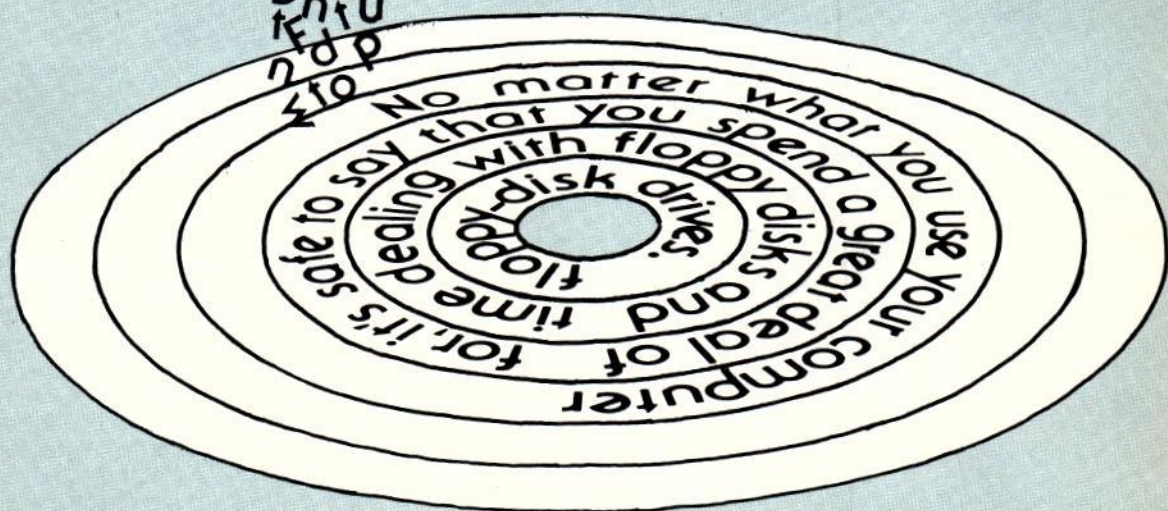


FLOPPY-DISK DATA STORAGE

Learn all about Apple and IBM disk formatting—including copy protection!



ROBERT GROSSBLATT

No matter what you use your computer for, it's safe to say that you spend a great deal of time dealing with floppy disks and floppy-disk drives. Loading programs and saving data are such common operations that we tend to forget how fragile the whole system is. But all it takes is one disk disaster to remind us of that fragility.

Of course, there are ways of protecting against those types of disasters, and other ways of dealing with them when they do occur. Performing regular backups is the best protection, but even that is not fail-safe. What happens if a disk crashes during a backup procedure?

In order to have any chance at all of recovering that data, as well as to back up copy-protected software, you need to know how data is stored on your disks. The more of the process you understand, the better your chances of successfully recovering a crashed disk. So in this article we'll examine how data is stored on both IBM and Apple floppy disks. The information provided will put you far on the road toward being a real "disk jockey."

Tracks and sectors

The standard 5¼-inch floppy disk consists of a disk of magnetically coated plastic that is contained in a jacket, as shown in Fig. 1-a. In order for your computer to use the disk, it must have a way of finding its way around the magnetic coating on the surface. It does so by treating the disk as a group of tracks that are divided into sectors. As shown in Fig. 1-b, the tracks are a series of concentric circles, each of which is divided into a number of segments, the sectors. In addition to tracks and sectors, disks also have two sides, as shown in Fig. 2. Not all disk

control hardware and software can use both sides, however.

The number of tracks and sectors determines how much data will fit on the disk. That amount is dependent on your computer's hardware and disk operating system (DOS). The numbers vary among computers and disk sizes, but the basic principles of operation are the same.

When you tell your computer to format a disk, the hardware moves the read/write head to track zero, the outermost track, and then forces it to deposit information on the surface of the disk that indicates the sector locations. The process is repeated for each track until the last track has been formatted.

Standard 5¼-inch Apple disks have 35 tracks on one side of the disk only, and the most common IBM format has 40 tracks on each side of the disk. Double-sided 3½-inch and 8-inch disks have 77 tracks per side, and the AT's quad-density 5¼-inch disks have 80 tracks on each side.

DOS (IBM or Apple) uses tracks and sectors to organize the disk's surface. At the DOS level, to find a particular piece of information, all you need are two pieces of information: track and sector numbers. With double-sided disks, you must also specify the head number.

The number of tracks per disk is usually a function of the hardware. The DOS talks to the disk controller, which, in turn, talks to the stepper motor in the drive and tells it to move the head in or out the desired number of tracks.

The number of sectors, however, is controlled by the DOS. IBM's DOS, for example, can format for eight or nine sectors per track, but standard Apple disks have sixteen sectors per track. So you can have more small sectors or fewer large sectors.

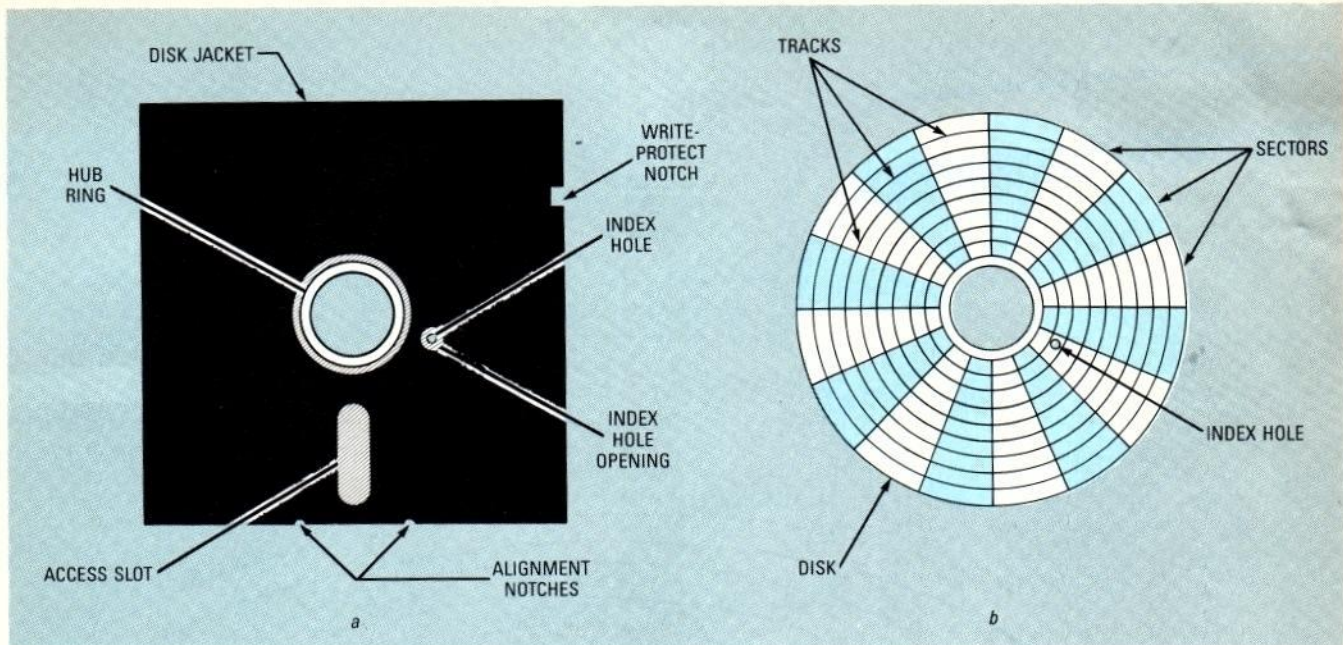


FIG. 1—DISK CONSTRUCTION: The magnetically coated disk is contained in a jacket (a), and is formatted to contain tracks and sectors (b).

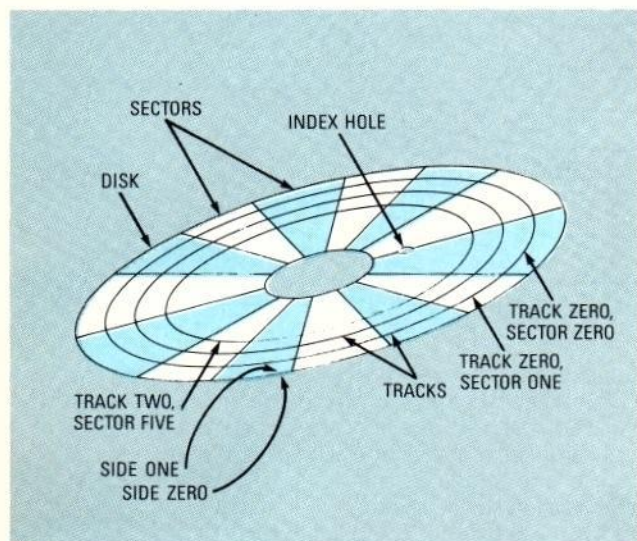


FIG. 2—BOTH SIDES of a disk are used by some disk control hardware and software.

Disk formatting

When a track is formatted, DOS writes three kinds of information in each sector: ID bytes, sync bytes, and gap bytes. The exact format of those bytes differs from computer to computer, but the same sort of scheme is used by every DOS. The reason is that DOS must have a way of determining exactly which sector it's looking at. Not only that, but there must be a way of ensuring that the special formatting bytes are never overwritten by data. If that does happen, DOS has no way to identify the sector and the result is what you might expect—a crashed disk.

There are actually two kinds of ID bytes on a sector—one is the signpost that marks the sector's location on the disk, and the other lets DOS know that it's looking at the beginning of the data stored in the sector.

Figure 3-a shows a dump of an Apple DOS 3.3 sector, and Fig. 3-b shows a dump from an IBM DOS 3.1 sector. At first glance, they both look meaningless—clearly different but equally meaningless. Those disk formats are the two most popular, and both the hardware and the software used to create them are

totally incompatible. It's even more interesting, therefore, to see that they use similar schemes to write disk data.

The ID marks on the IBM sector are written in hex on the disk; you'll find them in Fig. 3-b at offset 00A1h. The first three bytes (00, 00, and 01) show that you're looking at track 0, side 0, sector 1. The next byte (02) shows that the sector can hold 512 bytes.

Other sector sizes can be accommodated, as shown in Table 1. Normally, a maximum of about 6000 bytes can be written per track, so the final entry in the table may seem questionable. On the other hand, perhaps IBM has something up its sleeve.

The two bytes following the ID bytes contain a special error-detecting code called a CRC (cyclic redundancy check). The CRC is used by DOS to make sure that data read from the disk is correct. If the CRC calculated from the data that is read from the disk doesn't match the four CRC bytes in the header, DOS considers the data corrupt. Every time you change the data in a sector, DOS recalculates the CRC and writes it to the disk.

In order to keep those bytes from being overwritten accidentally, DOS uses sync bytes to mark the location of the ID bytes. When the floppy-disk controller writes a data byte on the disk, it sends out a steady clocked stream of ones and zeros. The Apple, for example, writes bytes to the disk at intervals of 32 microseconds. Sync bytes, however, are written at a different interval so they're easy to spot on the disk. Apple sync bytes are written in 40-microsecond intervals, and each sync "byte" is 10 bits long.

IBM sync bytes differ. The IBM sector in Fig. 3-b shows that there are three bytes containing a value of A1 beginning at offset 9D. Those are the specially written sync bytes that the floppy-disk controller uses to mark the location of the ID bytes. The twelve 00 bytes preceding the A1 bytes are also sync bytes. You can understand how they're used by tracing through the mechanics of a normal disk read.

When an IBM controller must read data, the first thing it does is make sure that it's looking at the right sector. It starts reading data, watching for a stream of 00 sync bytes, which lets it know that there's a chance that A1 sync bytes will follow. If they do, DOS knows that the following bytes are ID bytes.

Although ID bytes are used to mark both the signposts and your data, DOS can tell the difference by looking at the byte immediately following the A1 bytes. If it's an FE, the ID bytes are signposts, but if it's an FB then it's data. The amount of data is

```

Using drive B          Track 06h (6)      Side B      View data      Mode APPLE
Track Start: 0000      Track End: 1097      Track Length = 1090      Index: 1097

0000 == FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF *
0001 FF FF FF D5 AA 96 FE FE AB AE AE AF FA FF DE AA
0020 B4 00 E7 F9 FE FF FF D5 AA AD 96 96 96 96 96 96
0030 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
0040 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
0050 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
0060 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
0070 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
0080 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
0090 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
00A0 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
00B0 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
00C0 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
00D0 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
00E0 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
00F0 96 96 96 96 96 96 96 96 96 96 96 96 96 96 96
Pointer = 0000

F1-help 2-drive 3/4-data/clock 5/6-start/end 7/B-set start/end 9/edit 10-quit

```

```

Using drive B          Track 06h (6)      Side B      View data      Mode IBM
Track Start: 0000      Track End: 105B      Track Length = 105C      Index: 105B

0000 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E
0010 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E
0020 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E
0030 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E
0040 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E
0050 00 00 00 00 00 00 00 00 00 00 00 C2 C2 C2 FC 4E
0060 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E
0070 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E
0080 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E
0090 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E
00A0 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E
00B0 FE 00 00 01 02 C0 6F 4E 4E 4E 4E 4E 4E 4E 4E
00C0 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E
00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00E0 49 4E 4D 20 20 23 2E 33 00 02 02 01 00 02 70 00
00F0 00 02 FD 02 00 00 00 02 00 00 00 00 00 00 00
0100 00 0F 00 00 00 00 01 00 FA 33 00 0E 00 0C 00 7C
Pointer = 0000

F1-help 2-drive 3/4-data/clock 5/6-start/end 7/B-set start/end 9/edit 10-quit

```

FIG. 3—SECTOR DUMP of an Apple disk (a) and an IBM disk (b).

known because the sector size is specified in the signpost. The last non-data byte on the disk is called a gap byte. Gap bytes are insurance against worst-case operation. They're needed because not all disk drives turn at the same speed, so there's no way to guarantee that writing a new block of data to a sector won't overwrite existing ID and sync bytes. A disk drive only has one head per surface, so there's no way to read and write simultaneously. As long as drive speed is within tolerance, the DOS standards have been set so that there's no possibility of destroying any of the critical bytes needed to read the sector. On an IBM disk, the gap bytes usually have a value of 4E. Apple, on the other hand, uses 10-bit FF "bytes."

As for data bytes, if the sector hasn't been used, it will be filled with the DOS formatting bytes: IBM uses F6, Apple uses 96, and CP/M uses E5.

Although Apple's disk format is structurally similar to IBM's, the details are different because Apple's disk control hardware and software are unique. Most disk controllers store data in un-encoded format, so that a dump of an ASCII text file, for example, will be comprehensible.

The Apple hardware, however, limits the values that can be stored on disk. The high bit of each byte must be set, there can't be more than two adjacent zero bits, and at least two adjacent bits must be set in each byte. Some values are reserved for use as ID bytes, and hardware restrictions eliminate many others, so there are only 64 possible values that can be written to the disk to represent your data.

So, in order to be able to write all 256 combinations of eight bits, it's clear that the data must be encoded. In fact, Apple has gone through three major revisions of their encoding scheme. However, that's not a subject that can be covered here; see the books in the References sidebar for more information.

TABLE 1—IBM SECTOR SIZE ENCODING

ID Byte	Bytes/Sector
\$00	128
\$ 01	256
\$02	512
\$03	1024
\$04	2048
\$05	4096
\$06	8192

TABLE 2—APPLE DISK ENCODING

One Byte Volume Number	= B7 B6 B5 B4 B3 B2 B1 B0
Two Byte Disk Encoding	= \$FF FE
	= 11 11 11 11 11 11 11 10
Apple's Encoded Format	= 1B7 1B5 1B3 1B1 1B6 1B4 1B2 1B0
Decoded Binary Number	= 1 1 1 1 1 1 1 0
Volume Number	= \$FE
	= 254
One Byte Track Number	= B7 B6 B5 B4 B3 B2 B1 B0
Two Byte Disk Encoding	= \$AB AE
	= 10 10 10 11 10 10 11 10
Apple's Encoded Format	= 1B7 1B5 1B3 1B1 1B6 1B4 1B2 1B0
Decoded Binary Number	= 0 0 0 0 0 1 1 0
Track Number	= \$06
	= 6
One Byte Sector Number	= B7 B6 B5 B4 B3 B2 B1 B0
Two Byte Disk Encoding	= \$AE AF
	= 10 10 11 10 10 10 11 11
Apple's Encoded Format	= 1B7 1B5 1B3 1B1 1B6 1B4 1B2 1B0
Decoded Binary Number	= 0 0 0 0 1 1 0 1
Sector Number	= \$0D
	= 13

Apple format

Apple's sector format is somewhat different. Referring back to Fig. 3-a, the signpost ID bytes are located at offset 0013h after the series of FF sync bytes. The signpost bytes always begin with a prologue (D5 AA 96), which serves the same purpose as the FE marking the IBM signpost. The next eight bytes are encoded versions of the disk volume number, track, sector, and checksum. As shown in Table 2, by decoding them we see we're looking at volume 254, track 6, sector 13. The checksum is calculated by sequentially XORing all data bytes in that sector together.

Following that information is an epilogue, which can be seen beginning at offset 001Eh. It marks the end of the signpost area and has no counterpart on an IBM disk. The epilogue is there so that DOS can make sure it's been reading the correct signpost marks and that it is still in sync with the disk. They're not really necessary, but remember that the Apple system was devised in the late seventies when disk drives were not as reliable as they are today.

Following another group of sync bytes comes the data bytes. At offset 0028h is the prologue (D5 AA AD); then follow 342 bytes of data. Apple stores 256 bytes of data in each sector, but, because the data is encoded, 342 bytes are needed to do it. A checksum is calculated for the data and stored at the end of the data space along with the epilogue (DE AA EB).

Sector numbering

Although sectors are numbered sequentially, often they're not stored sequentially. When DOS looks for a particular sector, it must locate the signpost markers, read them, verify the read, and then see if they're the ones it was looking for. All that takes time, but meanwhile—the disk keeps spinning, so there's a good chance the next sector will have passed beneath the read/write head while the previous sector was being analyzed.

FLOPPY-DISK DATA STORAGE

continued from page 94

illegitimate software floating around, so it seems that something has to be done about basic human nature before copy protection will cease to be an issue.

Even though there are obvious (and subtle) differences between the hardware and software comprising various types of computers, the basic approach to copy protection is the same: Make the disk unreadable by the standard DOS. It's easy to do because any DOS must make a number of assumptions about disk format before it tries to read or write information. It must assume, for example, that it's going to find tracks formatted in a particular way, that each one will contain a specific number of sectors, and that those sectors will contain data written in a predefined fashion. If any of those conditions aren't met, DOS will throw in the towel, and, instead of data, all you'll get is an error message. The point is that any disk that has data organized in a non-standard way must also have a non-standard way to read that data.

When Apple introduced its disk system in the late seventies, the company emphasized software rather than hardware. That was a departure from the norm, because most disk systems were and are built around a single-IC LSI controller. As a result, Apple disks were (and still are) unreadable by most other machines. However, Central Point Software's Option Board allows an IBM to read Apple disks, and many others as well.

Doing most of the disk control in software makes it simple to

upgrade DOS. It also makes it easy for creative programmers to write copy-protection schemes that do strange things with the disk. That dependence on software, as we'll see, has produced methods of copy protection that are unique to the Apple.

Non-standard data formats


There are many methods of storing data in a non-standard format; we'll examine several in what follows. The most popular methods are these:

- Oddball track formatting
- Nibble counting
- Modified DOS
- Non-standard sectoring
- Unique data encryption
- Synchronized tracks
- Quarter tracks
- Spiral tracking

Of course, there are variations on those methods, and they're often used in combination. But attaining a good understanding of them will help you unravel any copy-protection scheme likely to come your way.

Those methods of copy protection are used on the Apple; due to differences in the IBM's disk-control hardware, it has fewer means of copy-protecting a disk. For example, the IBM cannot do quarter tracking. The most popular methods are:

- Oddball formatting
- Weak bits
- Laser burning

We'll examine those and other means of copy-protecting software next time. 

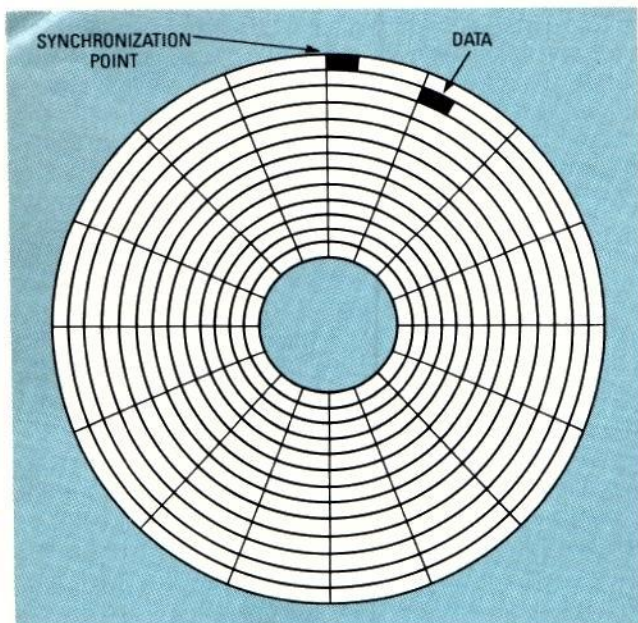


FIG. 5—TRACK SYNCHRONIZATION involves writing data at a particular location, stepping the read/write head, and writing additional data.

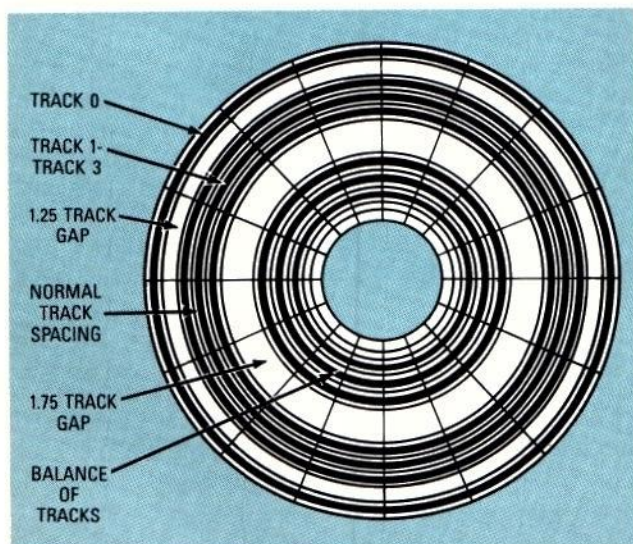


FIG. 6—NON-STANDARD TRACK WIDTH is achieved by stepping the read/write head in ¼-track increments.

outside of the disk (track 3, for example), the rest of the disk would be protected.

Another early method was to move the catalog from track 11, where it was normally found. The files were also made unlistable by changing the standard load locations so you couldn't even look at the files. A reset would wipe out memory and reboot.

Those tricks, and a few others, were attempts to protect programs by altering DOS. They were effective for awhile because no one had taken DOS apart yet, so DOS parameter locations, sector formats, and file structure weren't common knowledge. After those things became known, however, the simple copy-protection schemes were dropped in favor of more-sophisticated ones.

The 36th track

When the copy-protection industry was in its infancy, someone discovered that, although DOS was designed

around 35-track hardware, most Apple drives could actually read 36 tracks. That was the first time the disk hardware itself was used as the basis of a copy-protection scheme. Publishers put part of their code on track 36 and checked to make sure it was there whenever the program was run. None of the standard copy programs knew anything about the extra track so the disk was uncopyable—until the method was found out and publicized. Even without the discovery, use of the 36th track was ineffective because not all drives could read that track. So, as soon as consumers started returning software, the method was dropped.

Track syncing

The next type of copy protection can be understood by looking at Fig. 5. When data is either read from or written to a disk, DOS is told to go to a specific track and then look at a particular sector; no special relationship is assumed between adjacent tracks. However, software publishers discovered that it was possible to keep very strict timing relationships when reading the disk. This meant that if you knew where you were on one track at the instant you told the head to step, you knew where you would be when you arrived at the next track.

Building a protection scheme around that fact involves writing a program that reads a track, steps the head when a particular data pattern is found, and then immediately writes some data to the new track. You now have a disk with known data patterns written in a particular order on adjacent tracks. If you change the write to a read and don't find the data pattern you originally wrote there, you know that it's not the original disk.

Playing With Apple DOS

The ways in which you can "customize" DOS are limited only by your imagination. After all, DOS is just another program. Here are a few suggestions on how you can change the internal workings of DOS to provide a measure of copy protection to your own disks.

Normally DOS reads your keyboard commands and tries to execute them. However, by patching in your own routine, you can cause DOS to do just about anything. Location \$9FED is a good patch point. In an unmodified DOS, you'll see the following code:

```
9FED 59 A4 A8 EOR $A884,Y
```

That is the beginning of the code that parses the input line before going into the command table. Here are several ways to patch that code. First, by causing a jump to \$C600, any input line will cause the machine to reboot:

```
9FED 4C00 C6 JMP $C600
```

This line will cause any input line to jump to BASIC:

```
9FED 4C 03 EO JMP $E003
```

This line will cause any input line to beep and go into the monitor:

```
9FED 4C 65 FF JMP $FF65
```

This line will cause any input line to beep and print "ERR":

```
9FED 4C 2D FF JMP $FF2D
```

DOS has both warm and cold boot routines, and it can tell which one is required by looking at the byte stored at location \$03F4. If you change the value in DOS, you'll force a cold boot whenever the reset switch is pressed. Normally DOS has the following code:

```
9E36 49 A5 EOR #$A5
```

By changing the A5 to a 00, you'll be able to initialize a disk with a DOS that will reboot whenever reset is pressed.

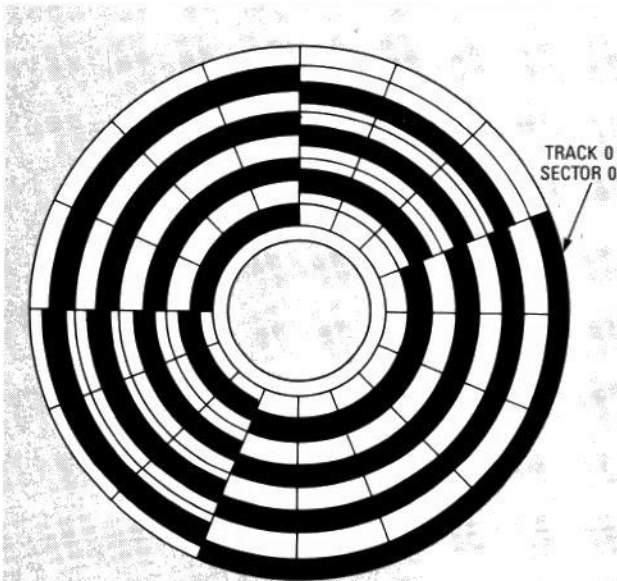


FIG. 7—A "SPIRAL" TRACK organization is achieved by offsetting alternate tracks or groups of tracks.

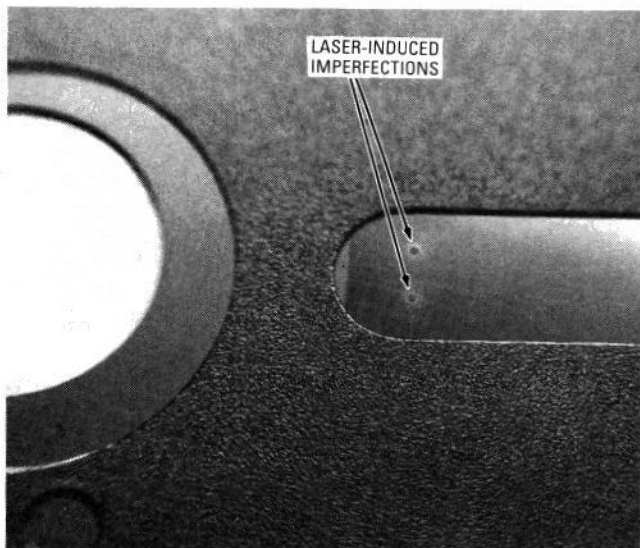


FIG. 8—A LASER-BURNED DISK can be backed up, reformatted, and the copies restored—and the copy-protection is still intact.

Using synchronized tracks became extremely popular. Publishers liked it because it was easy to implement and, at the time, none of the existing copy programs could get around it. Of course, that didn't last long. Most current copy programs can write synchronized copies, so that tracks on the copied disk are arranged in the same order as those on the original disk.

Quarter tracking

All of the copy-protection methods we've discussed so far are techniques that can be implemented from a regular DOS. Apple made their disk system very software intensive, so programmers have much control over the disk hardware. As more and more became known about DOS, programmers found new and sometimes bizarre ways to make their disks unreadable by normal methods.

Some unknown hero in the copy-protection business discovered that Apple DOS didn't actually move the

read/write head an entire track at a time, only a quarter of a track at a time. Being able to step between tracks seems useless because data must be at least one track apart for it to be read reliably by the computer. The problem is crosstalk—exactly the same sort of problem that crops up in audio and video tape. If the guard band is too small, or if the recorder's heads are out of alignment, the head can read from two tracks at the same time.

The secret lies in the fact that, although tracks must be separated by at least a whole track, they can also be separated by more than that. A disk using that method is shown in Fig. 6. Track zero is in the correct position, but there's a track and a quarter between it and track one. The next three tracks are one track apart, but then we have a gap that's one and three quarters of a track wide. The remainder of the tracks are separated by one whole track.

It doesn't take much to see that some of the data on the disk is going to be inaccessible to a normal DOS. It will be able to read track zero and the tracks from five to the end of the disk. But when DOS tries to read the odd-spaced tracks, the head won't be positioned over the center of the track, so the signal will be weak. Signal-to-noise problems guarantee that, although some of the data might be read correctly, a good part of it won't. The result is a disk unreadable by any DOS that doesn't know exactly how each track is positioned on the disk.

Writing quarter tracks on a disk requires careful attention to timing details. The stepping rate of the head must be carefully controlled, as does choosing the moment at which data can be read. Many disk drives, particularly older ones with slower stepping rates, have trouble reading a disk with quarter tracks. The problem is more pronounced with tracks that are written near the perimeter of the disk, because the disk turns with a slightly faster linear velocity there.

Spiral tracking

Next, the software industry developed the idea of spiral tracks, which solved the problem of wasted disk space, and made it even more difficult to make copies. Figure 7 is a representation of a disk with spiral tracks. You

LISTING 1

```

10 REM *****
15 REM * THIS PROGRAM WILL LEFT *
20 REM * YOU MOVE THE LOCATION *
25 REM * THAT DOS USES FOR THE *
30 REM * CATALOG TRACK. *
35 REM *****
40 :
50 OLDTRACK = PEEK (44033)
90 TEXT : HOME
100 PRINT "THE CATALOG IS NOW ON TRACK ";OLDTRACK
110 PRINT
120 INPUT "WHAT IS THE NEW CATALOG TRACK? ";YOURTRACK
125 VTAB 6: HTAB 1
130 IF YOURTRACK<3 OR YOURTRACK>35 THEN PRINT CHR$(7): GOTO 500
140 :
145 REM PATCH THE FILE MANAGER
150 POKE 44033,YOURTRACK
160 :
165 REM PATCH THE INIT ROUTINE
170 POKE 44703,YOURTRACK: POKE 44764,YOURTRACK
180 :
185 REM PATCH THE CATALOG ROUTINE
190 POKE 46012,YOURTRACK
200 :
210 PRINT "YOU CAN NOW INITIALIZE A DISK THAT HAS"
215 PRINT "FILES UNREADABLE BY A NORMAL DOS."
220 END
500 INVERSE : VTAB 24: HTAB 1
510 PRINT "THE NEW TRACK MUST BE BETWEEN 3 AND 35";: GOTO 125

```


LISTING 2

```

59E4    CF E7 59    DEC $59E7
59E7    CF          ???
59E8    EA          NOP
59E9    59 EF EA     EOR $EAFF,Y
59EC    59 AD 51     EOR $51AD,Y
59FF    C0 AD        CPY #$AD
59F1    54          ???
59F2    C0 AD        CPY #$AD
59F4    57          ???
59F5    C0 AD        CPY #$AD
59F7    52          ???
59F8    C0 20       CPY #$20
59FA    60          RTS
59FB    5B          ???
59FC    20 C5 5B     JSR $5BC5
59FF    20 4E 5B     JSR $5B4E
5A02    A9 04        LDA #$04
5A04    8D EC B7     STA $B7EC
5A07    A9 00        LDA #$00
5A09    8D EB B7     STA $B7EB
5A0C    A9 00        LDA #$00
5A0E    8D F0 B7     STA $B7F0
5A11    A9 60        LDA #$60
5A13    8D F1 B7     STA $B7F1
5A16    A9 40        LDA #$40
5A18    20 45 5A     JSR $5A45
5A1B    10 01        BPL $5A1E
5A1D    A9 20        LDA #$20
5A1F    91 5A        STA ($5A),Y
5A21    AD 50 C0     LDA $C050
5A24    A9 09        LDA #$09

```

• • •

can see that it meets the track-spacing requirement and that it makes maximum use of disk space. Trying to copy a disk like that can result in major brain damage. Not only do you have to know the track spacing, but you also must have the correct pattern.

There are two reasons why spiral tracks are a real problem for a copy program. The first is simply that it's hard to tell how many sectors have been placed on a particular track. The second more serious problem has to do with the nature of copy programs. We've already seen that there are so many ways to protect a disk that a good copy program can't make any assumptions about what it's going to find when it reads the disk. The more it expects to find, the less it will be able to deal with what's really there.

Let's suppose that you've just bought Acme Copy, the roughest, toughest, smartest, copy program in the world—it's so good it can even copy an unformatted disk—and you use it to make a copy of a spiral-tracked disk. You load the program and turn it loose. Even though

track zero is written upside down and backwards, Acme Copy copies it without a hitch. Let's also suppose that the rest of the disk is spiral tracked; only half of the sectors on each track contain real data. Acme Copy doesn't know anything about that—no assumptions, remember?

Acme Copy reads in a track, half of which is data and half of which is garbage—and that's where the problem comes in. There is no way for the program to distinguish garbage from copy-protected data. It reads the track's data, does some sort of analysis, writes out the copy, and steps to the next track. Of course the copy will be worthless. Even if Acme Copy goes through the disk quarter track by quarter track, the act of writing a quarter track will undoubtedly corrupt the previous quarter track. It's sad but true that the only way you can get it to work properly is to tell it what spiral pattern to follow as it goes through the disk.

IBM copy-protection

Most of the Apple protection methods we've looked at have their counterpart in the IBM world. Modifying DOS and messing around with sector information were done early on in PC history—and neither method lasted any longer there than in the Apple world. Most IBM copy programs can deal with those methods without even working up a sweat.

However, some of the more imaginative copy-protection methods found on the Apple simply couldn't be ported over to the IBM because of the basic difference in their disk systems. The PC's designers decided to let most of the disk system be handled by an LSI controller IC. That made it simpler to develop DOS, because the controller has built-in routines to handle disk primitives like moving the head, reading and writing data, formatting, and so on. The ability to do quarter tracking, for example, is impossible because the controller hardware can only step the read/write head in full-track increments. In fact, because of the limited repertoire of commands built into the floppy-disk controller IC, just about the only trick that appeared had to do with the index hole.

Contrary to popular belief, the IBM only uses the index hole when formatting a disk. Every time the head is stepped out to a new track, the PC waits for the index hole to appear and uses that point as the starting point for formatting the track. After a disk has been formatted, particular tracks and sectors are located using the same method as the Apple. The floppy controller reads in the sector address of its current position and then steps in or out to the track DOS wants it to read.

Some protection methods want the track-splice point (the place where start and end points meet) to be exactly at the index mark. As with any protection method, however, this one only baffled copy programs for awhile. After the method was uncovered, it wasn't long before most copy programs could handle it.

Undocumented op-codes

IBM's floppy controller is an NEC PD765, which is really a microprocessor that has been optimized to handle disk drives. Some programmers disassembled the microcode in the IC looking for features and abilities that didn't appear in the documentation.

Several undocumented features were found, the most popular of which was to mix FM (frequency modulation)

and MFM (modified frequency modulation) formatting on one track. None of the copy programs were able to handle that mixed formatting, because they didn't know how to make the PD765 do the trick. The result was a nearly unbeatable protection scheme.

However, there are two big problems with undocumented op-codes. The first is that the manufacturers who second-source the IC don't know about them. The second is that, because they aren't part of the 765's published vocabulary, there's no way to guarantee they'll still be there when new versions of the IC are released. And that's what spelled the death of mixed formatting on the IBM. The software ran well on computers that used the same run of 765's, but died on other machines. Needless to say, the scheme was dropped.

Other methods

The search is always on to discover new and wonderful ways to lock up disks. The older, software-only methods such as altering DOS or playing around with sector formatting and address bytes, are still used because they're inexpensive and easy to do. They're usually found on games and other low-priced software. Even though most copy programs know how to deal with them, some are still hard to beat, particularly nibble counting.

The publishers of more expensive software, however, have deep enough pockets to be able to afford more expensive protection schemes. There are two major high-end methods for protecting disks. Both are expensive because it takes more than just changing a few bytes to get them on a disk. The first involves what are called "weak bits," sectors written in such a way that they don't read the same way twice. All that's needed to activate weak-bit protection is to do two successive reads. If they're the same, the software knows it's running on a copy and can take appropriate action. Getting a sector like that on a disk involves the use of special duplicating equipment; it can't be done with a stock PD765.

Just as weak bits make the disk unique from a formatting point of view, laser holes make it unique from a physical point of view. The word *hole* is misleading because the disk isn't actually punctured, but burned, usually in two different places. The photograph in Fig. 8 shows what to look for if you suspect that the disk you're trying to copy has that kind of protection. The marks are usually located on the back side of the disk near the hub. It's easy for the software to check for the laser holes.

Laser-treated disks are probably the most expensive form of copy protection, but they have one big advantage for a publisher. Because the disk is physically unique, the publisher doesn't have to protect the files and can let you back them up on a regular disk. If you develop an error on the original disk, you can reformat it and copy the files from your backup. Remember that reformatting the disk has no effect on the laser holes. However, most copy programs can even get around laser-treated disks.

Because of the ease of overcoming most disk-based schemes, state-of-the-art copy protection these days is the hardware lock. It's a device that plugs into your serial or parallel port and remains totally transparent (in theory, at least) to the normal operation of the port. Software can check whether the device is present at various times during execution, and come to a screeching halt if it doesn't get the proper response.

References

Books (Apple): *Beneath Apple DOS*, Don Worth & Pieter Lechner, Quality Software, Computer Book Division, 21601 Marilla Street, Chatsworth, CA 91311, (818) 709-1721. *Beneath Apple ProDOS*, Don Worth & Pieter Lechner, Quality Software (address above).

Books (IBM): *The Programmer's Guide to the IBM PC*, Peter Norton, Microsoft Press, Dept. RC06, Box 97200, 10700 Northup Way, Bellevue, WA 98009. *Advanced MS-DOS*, Ray Duncan, Microsoft Press (address above).

Hardware (IBM only): The Copy II PC Option Board, Central Point Software, 9700 SW Capitol Highway, Suite 100, Portland, OR 97219, (503) 244-5782.

Software (Apple): *Locksmith 6.0*, Alpha Logic Business Systems, 4119 North Union Road, Woodstock IL, 60098, (815) 568-5166. *Copy II Plus*, Central Point Software, (address above). *Bag of Tricks 2*, Quality Software (address above). *Disk Repair Kit*, Penguin Software, 830 4th Avenue, P.O. Box 311, Geneva, IL 60134, (312) 232-1984.

Software (IBM): *Copy II PC*, Central Point Software (address above). *Master Key*, Sharpe Systems, Corp., 2320 E. Street, La Verne, CA 91750, (714) 596-0070. *CopyWrite*, Quaid Software Limited, 45 Charles Street East, Department 740, Third Floor, Toronto, Ontario M4Y 1S2, (416) 961-8243.

However, hardware locks are very expensive, so it's unlikely you'll see one protecting inexpensive software. Can a hardware lock be beaten? The answer is yes—sort of. The qualifier is there because, no matter what kind of protection scheme is employed, it only protects the disk, not the data. All the fancy tricks that have been used to lock up the software fall away when the program is loaded into the computer.

Your own copy protection

Several of the books mentioned in the References sidebar contain complete discussions and disassemblies of various versions of Apple DOS. By studying that information, you'll see that there are several ways to alter DOS and add simple copy protection to your own disks. DOS commands can be changed (or eliminated altogether), or your own code can be inserted in one of the unused areas of DOS. Then, when your program goes looking for it, if the code is there, the program will run normally. If it's not there—well, the choice is yours. You could be kind and just reboot the system, or you could be nasty and trash a couple of tracks.

If you're interested in playing with a modified disk organization, initialize a spare disk and run the program shown in Listing 1. It will create a diskette with the catalog located in a non-standard location.

To create a custom version of DOS, you can play with the ideas shown in the sidebar entitled *Playing With DOS*. However, just remember that those are fairly simple schemes, and they'll be no real obstacle to someone whose primary mission in life is to get a look at your code. The kinds of things you can do by messing around with DOS are the things a real "crack-ist" eats for lunch.

continued on page 102

FLOPPY-DISK DATA STORAGE

continued from page 94

Breaking copy protection

Copy protection—both making it and breaking it—is big business. There are two fundamental ways to get around copy protection. The first is to buy a program that knows how to copy protected disks. If you're lucky, it will know how to make sense of the particular protection scheme(s) used on your disk and will be able to copy it with no muss and no fuss. But the price you pay for that kind of mindless copying is that, if the program can't copy the disk, there isn't a thing you can do except try some other copy program.

The second method is to use some disk tools to snoop through the disk and remove the copy protection yourself. As with most things, each method has advantages and disadvantages. The method you choose depends on how good your tools are, how well you can use them, how much you know about your computer, how much time you want to spend, and how badly you want to make the copy.


There's simply not space to go into the details of how to break copy protection; it's an art in itself. Basically, it involves spending endless hours at a totally unnatural act: staring at and trying to decipher page after page of undocumented object code. Unless you've actually done it, there just aren't any words to describe the amount of work involved.

One reason is that the code you'll be looking at probably was written to be confusing. The code in Listing 2 is a perfect example. Go through it and see whether you can

understand what's going on. It's real code from a popular Apple game. There are no tricks here, either. It's just that things aren't what they seem to be. The real meaning (and the real code) is hidden. If you figure it out, drop us a note. And if there's enough response, we'll take up the subject of copy breaking in another article.

If you want to learn about the in's and out's of copy protection, you'll have to get familiar with the normal workings of the standard DOS used by your computer. Read the books mentioned in the References sidebar, and follow their tutorials. Those books won't tell you how to break copy protection, but they'll give you the basic tools you need to do so.

If you just want to back up your copy-protected (IBM) software, try the Option Board from Central Point Software. It's the ultimate tool for dealing with disks on a bits-and-bytes level, and it also helps copy "un-copyable" software. All the screen dumps in this article were produced using the Option Board, which also reads disks formatted on just about any computer, including IBM, CP/M, and even the Apple! That's a major accomplishment—especially for a \$100 piece of hardware.

You'll also find a list of some good copy programs in the References sidebar; you should have some of them in your library even if you don't need to copy protected disks. Whenever you get a floppy-disk data error, chances are what has happened is that at least one sector was written incorrectly. All it takes is one bad bit in a header and the sector will be unreadable by DOS. If the damage is in the directory, you'll be unable to access any of the data and will be faced with the thankless job of trying to reconstruct your files. The point is that many programs capable of dealing with protected disks can deal with damaged disks as well. 

XT TO AT UPGRADE

I recently upgraded my computer from an XT to an AT clone, and there seems to be some sort of problem with the disk drives. Whenever I try to read a disk from the AT on my XT, I get one of two kinds of errors. The most common one is that lots of read errors show up, but occasionally I can't read the disk at all. I can't even get a directory to show up on the screen. What's going on?—F. Scher, Amsterdam, NJ

You haven't given me all the particulars of your computers, but I can make a good guess as to the source of the problem. The chances are that you got your AT with a 1.2-MB 5¼ inch drive and your XT has a 360K drive. The two drives look very much alike on the outside, but there's a big difference internally. In order to understand what's causing your first problem, let's talk a bit about the basic difference between the drives.

The original 360K floppies have two sides with 40 tracks each, and each track has nine sectors. The 1.2-MB disks were organized a bit differently to get the increased amount of storage. The high-density disks have 80 tracks on two sides, and each track is divided into 15 sectors. Since you've got twice as many tracks and 60% more sectors, you can store more data on the disk. If you do the arithmetic, you'll see that the numbers work out correctly.

It makes sense that something had to be done to the original drives to allow them to hold so much more information. And it's what was done to the drives that's causing both of your problems.

Disk drives are essentially the same as tape recorders. They have a read/write head, and they record information on magnetic media (the disk surface). When the number of tracks was doubled, the distance between tracks was halved (makes sense), and doing that increased the chances of crosstalk between the tracks.

The problem was solved by reducing the write current on high-density drives. Since the signal was much lower, the unwanted noise from nearby tracks was reduced. In order to read the desired tracks, however, the read gain was also increased. The system worked well (and still does), but it was necessary to change the composition of the recording medium in order to make the system reliable. There's a real, physical difference between 360K and 1.2-MB disks, and each can only be used for its intended purpose.

If you want to use a 1.2-MB drive to write to a 360K disk, you have to use a disk made for 360K operation. Both the number of tracks and the number of sectors can be changed in software. When you issue the command `FORMAT A:/4`, you're telling the software to make the head put forty tracks and nine sectors on the disk—you'll be formatting a 360K floppy disk.

What's causing your problem is that while the software can force the drive to do the correct number of tracks and sectors, it can't do anything about the write current—that's an internal adjustment on the drive and the software can't do a thing about it.

When you write a 360K disk on a high-density drive, the information is going to be correctly organized on the disk but the recorded level will be very low. Since the 360K drive has its read gain set for a higher recorded level, the drive often has trouble reading the disk and that's the first problem you're having.

The second problem you're having—not being able to read the disk at all—is probably because you're trying to read a 1.2-MB floppy in the 360K drive. That can't be done at all.

The solution to your problems is through hardware, and the cheapest way to do it is to add a 360K drive to the AT. Adding a 1.2-MB to your XT will undoubtedly mean you'd need a new disk controller as well, and there's no reason to spend the extra cash.

WRITE- PROTECT NOTCH BY-PASS

*You paid for both sides of your disks.
Here's how to use both sides.*

NOEL NYMAN

■ If you own a single-sided floppy disk drive, you may have read that the opposite side of your diskettes can also be used to store data and programs. During manufacturing, all disks are tested for data recording integrity on both sides. Those not meeting manufacturer's standards on one side are packaged as single-sided disks.

Using the uncertified "backside" of disks isn't recommended for valuable data or for disks that will be read frequently. When you flip a disk over, the cleaning material inside the jacket may release particles of dust and oxide to the disk surface and corrupt your read/write head. Dual-sided drive owners don't have this problem: their disks turn in one direction only. However, many computer owners use this technique for archival or back-up disks which are read infrequently.

To write on a disk, the write-protect notch must be uncovered. On a single-sided disk, there is no write-protect notch for the back. Special punches are available that will cut a neat, square notch. Most users prefer to use a conductor's punch or a scissors. Using any of those methods may damage the disk jacket or warp the disk itself.

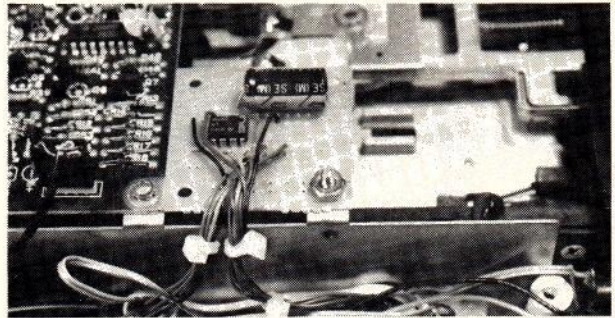
How it works

Here's how to modify your drive to electronically bypass the write-protect circuitry. We'll use the Commodore 1541 disk drive in our example, but the same idea should be adaptable to Atari drives or any other disk drive that doesn't use the small timing hole near the disk's center.

In most disk drives, the write-protect notch is sensed optically. An LED is mounted opposite a phototransistor with the write-protect notch lined up between them when the disk is inserted. If the notch is uncovered, the light from the LED causes the phototransistor to conduct.

On the Commodore 1541, this brings the write-protect line low (ground potential or near zero volts) and signals the drive circuitry that the disk can be written to.

If a write-protect tab is in place, or there is no notch on the jacket, the light path is blocked and the



SMALL CIRCUIT BOARD with IC, switch and resistor all in place illustrates the simplicity of this circuit. It allows you to write to both sides of the disk with no need for punching holes.

transistor does not conduct. This leaves the write-protect line high on the 1541 and the drive will not write to the disk.

To bypass the circuit, hold the write-protect line low by shunting the phototransistor with a resistor. This is easy in most drives since the phototransistor is mounted on the drive mechanism and the leads from it plug into the circuit board. No changes are required on the circuit board itself.

Be careful!

You may want to wait until the warranty expires before attempting any modification. If possible, obtain a schematic of your drive from a dealer or repair service. The drive circuits use CMOS chips which can be damaged by improper handling. Use normal CMOS precautions when working around the circuit board.

First unplug all cables, then remove the top cover from the Commodore 1541 by loosening the four mounting screws accessed through holes in the bottom cover. Remove the metal shield that covers the circuit board. Two screws on the left side secure the shield.

Look for the largest plug, labelled "P6" on most boards. It is a 15-pin plug but only a few wires are connected. Counting from the back of the drive, locate pins 12 and 13. These are the wires coming from the

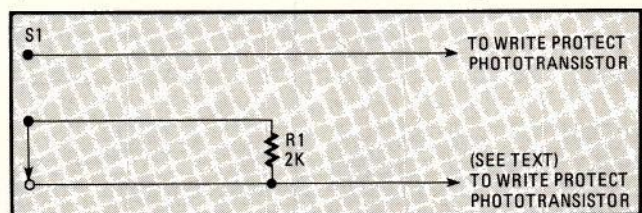


FIG. 1—IN ITS SIMPLEST FORM, the schematic above uses only a switch and resistor.

phototransistor.

To make sure you have the right wires, carefully bare the insulation near the plug and connect a voltmeter or logic probe to them. Pin 13 is the negative or ground side. Plug in the power cord and turn on the drive. Be careful not to touch the circuit board while the power cord is connected. The voltmeter should read near zero volts.

Put a disk part-way into the drive so the write-

protect area is blocked. The voltage should increase to almost three volts, a TTL logic one or high. If you get these readings, you have the proper wires.

Figure 1 is a diagram for installing a switch and resistor to bypass the phototransistor. A 2K resistor (R1) worked on the drives we tested, but you may have to try values between 1K and 2K to get reliable operation. Do not simply short the two wires together, as this might damage the phototransistor or other circuit components. If you mount the resistor directly to the switch, no separate circuit board or stand-offs will be required to hold it.

Additional circuitry

Although this simple modification will allow you to write to the uncertified side of the disk without punching notches in it, we recommend the circuit shown in Figure 2. This will flash the green "Power On" LED whenever the write-protect bypass switch is turned on.

We used the LM3909 (IC1) because it provides a bright LED flash at low voltage. This lets us use the 2 volts available at the green LED's plug directly with no

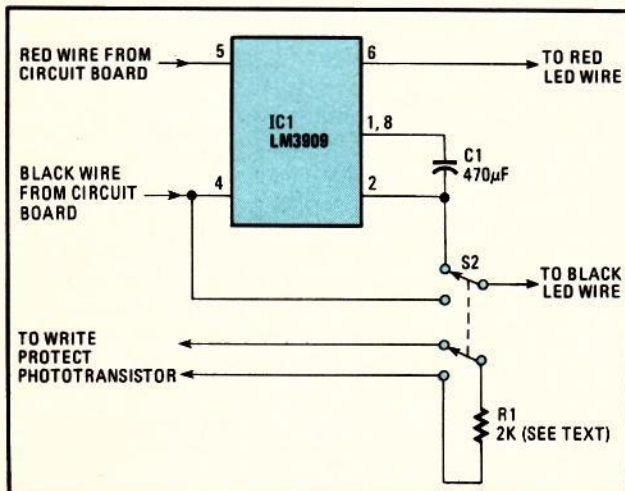


FIG. 2—MORE ELABORATE CIRCUIT is still not complicated, but accomplishes a great deal more. Resistor R1 might have to be changed. See text.

changes on the drive circuit board. Cut the red and black wires going to the green LED and connect them as shown in the schematic. You may want to use plugs and sockets to connect the circuit to the drive so you can remove it if you need to have your drive serviced.

Any double-pole, double-throw switch will work, but select one that will fit into the case past the drive chassis before you start punching holes. The switch we used is a miniature toggle that can be located almost anywhere. A slide switch might have been more compact, but would have required additional holes. The circuit board we used fits nicely in front of the "short" circuit board used in the newer 1541 drives and can be bolted to the unused circuit board mounting tab.

Once the switch and LED flasher are in place, test by trying to SAVE a program to a disk with a covered write-protect notch. With the switch in the on position,

the green LED should flash and the program will SAVE to the disk.

Avoid confusion

You should turn the switch on only when you SAVE to or format a disk with no notch. If the switch is left on, your drive can get very confused and give you strange errors. To illustrate this, turn the write-protect switch off, put a disk with an uncovered notch in the drive, and type the following in direct mode (Commodore only):

```
OPEN 2,8,2,"X,S,W"
```

This tells the drive that we're about to write information to a sequential file we've called "X." The red LED should come on and stay on, indicating that a data channel is open to the drive. Now remove the disk from the drive. The red LED will go out. The drive "knows" that you've removed the disk and that the data

PARTS LIST

- IC1—LM3909 LED Flasher
- R1—2000 ohm, 1/4 watt resistor
- C1—470µF Electrolytic Capacitor
- S1—SPDT Toggle Switch
- S2—DPDT Toggle Switch
- Circuit board, plugs and mounting hardware

channel shouldn't be held open.

Type: CLOSE2

To get rid of the open file in the computer, then try the same experiment with the write-protect switch on. This time, the red LED does *not* go out! The disk drive uses the high-to-low transition of the write-protect line as the back of the disk crosses the light path to tell that you've removed a disk. With the write-protect switch on, this line is held low and the drive doesn't see any change. If you change disks in this way, you will have difficulty LOADING files on the first try. More important, if you SAVE to the second disk, you may overwrite important data or programs because the drive will use the Block Availability Map of the previous disk.

Properly used, the write-protect switch will give you access to the back of your disks without the need for expensive punches or danger of damage. It also gives you a measure of security since there's no telltale notch to indicate that anything has been recorded on the back.

Using the electronic circuits shown here, you can write to the back of the disk at your own volition; you'll find this a great convenience if you haven't had this facility before. It effectively doubles the capacity of your disks.

However, it's always a good idea to mark or number your disks so you'll know which disks are written on both sides, and what information is contained on the backs. A separate sheet or ledger can be maintained as a menu so you can quickly and easily locate the information you require at any given time. You might also want to carefully clip one corner of the disk envelope so you can easily tap out any collection of oxides and/or debris that might accumulate in the envelope and possibly foul your heads. ◀▶

ADDENDUM TO CARING FOR DISKETTES

In preparing for publication the article "How to Care for Diskettes," in the November 1978 issue, one page of the original manuscript was inadvertently omitted. As a result, some additional information is necessary to clear up some misconceptions that may have been created due to the omission.

In small diskette systems, the type most popular with computer hobbyists, the actual diskette rotates within a protective jacket. After the diskette is loaded and the loading door closed, the internal mechanical arrangement forces a pressure pad to "squeeze" the flexible diskette to the head. In a sense, this produces a "dimple" in the relatively soft diskette at the point of contact.

Depending on the diskette and drive used, the relative head-to-diskette speed can reach about 8 mph. Thus, if there are any scratches on the head or if any foreign substance gets on the diskette so that it is forced between the head and the soft diskette surface, minute physical grooves can be cut creating data dropout. Figure 2, shown for what is called a "flying head" disk system, dramatically illustrates how foreign matter on the surface can create data loss on the disk.

During diskette operation, the pressure pad on the other side of the diskette "scours" the surface. It is possible for the pad to accumulate a layer of relatively hard dust, or even minute (metal) oxide particles scraped from the diskette—in most cases, even though only one side of a diskette is used, both sides are coated with magnetic oxide.

After some hours of use, the tiny hard particles adhering to the pressure pad can scratch the diskette surface. If the other side of the diskette is to be used, the rotation is then "backwards," which can cause surface damage and result in loss of data. It is probably for this reason that no small diskette drives use both sides of the diskette.

It should have been stated at the beginning of the "Foreign Matter" section that the mechanical data was for a large disk system whose diameter and drive speed are much higher than those of a small diskette. Thus the rpm is much higher. However, the information on the damage that can be caused by foreign matter—including smoke particles, dust, and grease—holds true for all diskettes.—*Les Solomon, Technical Director.*

MICRO-FLOPPY

RETROFIT



Retrofit your PC or XT with a 3 1/2-inch disk drive.

HERB FRIEDMAN

If you use an IBM PC or clone, you may be underwhelmed by all the fuss being made about 3 1/2-inch disks. However, many portable computers, and all of IBM's new line of PC's, use 3 1/2-inch disks. (See "Editor's Workbench" for reviews of two of the new PC's.) The small diskettes have many advantages over the 5 1/4-inch disk you're used to using, including:

- Increased capacity (two to four times that of a standard 360K floppy disk)
- Greater reliability, because each disk is completely enclosed by a hard plastic shell
- Smaller, shirt-pocket size

5 1/4-inch disks are by no means obsolete, but chances are that the industry will move steadily toward use of 3 1/2-inch disks, just as 8-inch disks were gradually supplanted by 5 1/4-inch disks. So in this article we'll show you how to retrofit your computer to use 3 1/2-inch disks. Then you'll be ready to handle the upcoming new wave of software and data. We'll discuss installation of IBM's model 2683190 disk-drive retrofit kit for PC and XT model computers. Similar kits are available from clone manufacturers, but installation may differ, so your drive's instructions carefully.

What it is

The retrofit kit consists of a cabinet-mounted 3 1/2-inch disk drive with attached signal and power cables, a Y-adaptor that lets you tap

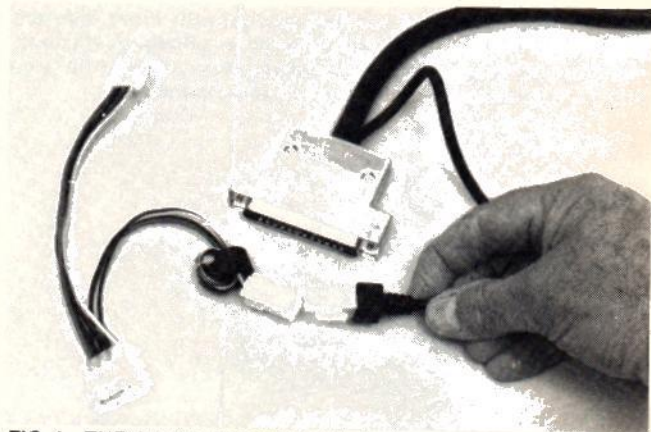


FIG. 1—THE CABLE FROM THE 3 1/2-inch drive has its own power connection take-off that matches the miniature power socket on the supplied Y-adaptor. The ring through which the adaptor's power wire loops is a toroid choke that help suppress RFI.

power from your computer's internal disk-drive power connector (shown in Fig. 1), and a kit of three pre-punched metal brackets

(shown in Fig. 2) that accept the Y-adapter's connector.

Installation is simple. First you mount the appropriate bracket on the rear apron of your computer. Then you install the Y-adapter in series with one of the existing internal disk-drive power connectors. Next, you push the small power connector through the hole in the bracket. That connector locks in position by means of mounting ears molded on the connector. Finally, you connect the cable from the 3½-inch disk drive to the controller card in your main computer.

With some PC's you won't need to install the power cable in series with the floppy power connector. The reason is that the power supplies in some PC's have four power connectors. So, if you haven't used all four, just connect the Y-adapter to one of the unused connectors.

Use the bracket that causes the least inconvenience. For example, if you use the relatively large standard rear-slot bracket shown in Fig. 2, you must give up an entire slot. Some PC's have only five slots, so it may prove impossible for you to use the large bracket. In that case you could use the smallest bracket, which will mount in the small hole above the cassette port (yes, the original PC included a cassette interface). The medium-size bracket can be used in the extra slot above the keyboard port on an XT.

Clone panel layouts may vary, so you might have to use the full-size bracket and give up a slot. Or you might just cut a hole of your own in which to mount the small bracket.

Standard controller

To use the adapter, you must have an IBM-type floppy-disk controller, the kind with a 37-pin D-connector on the floppy-disk bracket (as shown in Fig. 3), in addition to the regular floppy-disk

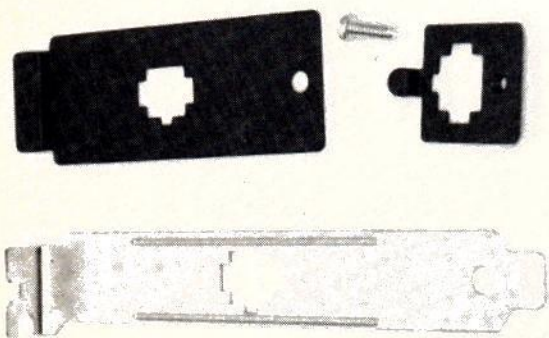


FIG. 2—THE RETROFIT KIT is supplied with three different brackets for the power sockets. Use the one that's most convenient for you, but remember that the standard bracket (the large one) may force you to give up use of one expansion slot.

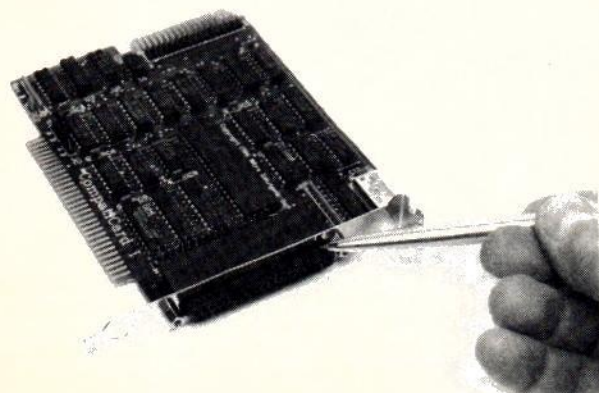


FIG. 3—IBM-TYPE DISK CONTROLLER cards have a 37-pin socket on the rear for external disk drives (C: and D:). The retrofit cable must connect to that socket.

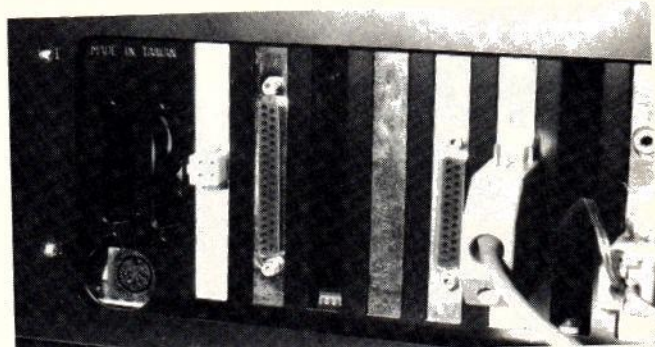


FIG. 4—THIS IS A TYPICAL CLONE INSTALLATION. The disk-controller socket and the 3½-inch drive's power socket are on adjacent brackets.

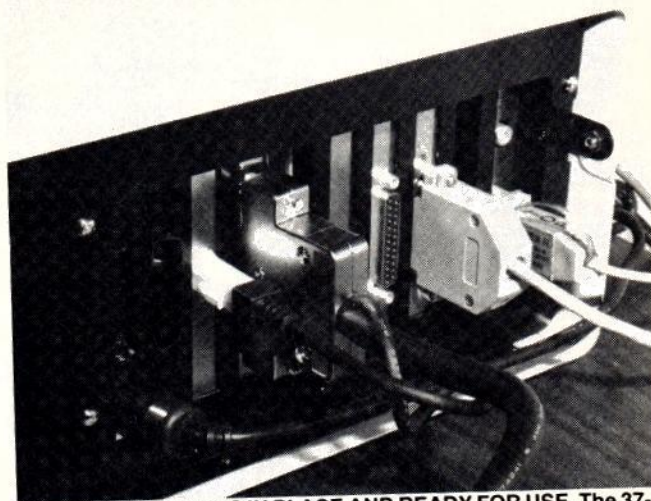


FIG. 5—CONNECTORS IN PLACE AND READY FOR USE. The 37-pin D-connector is a real heavyweight, so be certain that you tighten its mounting screws to ensure reliable operation.

connector. The IBM controller accommodates four floppy-disk drives: two internal and two external drives. Because the retrofit kit connects to the computer via the external 37-pin connector, you cannot use a multi-function disk controller (the kind that combines a disk controller, serial and parallel ports, a joy-stick interface, and a clock), because it has no connector for external floppy-disk drives. The controller itself needn't be an actual IBM device; having the external connector is the important point.

Figure 4 shows an XT clone ready to connect the 3½-inch disk drive. The external disk-drive connector is adjacent to the miniature power connector installed in the slot furthest left.

To install the 3½-inch drive, simply plug the appropriate connectors from the drive in the appropriate jacks, as shown in Fig. 5.

Device driver

Before you can use your new drive you must tell the computer that it's there by adding a device driver to your computer's CONFIG.SYS file, the configuration file that's automatically read when the computer boots. For example, adding the line:

```
DEVICE = DRIVER.SYS /D:2
```

to your CONFIG.SYS file will allow you to access a 3½ inch disk drive as the next available drive (D: on an XT). IBM's device driver comes only with DOS versions 3.20 and 3.30. (Some clone manufacturer's drives are available with drivers that work under DOS 2.11.—*Editor*) The device driver informs your computer that the 3½-inch drive exists, establishes its physical parameters, including number of tracks, sectors per track, number of heads, etc., and sets the drive's logical designation (D:, E:, F:, etc.).

Toshiba ND-354A with their "Universal Kit." The kit is very inexpensive and complete, with accessories for the PC, XT, AT, Compaq2, AT&T PC6300, and compatibles. Besides costing less than the IBM kit, the Toshiba kit has the advantage of not requiring the IBM's 37-pin D-connector.

My original configuration was one half high 5¼ FDD and one half-height 20-megabyte hard disk. The half height 3½-inch disk drive fits very nicely on top of my A: drive.

I am using PC DOS 3.2, and to format 3½ inch disks at 720K I used the undocumented DOS command called DRIVPARM in my CONFIG.SYS file. The complete command is DRIVEPARM=/D:1/F:2, where /D:1 is the drive number (B:) and /F:2 indicates the 3½-inch drive type.

Having a 3½ inch drive is necessary for me to maintain compatibility with the new IBM computers at work—and it is nice to have 720K floppy storage.

RICHARD F. PELLY
Huntington Beach, CA

MICRO-FLOPPY RETROFIT

Before I read the article, "Micro-floppy Retrofit," in the August 1987 **ComputerDigest**, I converted my XT B: drive to 3½ inch, using the



Floppy

rior to 1977, computers relied on tape drives to memorize and store data. With the introduction of the disk drive, home-based computing became more convenient, compact and faster.

Early 5 1/4 diskettes were single-sided and their limited storage capacity, even at 160K, was considered more than adequate for the home-user. Of course that mindset quickly changed and the upward memory spiral began in earnest, starting with the introduction of double-sided disks.

In 1984 Apple introduced its first generation Macintosh computer. It featured a 3 1/2" floppy drive, with a 400K capacity, and 128K RAM already on-board. From inception, the 3 1/2" disk drive was expected to quickly dominate the marketplace. The diskettes are sturdier, more compact and have a greater capacity. The micro-floppy, as it's called, has lived up to its promise, although versatility and variety seem to be guiding many PC buyers into having both 3 1/2" and 5 1/4" drives installed in their systems.

Alongside performance speed, computer memory (RAM, hard drive and floppies) has been on an upward, unremitting climb since the early days of personal computing. By 1984, IBM's AT (Advanced Technology) model boasted a 1.2 megabyte floppy drive (5 1/4"), eventually increased to 1.44mb (3 1/2"). In its turn, Apple launched the 1.44mb Superdrive. And last year, IBM bested this with a 2.8MB drive.

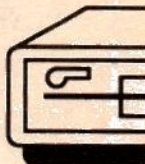
DISKS AS EXPENSIVE JEWELRY. Throughout this brief history, one thing hasn't changed: the vulnerability of diskettes, whatever their size or capacity, to damage and loss of data. If you value your data, the best general advice is to treat them like "expensive jewelry".

What that means in precise terms isn't always clear. Users may be unaware or unsure of what exactly will damage the diskette or how a disk became hurt or corrupted in the first place. So the questions, the mysteries, persist.

Here is a list of the major sources of disk problems, the causes of which may go undetected until it's too late:

- **THE IMAGE FADES OVER TIME.** This is regarded as the most common problem. Remedy: copy the file to another disk, "strengthening" the old data.
- **SURFACE DAMAGE.** The disk surface somehow becomes damaged, or a flake of magnetic material gets momentarily stuck to the write head. The result: information or data is incorrectly written to the disk.
- **THAT OLD MAGNETISM.** The disk may become demagnetized, leaving a magnetic image on the disk surface, rendering it unreadable. Remedy: keep your disks away from anything that is, or could be, magnetically-charged, ie. paper-clips.
- **CROWDED HOUSE.** The disk may become damaged due to overcrowding your collection of diskettes. Any severe pressure can cause data erasure.
- **CAN'T STAND THE HEAT.** Excessive temperature, especially heat, will damage the disk.
- **FINGERPRINTS ON THE MAGNETIC DISK.** This may cause the diskette to become unreadable;
- **VIRUS ALERT.** A computer virus or poorly-written software program. The disk is rendered temporarily unusable;
- **FRAGMENTATION.** This refers to the condition in which files are divided into pieces and scattered around the disk. Fragmentation occurs naturally through frequent use: creating, deleting and modifying files. It's undesirable because it slows the speed at which data can be accessed.

ILL-ADVISED FORMATS. Another problem worth extended treatment is in disk formatting, which prepares the disk (a storage medium) for reading and writing. When you Format a disk, the operating system (ie DOS) tests the disk



y Musings

History & Maintenance Tips

by Jack Singer

to make sure all sectors are reliable, marks bad sectors (those that are scratched) and creates internal "address tables" that it later uses to locate information. You must, of course, format a disk before using it: otherwise it's akin to a blank sheet of paper without lines or margins.

There's some debate about using high density diskettes in a low-density drive. (Bear in mind that latter-day computers have been designed to operate both high and low density diskettes). What about buying cheap 720K diskettes, punching a hole in them, and then formatting for high density, for example? Some people swear this is a neat trick that works. Others have found that these diskettes develop a short shelf-life of maybe a month or two. It may not be worth the risk.

WHEN CHEAP MEANS QUESTIONABLE. Never enough can be said about discouraging people from using cheapie diskettes. Admittedly, there's controversy here from those who claim the inexpensive bulk diskettes are quite adequate, adding that they always make back-ups anyhow, so that reduces the risk.

It's also true that buying 'brand names' doesn't always result in owning quality disks. If you're prepared to take the risk, or can reduce it substantially by regular back-ups, trial and error may be a safe solution. My own feeling is that I'd rather not take chances.

All diskettes utilize the same materials in their construction. The key difference between the low- and high-end disks (forgetting price differences) is in the kind and quantity of magnetic material used. Poor quality disks are especially flimsy.

With the 5 1/4s, there are differences in the sealing of the disk casing. Some brands are spot sealed; others are glued. Opinions vary as

to which one is the best sealing method, some preferring disks that are glued. I've checked several diskettes and found the more expensive diskettes just as likely to be spot sealed as the less expensive ones.

PREVENTION & UNDERSTANDING. All major diskette problems can be prevented and many corrected by special software, commercial and shareware. These products are not omnipotent. While they can move data from bad to good disk sectors or reformat without losing data, nothing can save a physically damaged disk. And you'd have to be a programmer to correct truly bad data or corrupted files, where the data integrity has been lost.

The best cure is prevention and understanding, and being able to manipulate, your operating system. □

TIPS TO PREVENT DISK DAMAGE

- ◆ Don't write on the diskette with a ballpoint—if anything, use a felt pen for labelling and write gently.
- ◆ Keep anything that's magnetic away from the disk. In the case of 5 1/4" diskettes, don't put paper clips on them to attach a little note. (Clips have been known to become magnetized and leave a magnetic image on the disk.)
- ◆ Keep your disks stored properly, avoid overcrowding your collection including excessive heat or cold.
- ◆ Don't handle the all-important mylar or PVC disk.
- ◆ Protect your software against computer viruses by using an up-to-date virus program, or two.