0，而反面之所有扇区识别字段之H值均为1。

R：存放该扇区之编号，其值为1到9。

例如：在任一磁道中第一个扇区，其识别字段之R值为1，第二个为2，…，最末一个扇区之R值为9。

N：存放该扇区之数据字段中可供使用者存放数据之大小的编号。

当N＝1时，代表该扇区可存放256 bytes的数据

当N＝2时，代表该扇区可存放512 bytes的数据

当N＝3时，代表该扇区可存放1024 bytes的数据

当N＝4时，代表该扇区可存放2048 bytes的数据

当N＝5时，代表该扇区可存放4096 bytes的数据

或者，可由下列公式求得：

$$\text{byte 数目} = 128 \times 2^{N}\text{(注 1)} \qquad \text{公式(1)}$$

在正常情况下，不论是在哪一面，哪一道，哪一个扇区，其N值均为2，也就是一个扇区可供使用者存放512 bytes的数据。

注 1：当 N=0 时，并非代表 128 bytes，其 byte 数目将由另外之参数得来，此参数往后将会提到。

以上所提到的 C、H、R、N 之值均为正常格式化之后所得到。为何说是在“正常情况”下得到的呢？因为在实际上，C、H、R、N 之值可以自由地设定，而不是一些固定的值。此种改变 C、H、R、N 之值可以用来作为一种保护的技巧，但要达到改变 C、H、R、N 值之方法，并非以 DOS 的 Format 命令可以完成，必须通过 BIOS 之 INT 13H 之中断服务程序(Interrupt Service Routine，简称 ISR)来完成，此种技巧将在下一节详述。

虽然 BIOS 提供了一些特殊格式化之功能，但却没有提供读取识别字段的功能，这是一件非常遗憾的事，或许，BIOS 故意不提供此项读取识别字段的功能就是为了让程序在保护的领域下有更大的发展空间吧！

为了克服 BIOS 无法提供读取识别字段的功能，笔者写了一个 DISK. EXE 的程序，它不利用 BIOS INT 13H 之 ISR，而是另写一个新的 BIOS，它可读出扇区识别字段的内容，填补了 BIOS INT 13H 的不足。不过，在此本书并不打算教读者如何改写 BIOS(因为此部分之内容相当深奥，并非一般人所能理解)，而是利用它来作一些说明与验证。此 DISK. EXE 程序并未附于本书中，或许在不久的将来笔者能再以专著为读者作更详尽的解说。

实验与观察

首先，我们拿一张 360K 的磁片，用 DOS 的 Format 命令去格式化(注意：要格式化成 360K bytes 必须用“/4”之参数)，其格式化以后的磁片为正常、无保护、并且可以拷贝的磁片：(见图 1-1-1. SCR)

```
D>format a:/4 ———————— 利用 AT 来 format 360k 的磁片须加上此参数
Insert new diskette for drive A:
and strike ENTER when ready

Format complete

    362496 bytes total disk space
    362496 bytes available on disk

Format another (Y/N)?n
D>dir a:

 Volume in drive A has no label
 Directory of  A:\

File not found

D>
```

图 1-1-1. SCR

再来，利用笔者所写的 DISK.EXE 程序来读出扇区之识别字段（见图 1-1-2.SCR）：

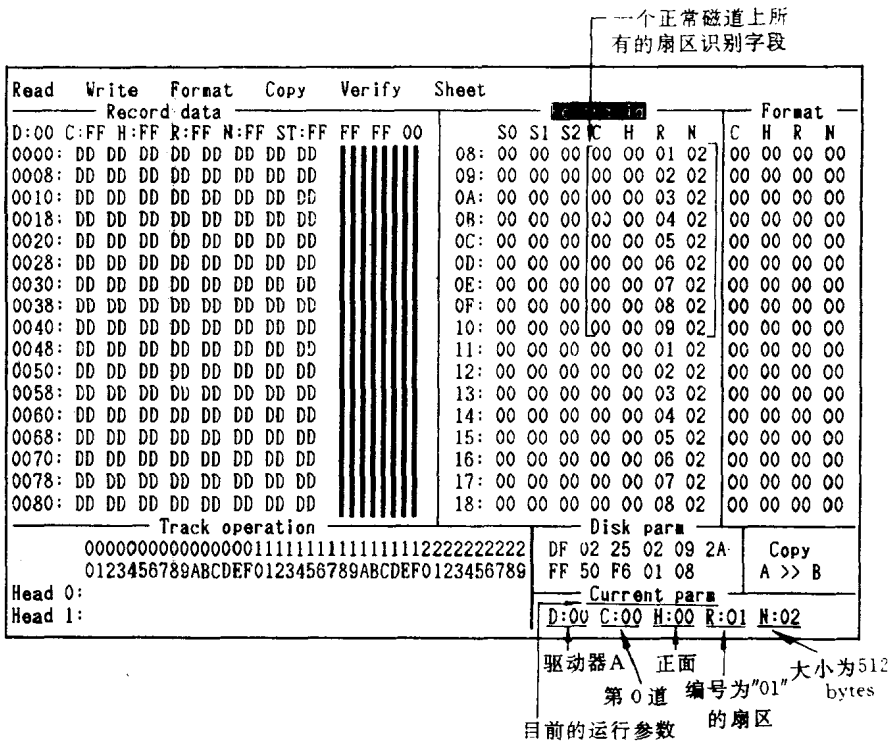


图 1-1-2.SCR

由右下方的 Current parm（意思为目前运行的参数）可知，所读的识别字段是从驱动器 A（由 D：00 代表驱动器 A）第 0 磁道（由 C：00 代表第 0 道）正面（由 H：00 代表正面）所读出来的。由 Record id 可观察到一整道所有扇区之识别字段（注意：循环出现同样的数据是重复的，我们只要观察一组即可。另外，S0，S1，S2 三个参数为驱动器传回之状态，读者可不予理会）其 C，H，R，N 之值均符合我们所讲的“正常状况下之格式化”的情况。

再读另一条磁道试试看：（见图 1-1-3.SCR）所读磁道为驱动器 A，第 10H 磁道，背面之所有扇区识别字段，我们亦可发现一切均很正常。

现在，利用 Verity 的功能去查验磁片上的各个磁道是否均为“正常”，它的结果将显示于 Track operation 的地方（见图 1-1-4.SCR）。

若磁道正常，则出现“.”，若不正常，则出现反白的“C”。我们见到该结果中，发现第 28H 磁道及第 29H 磁道（即第 40，41 两道）为不正常，这是怎么回事呢？原来这是因为 DOS 在 format（格式化）时只 format 了 40 条磁道（即第 0～39 号磁道，或第 00H～27H 磁道），第 40 及 41 号道根本不会出现在 DOS 的使用中。因为 DOS 未去 format 它，故在 Verify 时会出现错误。

或许读者会很惊讶，一张磁片不是只可 format 40 条磁道吗？怎么又多出了二条呢？

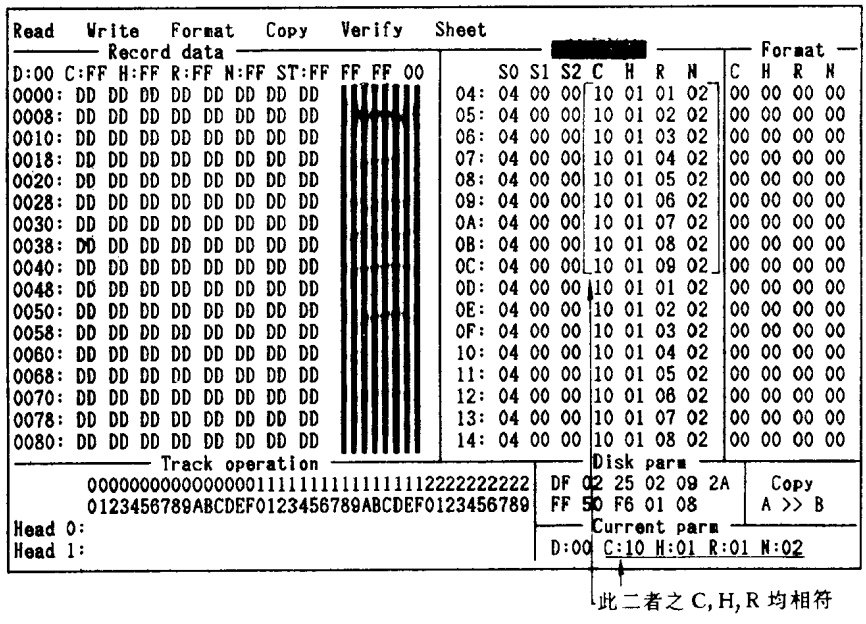


图 1-1-3. SCR

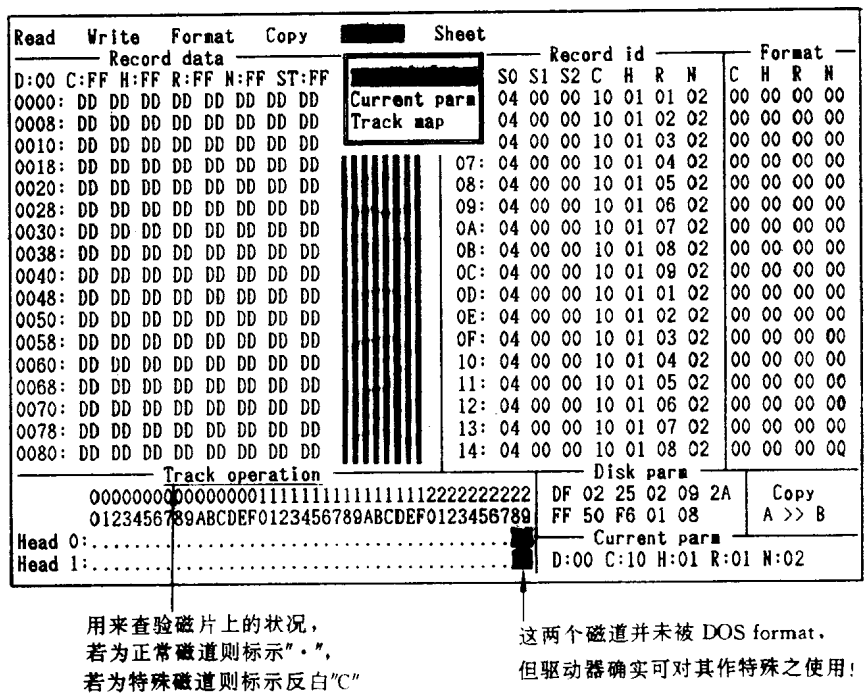


图 1-1-4. SCR

不错，在正常情况下确实是 40 条磁道，但是实际上驱动器可读写至 42 条磁道。由于多出来的这两条磁道一般人都不容易留意，加上 DOS 的操作又不理会这二道，故在这两条磁道上经常出现保护的花样。

前面已经看过了正常格式化的磁片之情形，现在我们再来看一张经过特殊保护的磁片，瞧瞧它究竟"怪"成什么情况！

首先，先用 Verify 功能来查验磁道，看看 Track operation 的报告如何（见图 1-1-5. SCR）。咦？怎么会有一堆错误？其中必定有问题！

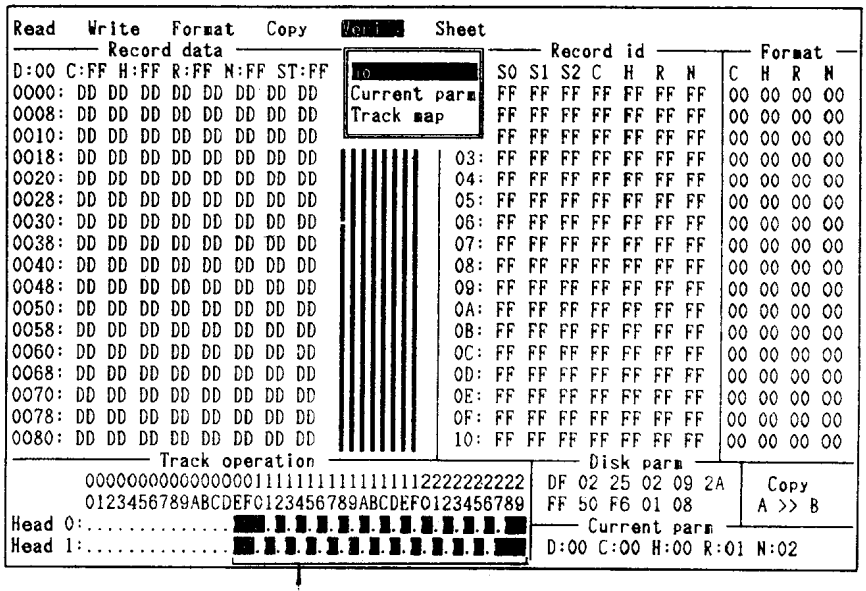


图 1-1-5. SCR

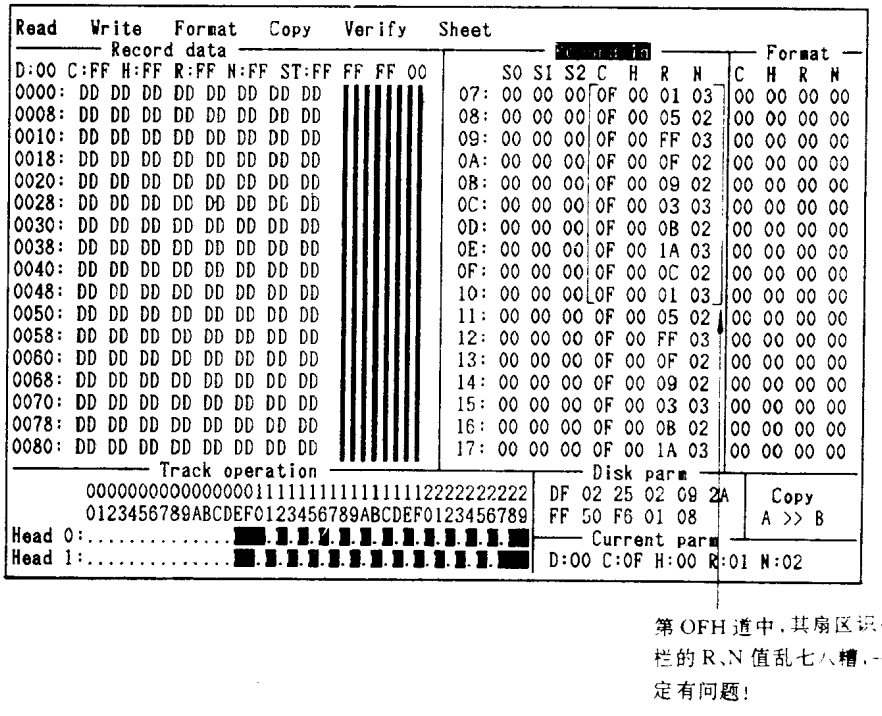
随便取一条磁道来瞧瞧，看看它的扇区识别字段究竟如何！由图 1-1-6. SCR 的 Record id 中看到，在正面第 0FH 号磁道中它的扇区识别字段非常地怪异，其 R 值乱七八糟，且 N 值也不对劲，怪？！怪？！怪？！……

再看看另一条磁道的情形又是如何，由图 1-1-7. SCR 可看到正面第 0EH 磁道的识别字段之 C、H、R、N 值。其 C、H、R 值均非常正常，但 N 值却不大对劲，居然每个扇区之 N 值均为 6。

按照公式(1)之算法，当 N＝6 时，每个扇区可供使用者存放 $128\times2^6=8192$ bytes 的数据，如此 9 个扇区应可存放 $9\times8192=73728$ bytes 的数据，这岂不是太夸张了吗？哪有这么大的容量？其中必定有诈！

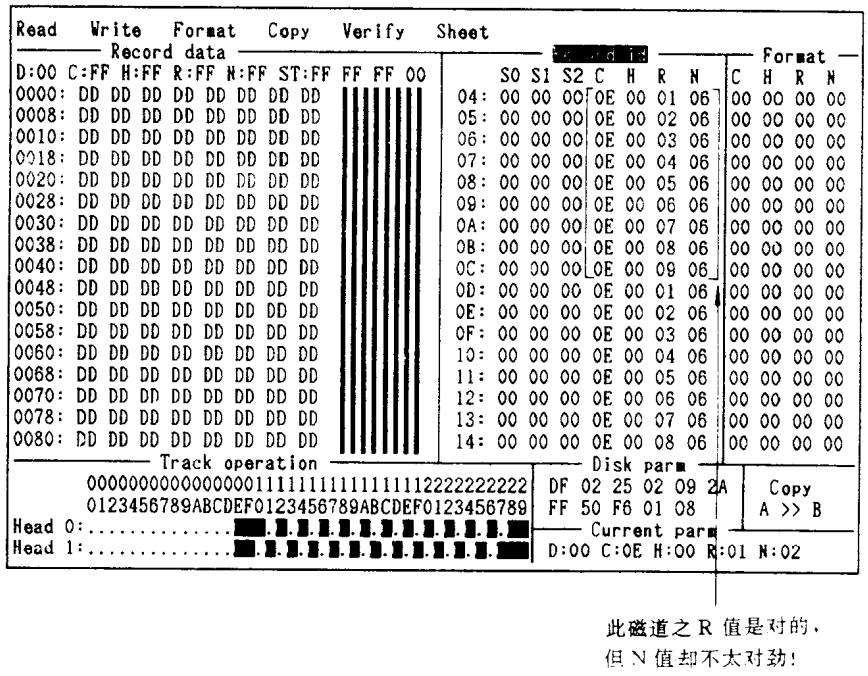
再看另一扇区：背面第 25H 号磁道（即第 37 道）之情形如何（见图 1-1-8. SCR）。

咦！居然读不到识别字段（由图中 C、H、R、N 之值均为 FFH 可做为判断），这更怪



第 OFH 道中，其扇区识别栏的 R、N 值乱七八糟，一定有问题！

图 1-1-6. SCR



此磁道之 R 值是对的，但 N 值却不太对劲！

图 1-1-7. SCR

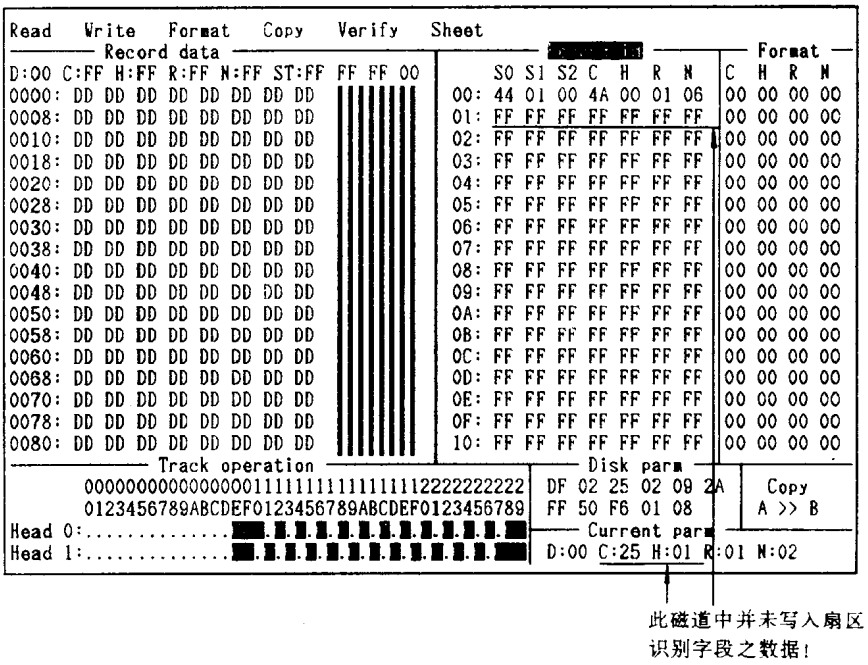


图 1-1-8. SCR

了.磁片必须先经过 format 之后才可使用,而 format 之后就必定有扇区识别字段.但为何却读不到了？？？……

看到这么多奇奇怪怪的磁道,你说这种磁片是否能被拷贝？纵使不谈拷贝,就算要弄懂它的道理都不容易了,更何况拷贝。不过别急,在下面的章节里将有详细的说明,并教您如何"做"出这些奇形怪状的磁道。

接下来,我们再来说明数据字段的部分。

数据字段大致又可分为二部分:前导同步信号及数据存放处。引导同步信号是为了让驱动器找到该扇区所在的位置后(由识别字段中的 C、H、R、N 之值作为识别)到磁盘真正开始读写的一段缓冲距离。到了数据存放处时,才是真正存放使用者的数据。一般而言,在正常情形下可存放 512 bytes 的数据,当驱动器在一个扇区写下使用者的数据时,它会由扇区识别字段之 N 值作为依据,若 N 值为 1,则驱动器在此部分(数据存放处)最多只能读写 256 bytes,若 N 值为 2,则在此部分最多只读写 512 bytes,余此类推。

正常的 format 时驱动器之动作如下:

在识别字段之后紧接着写下一段引导同步信号,再来写下 512 bytes 的 F6H。因此,一张经过 DOS 正常的 format 之后所有的用户数据均为 F6H。

实验与观察

再取出原来经过 DOS 正常格式化的磁片,利用 Read 的功能读取第 05H 号磁道第一扇区的用户数据,(见图 1-1-9. SCR)由 Record data 中可发现,所有的数据均为 F6H,此为正常现象。

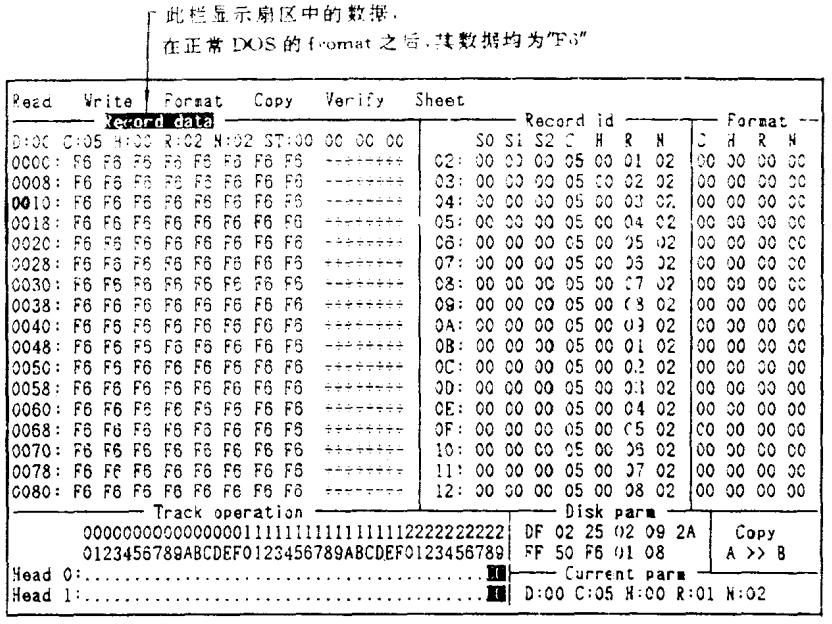


图 1-1-9. SCR

我们再拿一张经过保护的磁片做观察(以 GP 大赛车为例)，先用 Verify 查验所有的磁道是否有问题！果然，在正面第 27H 道有问题(第 28H、29H 道为空道，没有用到)，再利用 Read id 将扇区识别字段读出来瞧瞧，咦？一条磁道竟有 11 个扇区，且有二个很不寻常，其 R 值分别为 F1H 及 DEH(见图 1-1-10. SCR)。将 R 值为 F1H 之扇区内容读出来瞧瞧(见图 1-1-10. SCR 及图 1-1-11. SCR)，嘿！果真有问题。因为在正常情况下应是 F6H，但此处却为 4EH，且它连下一个扇区的识别字段都读出来了，像这一类的保护在一般的拷贝程序几乎都束手无策，更甭说是一般人了。或许，要做出这样的磁道大概得用拷贝机吧！

接下来再谈谈扇区的最后一个部分：缓冲区。

这个缓冲区存在于每一扇区的结尾部分，在作格式化时，由一个特定的参数(下一节会提到)来指定大小，正常情况下为 80 bytes，且内容全为 4EH。它的作用主要是用来作驱动器写入使用者数据之缓冲，因为每台驱动器的转速均不太一样，有的较快，有的较慢，若一张磁片由一台转速较慢的驱动器作 format，用另一台转速较快的驱动器作写入用户之数据，假如无此缓冲区，结果势必所写入的范围会超过 format 时所给予的数据存放处的范围，此将造成下一扇区之识别字段被覆盖，如此下一扇区就无法正常运作。

当我们在做数据的读写时，亦须给定另一参数，用以指明此缓冲区之大小。注意，此参数与 format 时用到的参数并不一样，通常此参数会小于 format 时所用的参数，此参数在读写数据时会使驱动器将它当作实际的缓冲区大小。因为在 format 时所给予的缓冲区较大，而在读写数据时认为缓冲区较小，这是一种安全的做法，如此才可确保下一个扇区之识别字段不受破坏。这个参数在正常情形下为 2AH，在下一节还会提到。

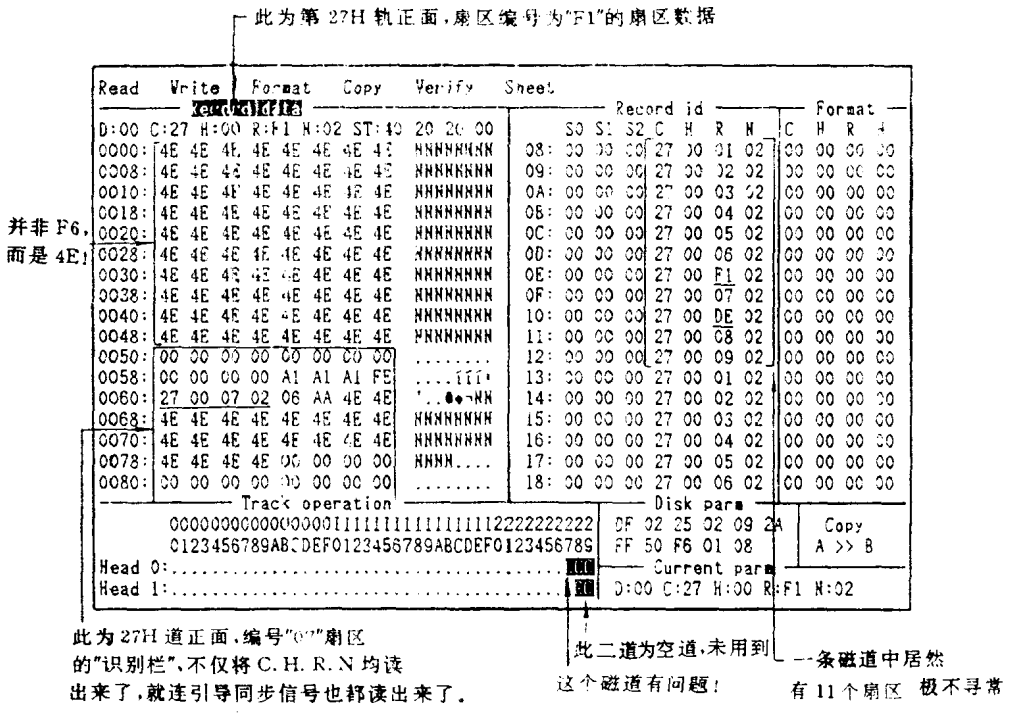


图 1-1-10.SCR

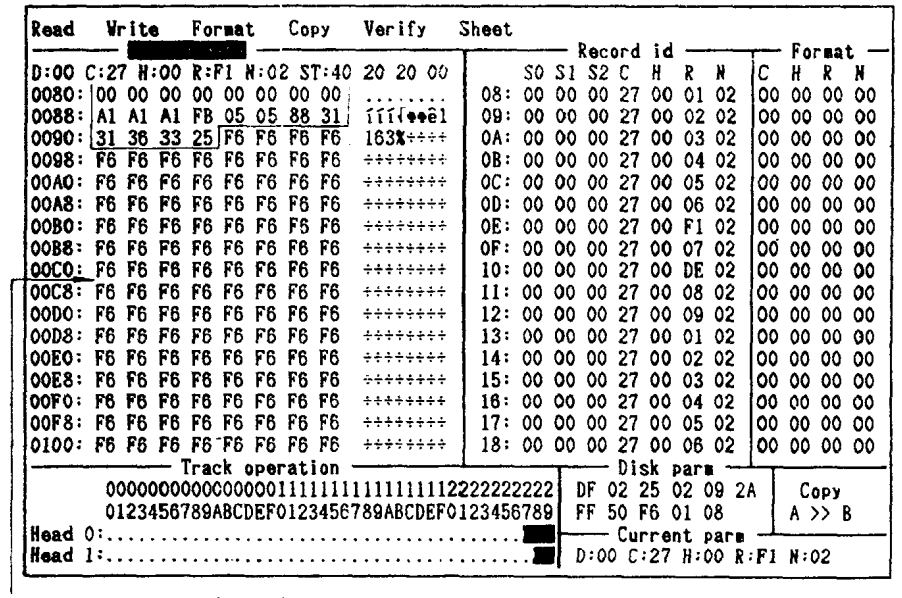


图 1-1-11.SCR