For The Serious User Of Apple][Computers

Hardcore COMPUTIST

Issue No. 21 $3.75

Softkey For Archon
Pg. 10

Demystifying The Quarter Track
Pg. 12

ProShadow: A ProDOS Disk Monitor
Pg. 16

Deprotecting Dazzle Draw
Pg. 24

Hardcore COMPUTIST
PO Box 110846-T
Tacoma, WA 98411

FOR THE SERIOUS APPLER

Announcing *Executor*, a collection of powerful new, interesting, educational and entertaining utilities. Several are combinations of exec, binary, and Applesoft to readily do things that might otherwise be very difficult.

Cracker adds the *Step*, *Trace*, and *Reset* features of the Old Monitor to your 64K II +, IIe, or IIc without changing a *Rom* chip. Excellent for debugging and analyzing binary files.

Tweezers lets you easily recover an intact *Dos 3.2 or 3.3* data sector no matter what damage has been done to the address field, prologue bytes, checksums, or epilogue bytes (some familiarity with disc formats required). Gets many "protected" discs with ease—slowly, one sector at a time.

Linefinder locates an Applesoft line number in memory and prints the location in hex and decimal in a fraction of a second.

Repair automatically recovers most Applesoft programs after FP and many other memory-glitching events.

Scan instantly locates any sequence of either hex or ASCII bytes in binary files.

Dosdasm disassembles *Dos 3.3* to your printer, with major subroutine headings and hexdumps for the data caches, while you have a cup of coffee.

The list goes on and on. Over 35 individual files, including one that prints out our handy pokes conversion chart **and source code** (*Toolkit* assembler) for the longer binary files. All copyable, listable, customizable.

$29.95 the copy, postpaid.

CRANE HILL
P.O. BOX 273 • GONZALEZ, FL 32560
904-623-1700

Requires 64K, printer, two drives.

WHO
LIKES
THE

HERE ARE JUST A FEW OF THE MANY

"an essential part of the Apple-user's repetoire" - APPLE USER

"a valuable buy . . . manual is practically worth having on its own" -WASHINGTON APPLE PI"

"the folks at Golden Delicious should be commended . . . worth waiting for" - HARDCORE COMPUTIST

"multifaceted" - NIBBLE

"the most comprehensive disk accessor I have ever come across" -A.B., VERNON, CANADA

"its ability to unlock other programs will greatly help me" - DR. B.P., SAN FRANCISCO, CALIFORNIA

"an excellent set of programs . . . just great - and good value too" -E.A.S., MILTON KEYNES, ENGLAND

"very, very educational . . . great manual . . . it is FANTASTIC!!!!!!!" - J.C., TUCSON, ARIZONA

"*a very enlightening piece of software/book . . . top of my list for good buys*" - H.S., *BLAINE MINNESOTA*

"I like yours the BEST" - R.R., CHICAGO

Why all the excitement about the CIA (confidential information advisors)? Probably because it is the ONLY set of utilities (5 in all) which enable even a beginner to investigate, edit, locate, list, trace, rescue, translate, patch, repair, verify, examine, protect, unprotect, analyse, encrypt, and decrypt programs on normal AND protected disks. You also get the "CIA Files", a 65000+ word book which contains detailed instructions for using the C.I.A. plus easy-to-follow, hand-holding tutorials about patching, repair, formatting, encoding, protection, and numerous other disk topics. You'll find plenty of material here which has never before appeared in print. PROGRAMS NOT COPY PROTECTED

To put the 5 C.I.A. utilities, plus book, on the trail of your Apple II+, IIe, & IIc disks, send $65.00 by check or money order to:

GOLDEN DELICIOUS SOFTWARE LTD.
350 Fifth Avenue, Suite 3308, Dept H, New York, New York 10118

Now-the ultimate back-up system!

EDD III® and TRAK STAR™

COMPLETE PACKAGE:
159⁹⁵*

Includes • TRAK STAR • 2-Drive Adaptor • EDD III • Trak Star Patching Software

PRICED SEPARATELY:

TRAK STAR 99⁹⁵

2-Drive Adapter (required for 2-drive systems) $12

Documentation: $3
Refundable with the
purchase of TRAK STAR

*Please add $3 for shipping & handling. Foreign airmail & nandling, add $8

Save copying time with nibble programs

- Works with nibble copy programs to display tracks and half tracks that the program accesses.
- Operates with any Apple®-compatible program.
- Save time by copying only the tracks being used.
- Displays up to 80 tracks and half-tracks; compatible with high density drives.
- If copied program doesn't run, Trak Star displays track to be recopied.
- Includes patching software for Trak Star.
- Compact size permits placement on top of disk drive.
- Does not use a slot in the Apple® computer.
- For Apple II, II +, IIe and compatibles.

Personal checks, M.O., VISA & Mastercard accepted.

Apple is a registered trademark of Apple Computer, Inc. EDD III is a trademark of Utilico Microware

Midwest Microsystems

To order, phone:
913 676-7242

9071 Metcalf / Suite 124
Overland Park, KS 66212

Many of the articles published in Hardcore COMPUTIST detail the removal of copy protection schemes from commercial disks or contain information on copy protection and backup methods in general. We also print bit copy parameters, tips for adventure games, advanced playing techniques (APT's) for arcade game fanatics and any other information which may be of use to the serious Apple user.

Hardcore COMPUTIST also contains a center CORE section which focuses on information not directly related to copy protection. Topics may include, but are not limited to: tutorials, hardware/software product reviews and application and utility programs.

What Is a Softkey Anyway? Softkey is a term which we coined to describe a procedure that removes, or at least circumvents, any copy protection on a particular disk. Once a softkey procedure has been performed, the resulting disk can usually be copied by the use of Apple's COPYA program (on the DOS 3.3 System Master Disk).

Commands and Controls: In any article appearing in Hardcore COMPUTIST, commands which a reader is required to perform are set apart from normal text by being indented and bold. An example is:

```
PR#6
```

Follow this with the RETURN key. The RETURN key must be pressed at the end of every such command unless otherwise specified.

Control characters are indicated by being boxed. An example is:

```
6⌃P
```

To complete this command, you must first type the number 6 and then place one finger on the CTRL key and one finger on the P key.

Requirements: Most of the programs and softkeys which appear in Hardcore COMPUTIST require one of the Apple][series of computers and at least on disk drive with DOS 3.3. Occasionally, some programs and procedures have special requirements. The prerequisites for deprotection techniques or programs will always be listed at the beginning of the article under the "Requirements:" heading.

Software Recommendations: The following programs (or similar ones) are strongly recommended for readers who wish to obtain the most benefit from our articles:

1) **Applesoft Program Editor** such as Global Program Line Editor (GPLE).
2) **Sector Editor** such as DiskEdit, ZAP from Bag of Tricks or Tricky Dick from The CIA.
3) **Disk Search Utility** such as The Inspector, The Tracer from The CIA or The CORE Disk Searcher.
4) **Assembler** such as the S-C Assembler or Merlin/Big Mac.
5) **Bit Copy Program** such as Copy][Plus, Locksmith or The Essential Data Duplicator
6) **Text Editor** capable of producing normal sequential text files such as Applewriter][, Magic Window][or Screenwriter][.

You will also find COPYA, FID and MUFFIN from the DOS 3.3 System Master Disk useful.

Super IOB: This program appeared in Hardcore COMPUTIST No. 9, No. 14 and The Best of Hardcore Computing. Several softkey procedures will make use of a Super IOB controller, a small program that must be keyed into the middle of Super IOB. The controller changes Super IOB so that it can copy different disks. It is recommended that you get the latest version of this program (only appearing in Hardcore COMPUTIST No. 14).

RESET Into The Monitor: Many softkey procedures require that the user be able to enter the Apple's system monitor during the execution of a copy protected program. Check the following list to see what hardware you will need to obtain this ability.

Apple][Plus - Apple //e - Apple compatibles: 1) Place an Integer BASIC ROM card in one of the Apple slots. 2) Use a non-maskable interrupt (NMI) card such as Replay or Wildcard.

Apple][Plus - Apple compatibles: 1) Install an F8 ROM with a modified RESET vector on the computer's motherboard as detailed in the "Modified ROM's" article of Hardcore COMPUTIST No. 6 or the "Dual ROM's" article in Hardcore COMPUTIST No. 19.

Apple //e - Apple //c: Install a modified CD ROM on the computer's motherboard. Don Lancaster's company (Synergistics) sells the instructions necessary to make this modification. Making this modification to an Apple //c will void its warranty but the increased ability to remove copy protection may justify it.

Recommended Literature: The Apple][Reference Manual and DOS 3.3 manual are musts for any serious Apple user. Other helpful books include: *Beneath Apple DOS*, Don Worth and Peter Leichner, Quality Software, $19.95; *Assembly Language For The Applesoft Programmer*, Roy Meyers and C.W. Finley, Addison Wesley, $16.95; and *What's Where In The Apple*, William Lubert, Micro Ink., $24.95.

Keying in Applesoft Programs: BASIC programs are printed in Hardcore COMPUTIST in a format that is designed to minimize errors for readers who key in these programs. To understand this format, you must first understand the formatted LIST feature of Applesoft.

An illustration- If you strike these keys:

```
10 HOME:REMCLEAR SCREEN
```

a program will be stored in the computer's memory. Strangely, this program will *not* have a LIST that is exactly as you typed it. Instead, the LIST will look like this:

```
10  HOME : REM CLEAR SCREEN
```

Programs don't usually LIST the same as they were keyed in because Applesoft inserts spaces into a program listing before and after every command word or mathematical operator. These spaces usually don't pose a problem except in line numbers which contain REM or DATA command words. The space inserted after these command words can be misleading. For example, if you want a program to have a list like this:

```
10  DATA 67,45,54,52
```

you would have to omit the space directly after the DATA command word. If you were to key in the space directly after the DATA command word, the LIST of the program would look like this:

```
10  DATA  67,45,54,52
```

This LIST is different from the LIST you wanted. The number of spaces you key after DATA and REM command words is very important.

All of this brings us to the Hardcore COMPUTIST LISTing format. In a BASIC LISTing, there are two types of spaces; spaces that don't matter whether they are keyed or not and spaces that must be keyed. Spaces that must be keyed in are printed as delta characters (Δ). All other spaces in a Hardcore COMPUTIST BASIC listing are put there for easier reading and it doesn't matter whether you type them or not.

There is one exception: If you want your checksums (See "Computing Checksums" section) to match up, you *must not* key in any spaces after a DATA command word unless they are marked by delta characters.

Keying In Hexdumps: Machine language programs are printed in Hardcore COMPUTIST as both source code and hexdumps. Only one of these formats need be keyed in to get a machine language program. Hexdumps are the shortest and easiest format to type in.

To key in hexdumps, you must first enter the monitor:

```
CALL -151
```

Now key in the hexdump exactly as it appears in the magazine ignoring the four digit checksum at the end of each line (a "$" and four digits). If you hear a beep, you will know that you have typed something incorrectly and must retype that line.

When finished, return to BASIC with a:

```
E003G
```

Remember to BSAVE the program with the correct filename, address and length parameters as given in the article.

Keying In Source Code The source code portion of a machine language program is provided only to better explain the program's operation. If you wish to key it in, you will need an assembler. The S-C Assembler is used to generate all source code printed in Hardcore COMPUTIST. Without this assembler, you will have to translate pieces of the source code into something *your* assembler will understand. A table of S-C Assembler directives just for this purpose was printed in Hardcore COMPUTIST No. 17. To translate source code, you will need to understand the directives of your assembler and convert the directives used in the source code listing to similar directives used by your assembler.

Computing Checksums Checksums are four digit hexadecimal numbers which verify whether or not you keyed a program exactly as it was printed in Hardcore COMPUTIST. There are two types of checksums: one created by the CHECKBIN program (for machine language programs) and the other created by the CHECKSOFT program (for BASIC programs). Both programs appeared in Hardcore COMPUTIST No. 1 and The Best of Hardcore Computing. An update to CHECKSOFT appeared in Hardcore COMPUTIST No. 18. If the checksums these programs create on your computer match the checksums accompanying the program in the magazine, then you keyed in the program correctly. If not, the program is incorrect at the line where the first checksum differs.

1) To compute CHECKSOFT checksums:

```
LOAD filename
BRUNCHECKSOFT
```

Get the checksums with

```
&
```

And correct the program where the checksums differ.

2) To compute CHECKBIN checksums:

```
CALL -151
BLOAD filename
```

Install CHECKBIN at an out of the way place

```
BRUN CHECKBIN,A$6000
```

Get the checksums by typing the starting address, a period and ending address of the file followed by a ⌃Y.

```
xxx.xxx⌃Y
```

And correct the lines at which the checksums differ.

How-To's Of Hardcore

Welcome to Hardcore COMPUTIST, a publication devoted to the serious user of Apple][and Apple][compatible computers. Our magazine contains information you are not likely to find in any of the other major journals dedicated to the Apple market.

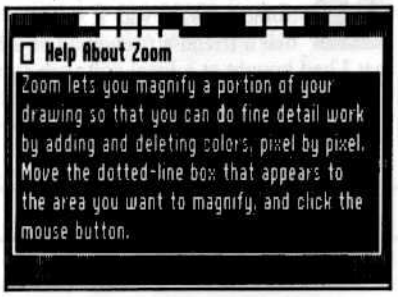
Our editorial policy is that we do NOT condone software piracy, but we do believe that honest users are entitled to backup commercial disks they have purchased. In addition to the security of a backup disk, the removal of copy protection gives the user the option of modifying application programs to meet his or her needs.

New readers are advised to read this page carefully to avoid frustration when attempting to follow a softkey or when entering the programs printed in this issue.

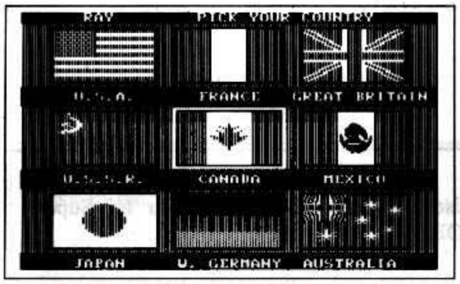
Issue No. 21

Hardcore COMPUTIST

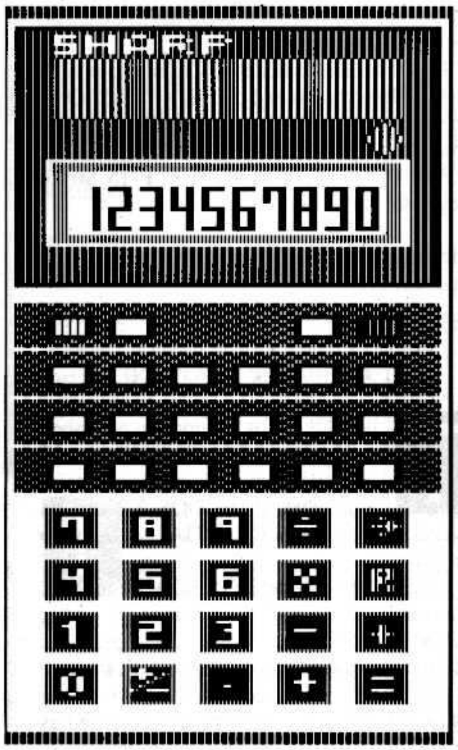
Publisher/Editor: Charles R. Haight **Technical Editor:** Ray Darrah
Production & Graphics: Lynn Campos-Johnson **Circulation:** Michelle Frank
Advertising: (206) 474-5750 **Printing:** Valco Graphics Inc., Seattle, WA
Hardcore COMPUTIST is published monthly, except December, by SoftKey Publishing, 5233 S. Washington, Tacoma, WA 98409
Phone: (206) 474-5750



Zoom lets you magnify a portion of your drawing so that you can do fine detail work by adding and deleting colors, pixel by pixel. Move the dotted-line box that appears to the area you want to magnify, and click the mouse button.



Pg. 7



Pg. 24

This month's cover: A hi-res graphic taken from Dazzle Draw (copyright Broderbund Software).

Address all advertising inquiries to Hardcore COMPUTIST, Advertising Department, PO Box 110816, Tacoma, WA 98411. Mail manuscripts or requests for Writer's Guides to Hardcore COMPUTIST, PO Box 110846-K, Tacoma, WA 98411.

Return postage must accompany all manuscripts, drawings, photos, disks, or tapes if they are to be returned. Unsolicited manuscripts will be returned only if adequate return postage is included.

The editorial staff assumes no liability or responsibility for the products advertised in the magazine. Any opinions expressed by the authors are not necessarily those of Hardcore COMPUTIST magazine or SoftKey Publishing.

Apple usually refers to the Apple][or][Plus Computer, and is a trademark of Apple Computers, Inc.

SUBSCRIPTIONS: Rates (for 6 issues): U.S. $20, U.S. 1st Class $24, Canada $34, Mexico $39, Foreign $60. Direct inquiries to: Hardcore COMPUTIST, Subscription Department, PO Box 110846-T, Tacoma, WA 98411. Please include address label with correspondence.

DOMESTIC DEALER RATES: Call (206) 474-5750 for more information.

Change Of Address: Please allow 4 weeks for change of address to take effect. On postal form 3576 supply your new address and your most recent address label. Issues missed due to non-receipt of change of address may be acquired at the regular back issue rate.

Input

Remarks About REM's

I've been a subscriber to your magazine since mid-1984 and have found your articles and other technical information to be interesting and, for me, educational. Keep up the good work and, above all, let's keep Hardcore for Apple II users only. While this request may sound selfish, I personally am not interested in articles on Mac or other enhanced versions that may come down the pike. There must be enough "II" users to keep Hardcore profitable?!

And now to my reason for writing. I'd like to see your BASIC program listings start off with simple REM statements without all the fancy stars and other designs such as the Lone Arranger on page 21 of Issue No. 16. It's frustrating and time consuming to try to exactly type in the designs that follow the REM statements. They don't do anything for the program; however, if they aren't typed in exactly, then the checksums are incorrect throughout the whole listing. In other words, let's keep it simple for those of us who have trouble entering backslashes, etc.

Again, I enjoy your magazine and it has been of great help to me as I try to master the nuances of the Apple!

Harry Nadin
Scottsdale, AZ

***Mr. Nadin:** Thanks for your support and that given by so many of our others readers. Our staff greatly appreciates it.*

Please be aware that the version of Checksoft which produces the checksums listed for the Applesoft program appearing in Hardcore COMPUTIST is configured to ignore all REM's. This is the reason why checksums for the first few lines of different programs listed in our magazine often will be identical even though the actual text in the REM statements vary. Therefore, you really do not have to bother typing in anything else beyond the actual REM command itself.

Needs Help Soldering

I have been reading Hardcore since its birth several years ago. Sometimes, it was a xerox of one that a friend's friend had or a copy that I had bought at a local software store. I finally have a subscription and feel that it is a great help.

I am desperately in need of some information. I have been trying to backup my copy of Robot Odyssey I by The Learning Company. It appears to copy and boots beautifully, but then the soldering pen (an essential part of the program) does not work. It uses one of the trickiest protection schemes I have ever run into and it looks like 3.3. AAARRRRGGGHHH!

I recently purchased a card known as a SPEEDemon from MCT. It is great as it takes almost all of the time out of analysis for any bit copy program. Many games are also interesting in the speeded-up mode.

Finally, some suggestions for Millionaire. You do not need a fancy ROM or Wildcard. Just pressing RESET many times works. The stack overflows and throws you into a listable unprotected program. Move the DOS and use the Swap Controller (HC No. 15) from there for a completely safe copy.

Is there anyone out there who thinks that something run by Hardcore COMPUTIST on the Source or Compuserve would be useful? I would love to see something of that nature.

B. Pierce
Tokyo, Japan

King Cribbage Deprotection

I have subscribed to Hardcore COMPUTIST for almost a year now and I have to commend you on a most informative and entertaining magazine.

Perhaps your readers can make use of the following softkey:

King Cribbage from Hayden Software is a good adaptation of the card game of cribbage and I have found it very interesting and educational. However, I do not like to use my originals and could not duplicate the disk successfully with either EDD or Copy II Plus. I did find it easy to unprotect it using Super IOB 1.2 and the enclosed controller. The entire procedure is as follows:

Boot King Cribbage and, as soon as the drive stops, reset into the monitor. Save the Hayden WTS by typing

```
6000<B800.BFFFM
```

Boot a DOS 3.3 slave disk with no HELLO program. After you get the Applesoft prompt, insert your disk with Super IOB on it and type

```
BSAVE RWTS,A$6000,L$800
```

Use a SWAP RWTS controller for Super IOB.

You can set the track start set-up at 17 ($11) because none of the tracks before this are used. Now use Super IOB to copy King Cribbage onto a normal DOS disk initialized for a HELLO program.

This is all that is necessary to unprotect the disk. I also recommend putting a "fast DOS" on the disk as it will speed up the loading time considerably.

P.S. Please add Microprose' F-15 Strike Eagle to your Most Wanted List.

Richard Moldovan
Tucson, AZ

Millionaire Revisited

The softkey for Millionaire published in Hardcore COMPUTIST No. 12 is insufficient in several respects. It will work if the DOS 3.3 disk is initialized with a null INITIAL as a hello program.

Using FID from the DOS 3.3 Master Disk, transfer the files mentioned plus RANDOM>DTA. Boot the original Millionaire disk as stated, push CTRL RESET at the first menu and SAVE INITIAL to the DOS 3.3 disk. The program that is saved as INITIAL is contained on the original disk. The start-up program cannot be called HELLO without also modifying line 248 of the Applesoft program PLAY on the original disk. After playing and saving a game, the above line runs INITIAL to begin a new game.

Now I will describe a better softkey. After reviewing the files on the original Millionaire disk using a nibble editor, it was apparent that the program files are in standard DOS 3.3 format; therefore, there is no need to modify DOS. Using ARMONITOR (Issue No. 12) and running ENTRA, it can be seen that the arm moves to track 23 (where the protection is located) on boot-up. Millionaire can easily be deprotected as follows:

1. Using the DOS 3.3 Master Disk or preferably a fast-DOS Master such as Pronto DOS, initialize a blank disk with a null INITIAL as a start-up program

```
INIT INITIAL
```

2. Using FID from the DOS 3.3 Master disk or Copy Files from Copy][Plus, transfer the following files from the original Millionaire

disk to the newly initialized disk:

```
INITIAL      COMMON     DESCRIP
RANDOM.DTA   INDUST     PLAYER
SAVE         STOCKS     PLAY
CHAIN        MESDATA
```

3. Boot the copy and play. This is a COPYAble disk.

The original can also be copied by first copying with COPYA and then bit-copying track 23 using Copy][Plus with default settings.

A.L. Head, Jr.
Sanger, TX

***Mr. Head:** We have tested your procedure and it seems to not only work for Millionaire, but for Squire as well. Thanks for the info.*

40 Track Drive

I really enjoy your magazine and find it very useful. I hope that you will find it possible to provide a special issue which would include softkeys published in issues no longer available.

In the February issue of Nibble magazine, I read a letter from Yin H. Pun who showed a way to increase the capacity of a diskette to 40 tracks. You might want to publish this information:

Boot normal DOS

```
CALL -151
AEB5:A0
B3EF:28
BEFE:28
```

Initialize a disk with this modification and you will have 40 tracks. The same can be accomplished from Applesoft:

```
POKE 44725,160:POKE 46063,40:
  POKE48894,40
```

FID will work properly, but COPYA has to be modified in order to initialize and read/write a 40 track disk:

```
BLOAD COPY.OBJ0 (From the DOS 3.3
  System Master)
CALL -151
302:28 N 35F:28
BSAVE COPY.OBJ0.40,A$2C0,L$10B
CTRL C
LOAD COPYA
```

Now type

```
70 PRINT CHR$(4);"BLOAD COPY.OBJ0.40" : REM
    $2C0
75 POKE 44725, 160 : POKE 46063, 40 : POKE
    48894,40
SAVE COPYA 40 TRACK
```

This should work on most disk drives.

R. Boreiko
Calgary, Alberta
Canada

***Mr. Boreiko:** As a matter of fact, Hardcore COMPUTIST has just released the Book of Softkeys Vol. 1, a manual which contains deprotection techniques compiled from Hardcore COMPUTIST No's. 1-5 and the three original issues of Hardcore Computing magazine (100+ pages). See the ad on the back cover of this issue for ordering information.*

PFS:Write & Sensible Speller

It looks as though I (and probably others) are being discriminated against because we use PFS:Write and Sensible Speller 4.1E to check documents. In past issues, there have been many things written about various Sensible Speller versions but I, having version 4.1E, seem to have been left out. I also know relatively little about machine language, so I don't know how to release it myself. Could you or anyone tell me how to do it?

For those who use PFS:Write, it may be released by:

1) Copying it with COPYA
2) Sector edit track 19, sector 02
   Change address: 63 from 04 to 29
   Sector edit track 1D, sector 02
   Change address: 63 from 04 to 29
3) Write protect before using.

This is essentially the same thing that Gary Wolfe did in Issue No. 14 for PFS Software.

Paul Pokorny
Ames, IA

Digging Into PFS

Re. The PFS: Series of Software. I contacted you about three months ago and you sent me an update on the softkey information. I then submitted updated information to you and have not received a reply since. I assumed it was inadvertently lost somewhere in the shuffle. Since that time, I have had many occasions to look at PFS: releases and have found that, in every instance, searching for and changing the following strings as shown will give a functioning COPYAable copy. One note: The built-in copy routine on the //e, //c series will NOT function from the COPYA copy - but who needs it?

I use the Inspector/Watson routine for Disk Search and Edit. The strings to search for and change are as follows:

```
1: D0 04 88 98 F0 27 (Change $04 to $29)
2: D0 0E 88 98 F0 31 (Change $0E to $33)
```

The number of occurrences and position on the disk vary widely. In the latest //e, //c releases (those with the built-in copy routine) I have had these results:

A. 1 occurrence of String 1and 2 occurrences of String 2
B. 2 occurrences of String 2
C. 1 occurrence of String 1 and 1 occurrence of String 2

H.W. Madison
Bellingham, WA

Bank Street Problems

Still recording votes? Add my vote to keep Hardcore COMPUTIST an Apple][magazine only. Too many good magazines have gone by the wayside branching out, and yours has been getting better. I have every copy and updates you published for reference.

I really enjoyed the Print Shop Grabber program (Issue No. 17) and have discovered that, using the shrink portion of my Complete Graphics program, I can reduce the complete hi-res screen so it can be saved by the Graphic Grabber program.

I am also enclosing a printout of BSW.Controller from Issue No. 18. Can you tell me if there was a misprint in lines 1030 and 1250? I used your new Checksoft v1.2 from the same magazine to discover that, even though these two lines agree with your magazine, their checksums do not. I also discovered that it will not copy my Scholastic Version No. 0-590-95570-S of Bank Street Writer. This is the new expanded version.

Joseph F. Brown
Fort Wayne, IN

Input cont...

Mr Brown: The controller we printed for the Scholastic version of Bankstreet Writer was correct. However, by some act of God, the checksums were not. Listed below are what the checksums should have read. In addition, we have noted that some of the Scholastic Bank Street Writers out there can only be deprotected using the method outlined in Hardcore COMPUTIST No. 10.

```
1000 - $356B    1140 - $1650
1010 - $ED2E    1150 - $0850
1020 - $F6AC    1160 - $1CA1
1030 - $B776    1170 - $2438
1040 - $873E    1180 - $74EF
1050 - $2483    1190 - $9A7E
1060 - $277E    1200 - $D3BD
1070 - $04B8    1210 - $0099
1080 - $90B0    1220 - $91E0
1090 - $08F6    1230 - $253A
1100 - $39F6    1240 - $8AAB
1110 - $4F57    1250 - $1010
1120 - $5386    1260 - $0F1A
1130 - $6BFE
```

Help With Applewriter

I am writing for a couple of reasons. First I would like to tell you that you produce the finest magazine on the market. The information you provide is invaluable to anyone with an Apple computer. Also, I wish to thank those who contribute their information so freely to this publicaiton. Keep up the good work and keep the magazine coming (I eagerly await each issue).

Now for my problem. In Hardcore COMPUTIST No. 18 an excellent softkey for Applewriter //e was published. The author, Peter Edelsten, stated that you can make a copy of this program using COPYA. I found that I had to use a modified version of COPYA so that it would ignore the read errors as it encountered them. The modified version is pointed out in the documentation for Bag of Tricks and in Issue No. 15, page 10. It is as follows:

1) Copy a copy of COPYA to another disk.
2) Then

```
BLOAD COPY.OBJ0
```

and do the following

```
POKE 929,234
BSAVE COPY OBJ.0,A$2C0,L$10B
```

to the same disk on which you saved COPYA.

Now follow the rest of the softkey and you will have an unprotected copy of Applewriter //e (I have been waiting for this softkey for a long time!).

Finally, has anyone come up with a softkey for the new enhanced version of Bankstreet Writer by Broderbund? This version does not eliminate some of the old hassles.

Ron Kemp
Grand Rapids, MI

Digging Into RDOS

I have discovered a way to read all SSI:RDOS games.

To use this you must make a 32K boot disk (as per instructions from Lion's Share in Hardcore COMPUTIST No. 12). After the 32K boot is made, boot up an SSI disk with RDOS. When the disk drive stops, insert the 32K boot disk. Now you must boot up this disk (pressing RESET will work). Once this is done, type

```
BSAVE RDOS,A$B100,L$EFF
```

Now make a text file containing the following information:

```
POKE72,0:CALL -151
3D0:4C B0 B9 4C 00 E0 4C FD
3D8:AA 4C 00 BA AD 0F 9D AC
3E0:0E 9D 60 AD C2 AA AC C1
3E8:AA 60 60 60 60 EA EA 4C
3F0:59 FA B0 B9 1C 4C 03 B3
3F8:4C 65 FF 4C 65 FF 65 00
48:D4
3F2:59 FF 5A
D43CG
```

Now you are ready to investigate SSI:RDOS games. To do this, just boot up your 32K disk, and type

```
BLOAD RDOS
EXEC (name of textfile)
```

Now, when you want to work with an SSI game, type & and command. Some of these commands are CAT- (does a CATALOG), RECALL" (does a BLOAD), STORE" (does a BSAVE), LOAD (does a LOAD), SAVE (does a SAVE).

Well, that's it. If you want to save a program to a normal DOS 3.3, just use normal DOS commands without the &.

Shantul Nigam
Southington, CT

Echo Patron

Here is a parameter I'd like to pass on concerning the Sales Edge by The Human Edge.

Every attempt to make a copy using the most popular bit copiers failed, but ECHO v1.0 easily makes a copy. More people should look into this excellent bit copier and v2.0, when it comes out.

Mike Coffey
Columbia, MO

List Handler Info

Re. M.D. Mullins' letter on the Handlers package. Although I don't use Spell Handler, I use Word and List Handler frequently. I also have a continuing love/hate relationship with the software (it's amazing how personal you can get over a piece of code!).

I think that for price and performance, these programs are excellent; I like the Word Handler's features for the price, even though the screen refresh on 80-columns and somewhat limited print options drive me bats. List Handler is quite easy, very fast on sort, and has good storage using data compression. Now if only they could sort several deep and expand the reports generator??

If Mr. Mullins has an earlier List Handler version, on the utility side of the disk there is a word handler-ASCII text file (both ways) conversion utility. Regarding the copy-protection, ALS now offers current versions that are COPYA to run on the //c. Unfortunately, the programs were not otherwise enhanced and they eliminated the W.H.-ASCII file utility from the utility disk. I guess that's progress. Anyway, ALS will provide the updates for about $10 each.

P. Barbiere
Batavia, OH

Please address letters to: Hardcore COMPUTIST, Editorial Department, PO Box 110846-K, Tacoma, WA 98411. Include your name, address and phone.

Correspondence appearing in the INPUT section may be edited for clarity and space requirements. In addition, because of the great number of letters that we receive and the small size of our staff, a response to each letter is not guaranteed.

Readers' Softkey & Copy Exchange

Advanced Blackjack Softkey

By Jim Mitchell

Advanced Blackjack
Muse Software
347 N. Charles Street
Baltimore, MD. 212Ø1
$49.95

Requirements:
Apple][Plus or //e
Super IOB v1.2
One blank disk

Advanced Blackjack is a casino game program with several nice features. If you simply enjoy playing blackjack or, like me, are serious about the game and want to practice before hitting the casinos, then this program is for you. The program is designed to teach you how to play the game of blackjack, as well as the ZEN card counting system.

The normal methods for making a copy didn't work well on this disk and, even when I could get a working bit copy, it still took forever to load.

Therefore, (like many of my fellow Hardcore COMPUTIST readers), I decided to unprotect the program and see if I could use a faster DOS. The documentation states "DO NOT UPDATE this disk with other versions of the disk operating system" or it will destroy the program disk. While poking around through the monitor, the reason for this become apparent: the copy protection schemes on this program involved several changes to DOS.

First, the address field header was changed from D5 AA 96 to D5 DA 96, the data field header from D5 AA AD to D5 DA AD, and the end of data markers from DE AA to DE DA. After booting a normal DOS 3.3 disk, then making these changes in my normal DOS, I found that I could CATALOG the original disk but that some of the files were garbled. I next checked the RWTS translate tables. If you would like to see these tables as they exist in normal DOS, boot your system master (or any disk with normal DOS), enter the monitor (CALL -151), and type "BA29.BA68" for the nibble translate table and "BA96.BAFF" for the byte translate table.

Comparing the RWTS tables for Advanced Blackjack with those of normal DOS I found that there were indeed changes in the tables. At location BA4C I found AA instead of DA and at BAAA I found 23 instead of AA. After making these changes to normal DOS, as well as the previous changes above, the original disk could be CATALOGed and the program would run. A quick trial of Diversi-DOS and David DOS showed that these two fast DOS versions would also run the program.

Step-By-Step

Now to make an unprotected copy by using the Super IOB program:

1) First make a slave disk using the DOS version that you wish to use

INIT HELLO

2) Second, use the controller below with Super IOB to copy the disk.
3) You now have a backup copy for Advanced Blackjack.

Advanced Blackjack Controller

```
1ØØØ REM ADVANCED  BLACKJACK
1Ø1Ø TK = 3 :ST = Ø :LT = 35 :CD = WR
1Ø2Ø T1 = TK : GOSUB 49Ø : RESTORE : GOSUB 19Ø
     : GOSUB 21Ø : GOSUB 17Ø : POKE 47786 ,35
1Ø3Ø GOSUB 43Ø : GOSUB 1ØØ :ST = ST + 1 : IF ST
     < DOS THEN 1Ø3Ø
1Ø4Ø IF BF THEN 1Ø6Ø
1Ø5Ø ST = Ø :TK = TK + 1 : IF TK < LT THEN 1Ø3Ø
1Ø6Ø GOSUB 23Ø : POKE 47786 ,17Ø : GOSUB 49Ø : TK
     = T1 :ST = Ø
1Ø7Ø GOSUB 43Ø : GOSUB 1ØØ :ST = ST + 1 : IF ST
     < DOS THEN 1Ø7Ø
1Ø8Ø ST = Ø :TK = TK + 1 : IF BF = Ø AND TK < LT THEN
     1Ø7Ø
1Ø9Ø IF TK < LT THEN 1Ø2Ø
11ØØ HOME : PRINT "DONE^WITH^COPY" : END
5ØØØ DATA 213 ,218 ,15Ø ,213 ,218 ,173
5Ø1Ø DATA 222 ,17Ø ,222 ,218
```

Controller Checksums

```
1ØØØ  - $356B      1Ø7Ø  - $A35C
1Ø1Ø  - $3565      1Ø8Ø  - $F51B
1Ø2Ø  - $EFE8      1Ø9Ø  - $F2FF
1Ø3Ø  - $F9E9      11ØØ  - $A826
1Ø4Ø  - $Ø586      5ØØØ  - $827D
1Ø5Ø  - $3D2B      5Ø1Ø  - $2396
1Ø6Ø  - $AB59
```

Deprotecting Megaworks

By Mike Stafford

Megaworks
Megahaus
57Ø3 Oberlin Drive
San Diego, CA 92121
$149.95

Requirements:
Megaworks
1 blank disk
COPYA

Anyone who regularly uses the Appleworks program can tell you that although it is an excellent program, its usefulness is greatly limited by the absence of a spelling checker and mailmerge option. That problem has now been remedied by the Megahaus company with the release of their excellent Megaworks program. Unfortunately, they have chosen to copy protect this program.

The copy protection on this disk is minimal and apparently was added just to prevent the disk from being copied with the COPYA program on your DOS 3.3 System Master. The only protection scheme that was used on my copy was in the form of a non-standard address format on track 22, sector 8. When you copy this disk with COPYA, it halts at this point due to a read error. This is simple enough to remedy. Just modify COPYA to ignore the error and the disk will copy normally.

Here's the "cookbook" method:

1) Get out your DOS 3.3 System Master. Load COPYA and it's object file

RUN COPYA

2) As soon as COPYA comes up and asks you to select the source slot, break into BASIC

CTRL C

3) Delete line 7Ø and modify the program so that it ignores the read error on track 22, sector 8

7Ø
CALL -151
BE48:18

4) Run the program

EØØ3G
RUN

5) Now follow the prompts and make your copy.

Your Megaworks disk is now unprotected. The Dictionary disk and the Example disk are not protected at all so you can copy them with any program you like.

Unprotecting Summer Games

By Dan Lui

Summer Games
EPYX
1043 Kiel Court
Sunnyvale, CA 94809
(408) 745-0700

Requirements:
Apple][with 64K
One or more disk drives
Summer Games program disk
Super IOB v1.2
Two blank disk sides

Summer Games from EPYX requires 64K but deprotecting it can be accomplished with a 48K machine.

The protection on this disk was easily determined: EPYX has changed the address and data trailers from $DE AA to $FF FF and included a nibble count routine.

After a thorough investigation (and boot code tracing) I found that the nibble checker routine resided in the RAM card at $D003. I then found the routine that calls the nibble count. The call was in memory at $B70D (part of the DOS second stage boot. A quick scan of the disk revealed the call to the nibble count at only one location on the disk. Therefore, all I had to do was copy the disk using normal ending markers on the copy, NOP the call to the nibble count routine and edit the Summer Games RWTS so that it would read and write with normal sector markers.

The Super IOB controller at the end of this article performs all of these tasks in one sweep. The resulting disk can be copied by COPYA and works exactly like the original.

Step-By-Step

1) Type in the controller at the end of this article and install it into Super IOB v1.2.
2) Start up Super IOB v1.2 with the controller modification made to it

RUN

3) Answer the questions and copy both sides of the original disk.

What Happened

When Super IOB copied your disk, it changed the end of address and end of data markers from $FF FF to the normal value of $DE AA. In addition, it performed the following sector edits which tell the Summer Games DOS to read and write normally and disable the nibble count routine.

Track	Sector	Byte	From	To
$00	$02	$9E	$FF	$DE
$00	$02	$A3	$FF	$AA
$00	$03	$35	$FF	$DE
$00	$03	$3F	$FF	$AA
$00	$03	$91	$FF	$DE
$00	$03	$9B	$FF	$AA
$22	$02	$0D	$20	$EA
$22	$02	$0E	$03	$EA
$22	$02	$0F	$D0	$EA

Enjoy your unprotected Summer Games and go for the GOLD!

Summer Games Controller

```
1000 REM SUMMER GAMES
1010 TK = 0 :ST = 0 :LT = 35 :CD = WR
1020 T1 = TK : GOSUB 490 : RESTORE : GOSUB 170
1030 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST
     < DOS THEN 1030
1040 IF BF THEN 1060
1050 ST = 0 :TK = TK + 1 : IF TK < LT THEN 1030
1060 GOSUB 310 : GOSUB 230 : GOSUB 490 :TK = T1
     :ST = 0
1070 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST
     < DOS THEN 1070
1080 ST = 0 : TK = TK + 1 : IF BF = 0 AND TK < LT THEN
     1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "DONE^WITH^COPY" : END
5000 DATA 255 ,255 ,255 ,255
5010 DATA 9^CHANGES ,0 ,2 ,158 ,222 ,0 ,2 ,163
     ,170
5020 DATA 0 ,3 ,53 ,222 ,0 ,3 ,63 ,170 ,0 ,3 ,145
     ,222 ,0 ,3 ,155 ,170
5030 DATA 34 ,2 ,13 ,234 ,34 ,2 ,14 ,234 ,34 ,2
     ,15 ,234
```

Controller Checksums

1000	- $356B		1080	- $0406
1010	- $3266		1090	- $7922
1020	- $7F0A		1100	- $65C7
1030	- $690B		5000	- $AF8D
1040	- $51E4		5010	- $53FE
1050	- $FDC1		5020	- $12D0
1060	- $54A8		5030	- $9875
1070	- $5CAD			

Backing-Up The College Entrance Exam Prep

By Joel Huse

College Entrance Examination Prep
Borg-Warner

Requirements:
Apple][,][Plus, //e, //c
At least one disk drive
Any bit copy program
DOS 3.3 editor
Twelve blank disks
One formatted disk

This procedure is dedicated to all the schools (like our own) who have begun to lose their original College Entrance Examination Prep (CEEP) disks.

Anyone who is familiar with this valuable teaching aid has probably discovered that CEEP is very fragile. The program used by our school already shows errors after only short use. And although the copy protection has, so far, defied my cracking attempts (this procedure will allow you to backup the disks, only), it is not too difficult to copy.

Extensive modifications have been made to the track structure which will confuse anybody's RWTS and some sort of nibble count routine is performed on the tracks around 9.5. Unfortunately, if you copy track 9.5, it erases tracks 9 and A which are accessed quite often. I've tried very dilligently to copy tracks 8.75 through A.25 accurately, but to no avail. Instead of buying 70 track drives, you'll be happy to know that there is an easier way.

If you don't care how I discovered the routine, ahead skip to the section entitled Step-By-Step.

Hitting The Books

The boot sounded pretty much normal, so I decided to examine track zero with my sector editor that can read altered sector markers. Track 0, sector 0 was identical to track 0, sector 0 of a normal DOS disk. I followed the boot further and was able to capture the CEEP RWTS (explained in Steps 2 and 3).

Following the boot further I found a JuMP to $B444 which turned out to be the start of the track 9.5 nibble count routine. I then devised a modification which would disable it.

Step-By-Step

1) First use a bit copy program to copy all twelve sides of CEEP using the default parameter settings. If you are using a pre-4.4C version of Copy][Plus, make these parameter changes: 10=AD, 9=0, A=3.
2) Boot one of the copied disks and, before it completes booting, press RESET (listen to your drive when it boots and, just after hearing the drive head slide a short distance (to track $02), press RESET).

We will now use the CEEP RWTS in memory to read track 2, sector 3 where the nibble count routine is stored. Then we will modify this routine so that it no longer functions as intended and write it back out. The modification presented here may not be the most efficient, but it is the most reliable.
3) Set up the CEEP RWTS to read track $20, sector $03 into memory at $4400

```
CALL -151
B7E8:01 60 01 00 02 03
B7F0:00 44 00 00 01
```

4) Now we need a routine that sets up the 6502 registers and calls the RWTS

```
300:A9 B7 A0 E8 20 B5 B7 A9
308:00 85 48 60
```

5) Read in the sector using this routine

```
300G
```

6) The nibble count routine is located at $4400. Type in this hexdump to defeat it

```
447B:A9 D4 85 00 A0
4480:0F B9 90 B4 99 00 15 88
4488:D0 F7 4C B8 B4 00 00 00
4490:00 D5 AD BE B7 15 15 F2
4498:F3 DA AD DA AD E6 9D D5
```

7) Tell the RWTS to write this sector back to track 2, sector 3

```
B7F4:02
```

8) Write the sector to a CEEP copy disk by using the $300 routine

```
300G
```

9) Insert another CEEP copy disk and follow the procedure each time through Step 8 until all the disk sides have been done.

That should do it. Your backup of College Entrance Exam Prep is exactly like the original and performs like the original.

I have seen some people try to crack CEEP using Super IOB and the Swap Controller, but I can say, from personal experience, that there have been some pretty sad endings. The protected CEEP RWTS can't even read half of the sectors on the original CEEP disk. But, maybe this is all for the best. I can just imagine a COPYAable version of this program in a classroom enviroment. It could prove somewhat devastating to Borg-Warner.

I hope this information is of help to Mr. D.J. Ward of South Williamsport (Hardcore COMPUTIST No. 16) and any others who have been unable to copy the program. And although the Critical Reading disks probably use a similar protection, I haven't been able to get a copy of the program to verify it.

Applewriter //e Softkey Revisited

By Denny Colt

Requirements:
Apple //e
COPYA or any other normal copy program
One blank disk
Disk editor

(This is an addition to the Applewriter //e softkey which appeared in Hardcore COMPUTIST No. 18- Editor.)

My local library uses Applewriter //e on its Apple. Because of the number of untrained people who use the disk, the need for a backup became critical. I previously had used Copy][Plus to make a backups for them but, thanks to Hardcore COMPUTIST, they can now make their own.

I did run into some small problems softkeying MY version on Applewriter that I'd like to pass on to you readers.

First of all, I found that the sector edit method (presented in Hardcore COMPUTIST No. 18) would not work for my disk. I then loaded in the files, but the routines had moved. Then I realized that Mr. Edelsten's version BLOAD's at $2300 and mine BLOAD's at $1900. The files are the same, but have different starting addresses. To softkey the disk, I needed to subtract $A from the locations in Issue No. 18.

In A Nut Shell

To deprotect this slightly different version of Applewriter //e:

1) Boot a normal disk.
2) Enter the monitor and type

```
CALL -151
BLOAD OBJ.APWRT][F
3104: EA EA EA
UNLOCK OBJ.APWRT][F
BSAVE OBJ.APWRT][F,A$1900,L$30D1
LOCK OBJ.APWRT][F
BLOAD OBJ.APWRT][E
2FAD: EA EA EA
UNLOCK OBJ.APWRT][E
BSAVE OBJ.APWRT][E,A$1900,L$2F58
LOCK OBJ.APWRT][E
```

To put the free sector patch in place, type it in just as mentioned, but start it at $230F.

Once I had the program softkeyed I couldn't resist making a few improvements. I first did a little work on the free sector patch above. Mr. Bragner's original patch will only determine how much free space is on the last disk accessed. With a little work, it can be made to tell how much space is on the current disk in the drive. To do this:

1) Type in the patch in HC No. 18
2) Before saving it, do this:

```
30F<2D0F.2D42M (230F for mine)
2D12<30F.342M
2D0F:20 F7 E7
```

3) Save it as normal.

This puts a JSR to the VTOC read routine at $E7F7 and removes the bell for space (this routine replaces the Convert Applewriter 1.1 files option in the CTRL-Q menu). To recover this option, and add the ability to use 40 or 80 columns at will, make the following patch to the boot program:

```
BLOAD OBJ.BOOT
CALL -151
1C51:D5 1E
1CD7:C2 1E
1EC2:20 EA 1D AD 00 C0
1EC8:C9 B4 D0 08 8D 10 C0 A9
1ED0:00 8D 3E 1D 60 AD 00 C0
1ED8:C9 B1 D0 08 8D 10 C0 A9
1EE0:C5 8D D9 1D 4C B9 1D
UNLOCK OBJ.BOOT
BSAVE OBJ.BOOT,A$1C00,L$2E7
LOCK OBJ.BOOT
```

To use the 40-column version, just press "4" during the boot process. To convert Applewriter 1.1 files, press "1" during the boot. This will load in the 64K version of Applewriter that still has convert option intact.

One final tidbit: I noticed that the Applewriter DOS seems to be much like regular DOS, except that it's up $3800 bytes higher than regular DOS. I hope that helps all those people who would like to upload the program to a hard disk.

Softkey for ARCHON

By Steve McLendon

Archon
Electronic Arts
2755 Campus Dr.
San Mateo, CA 944Ø3
$39.95

Requirements:
64K Apple][Plus or equivalent
A sector editor
Super IOB v1.2
One blank disk

Just when Pete Levinthal blew the whistle on the protection used for several of Electronic Arts' games (HC No. 13, pg. 26), EA upgraded their protection schemes. Because of the new technique, I met a real challenge in deprotecting one of their most recent releases: Archon.

First, there are two nibble count tracks--Trks 5 and 6. In addition, self-writing code and indirect jumps are used throughout the boot process.

To begin with, the start of data bytes have been changed from $DE AA AD to the usual EA mark of $DE BB CF. This is easily circumvented with a Super IOB controller that reads using these marks and writes using the normal ones. In addition, the Super IOB controller performs the following sector edits which fixes the ES RWTS so that it looks for the new normal marks:

Trck	Sctr	Byte	From	To
Ø2	Ø3	47	BB	AA
Ø2	Ø3	51	CF	AD

However, if you boot your Super IOB'd copy you will see that it has one minor drawback--it doesn't work. Ah, yes. The nibble count. The routine is accessed a total of three times; first, after the EA logo is displayed but just prior to the ARCHON graphics; second, after the options page just as the board is about to be displayed and; last, just after you make your fourth move on the board (but this time both tracks are checked, as well as some other sneaky stuff).

If you look at tracks 5 and 6 with a nibble editor you'll see a bunch of B4's. This is an important clue and, if you have a utility which can scan the entire disk for certain bytes, we can begin by searching for the two bytes "C9 B4" (CMP #$B4). Bingo! The nibble count routine is instantly exposed in two places---Trk Ø1, Sctr ØF and Trk Ø1, Sctr ØC.

The obvious thing to do here is to replace the AØ 2Ø at byte $Ø3 with 18 6Ø. Try it, then boot. As you can see, your disk drive is now completely confused and doesn't know what to do. Unfortunately, the EA security guards have anticipated your attempt and incorporated a checksum scheme into the boot stage that will detect any sort of tampering. Even this routine may be found and modified, but I'll save you a lot of hassle here and tell you it will not do any good because the routines are rewritten in memory for subsequent checks.

Now we fight fire with fire. If EA can use self-writing code then, so can we. Track 2 contains the ARCHON RWTS and, scanning this track for unused space, we see that the last half of sector Ø contains nothing but FF's and ØØ's. So, let's put a routine in the RWTS itself that will defeat the nibble count for us. With your sector editor, input the following code on Trk 2, Sctr Ø, beginning at byte $FØ:

```
$FØ:48 A9 EA 8D ØØ AØ 8D Ø1
$F8:AØ 8D Ø2 AØ 68 4C D4 BC
```

This translates into the following code:

```
PHA        SAVE A-REG
LDA #$EA   CODE FOR NOP
STA $AØØØ NOP THE CODE
STA $AØØ1 THAT CALLS THE
STA $AØØ2 NIBBLE COUNT
PLA        RESTORE A-REG
JMP $BCD4 CONTINUE BOOT
```

Now change Trk 2, Sctr 3, bytes $Ø1 and $Ø2 from D4 BC to FØ BF (we can modify other tracks at will- it's only track 1 that EA checksums to death). What we have done is to insert a small routine into ARCHON's RWTS that will NOP the instructions to do the nibble count. The mod to Trk 2, Sctr 3 was needed to direct the program to jump to our little routine instead of continuing with its own dirty work.

Now boot the disk. The title page loads and the music plays--you can even get to the Options Menu. But, after you load in the rest of the game program and the board is about to be displayed, you hear the nibble count again and the game hangs. Not only has the count been relocated, but it has also been modified. Even if you can reset into the monitor at this point and scan memory for the nibble count, you might have trouble finding it because it has been relocated up to Bank 1 of your Ramcard (it used to be at $57ØØ). If we snoop around in memory we will find that the count is invoked from $6A4A, with the drive motor being turned on at $6A3C. Don't look for this code on the disk anywhere because you will not find it (remember, I said it was self-writing).

We can, however, pull the same trick and have the program rewrite the code to suit our own purposes. Make the following mods to Trk 2, Sctr Ø, beginning with byte $D8:

```
$D8:A9 18 8D 3C 6A A9 6Ø 8D
$EØ:3D 6A 4C ØØ Ø8
```

This translates into the following code:

```
LDA #$18  CODE FOR CLC INSTRUCTION
STA $6A3C PUT AT START OF ROUTINE
LDA #$6Ø  CODE FOR RTS INSTRUCTION
```

```
STA $6A3D PUT AFTER CLC INSTRUCTION
JMP $Ø8ØØ CONTINUE PROGRAM
```

Also, on Trk 12, Sctr ØC, change bytes $95 and $96 from ØØ Ø8 to D8 BF.

If you now boot the disk, everything seems to go just fine. You can even start playing the game. But after your fourth move the program accesses the disk again and hangs up. We could have put our 18 6Ø mod up at $D8ØØ which would have allowed the game to continue after the fourth move, but your wizard would have been unable to cast his spells.

Looking for a routine which calls the nibble count here will be to no avail, as it is cleverly disguised in an indirect jump. I mentioned that the count now resides in Bank 1 of the Ramcard. To address it, it must be enabled with a $CØ8B (LDA $CØ8B, BIT $CØ8B, etc.). Scanning through memory we find only a few such occurrences, and it doesn't take much to discover that the one at $9189 is the culprit. This subroutine starts at $9174 and is called from $DD33.

We could put three NOP's here and all would seem to be well but, again, your wizard would not be able to cast any spells after your fourth move. The program sets some flags and the nibble count routine resets those flags back to normal. Bypass the nibble count and the flags are not reset.

Since the nibble count occurs after your fourth move, we should strongly suspect that a particular memory address acts as a counter, being decremented after each move. Looking backward through memory from $DD33 we soon run across something interesting. At $DCC5 we find that memory address $DCBB is checked and decremented. On a hunch, let's NOP the DEC instruction at $DCCA and see what happens.

To do this, change the instruction before it is moved into the Ramcard. All of the code in Ramcard Bank 2 was relocated from $2ØØØ-$4FFF so, if we are going to change $DCCA, we must do so at $2BCA. Make the following mod to Trk 2, Sctr Ø starting at byte $E2:

```
$E2:A9 EA 8D CA 2B 8D
$E8:CB 2B 8D CC 2B 4C ØØ Ø8
```

This translates into the following code:

```
LDA #$EA  CODE FOR NOP
STA $2BCA PUT AT DECREMENT
STA $2BCB OF TURN COUNTER
STA $2BCC
JMP $Ø8ØØ CONTINUE PROGRAM
```

As you can see, we are now defeating the "4-turn counter" as well as the second nibble count before doing the jump to $Ø8ØØ. Write this back to disk and all should work well, and you will no longer have to put up with that annoying 5-6 second delay at the end of your fourth turn all the time.

Summary

1) Install the ARCHON controller and run Super IOB.

2) With a track/sector editor make the following modifications:

Trk	Sctr	Byte	From	To
Ø2	Ø3	Ø1	D4	FØ
Ø2	Ø3	Ø2	BC	BF
12	ØC	95	ØØ	D8
12	ØC	96	Ø8	BF

3) Also, with your track/sector editor, make the following modifications to Track 2, Sector Ø, beginning with byte $D8:

```
$D8:A9 18 8D 3C 6A A9 6Ø 8D
$EØ:3D 6A A9 EA 8D CA 2B 8D
$E8:CB 2B 8D CC 2B 4C ØØ Ø8
$FØ:48 A9 EA 8D 4C A1 8D 4D
$F8:A1 8D 4E A1 68 4C D4 BC
```

Archon Controller

```
1ØØØ REM ELECTRONIC ARTS (5/6 )
1Ø1Ø TK = Ø :ST = Ø :LT = 35 :CD = WR
1Ø2Ø T1 = TK : GOSUB 49Ø : IF TK > 3 THEN RESTORE
     : GOSUB 21Ø
1Ø3Ø GOSUB 43Ø : GOSUB 1ØØ :ST = ST + 1 : IF ST <
     DOS THEN 1Ø3Ø
1Ø35 IF TK = 2 THEN GOSUB 21Ø
1Ø4Ø IF BF THEN 1Ø6Ø
1Ø5Ø ST = Ø :TK = TK + 1 + (TK = 4 ) * 2 : IF TK < LT
     THEN 1Ø3Ø
1Ø6Ø GOSUB 31Ø : GOSUB 23Ø : GOSUB 49Ø :TK = T1
     :ST = Ø
1Ø7Ø GOSUB 43Ø : GOSUB 1ØØ :ST = ST + 1 : IF ST <
     DOS THEN 1Ø7Ø
1Ø8Ø ST = Ø :TK = TK + 1 + (TK = 4 ) * 2 : IF BF = Ø
     AND TK < LT THEN 1Ø7Ø
1Ø9Ø IF TK < LT THEN 1Ø2Ø
11ØØ HOME : PRINT : PRINT "DONE^WITH^COPY" : END
5ØØØ DATA 213 ,187 ,2Ø7
5Ø1Ø DATA 2^CHANGES ,2 ,3 ,71 ,17Ø ,2 ,3 ,81 ,173
```

Controller Checksums

```
1ØØØ  - $356B      1Ø6Ø  - $Ø8Ø3
1Ø1Ø  - $3266      1Ø7Ø  - $ØØØ6
1Ø2Ø  - $2E67      1Ø8Ø  - $8DØD
1Ø3Ø  - $3866      1Ø9Ø  - $4CD7
1Ø35  - $7BØB      11ØØ  - $31Ø7
1Ø4Ø  - $2B8Ø      5ØØØ  - $E127
1Ø5Ø  - $7275      5Ø1Ø  - $161B
```

Adventure Tips

Knight of Diamonds
Sir-Tech Software

The answer to the riddle at the end is, of course, Knight of Diamonds.

Legacy of Llylgamyn
Sir-Tech Software

Level 2:

I am always around you always
But you have never seen me
You would never leave me
But were I gone
You would not cry out for me
What am I?

Answer: AIR

Level 4:

Ours is an uneasy friendship
Your fathers toiled long to tame me
But even now I will go my own way
If you do not tend me well
I am a powerful destroyer
Yet my frailty is great
What am I?

Answer: FIRE

Level 5:

I am one of the deck of 78
And victory I foretell
Two bold steeds draw me forth
And stars shine from my crown
What card am I?

Answer: THE CHARIOT

Level 6:

On my roll
You move forward
My turn is your turn
What am I?

Answer: WHEEL

Black figure on a black horse
He carries the white rose
The rising sun-change will come
It follows wherever he goes
What card am I?

Answer: DEATH

The password asked for is:
"Abdul sent you"

Contributed by Paul Vorndam

Demystifying The Quarter Track

By Bruce Wayne Jones

Requirements:
Apple computer with 48K or more
Disk drive capable of half-tracks
DOS 3.3

I am writing this article for two reasons: the first is that I would like to address a statement made by Ray Darrah in the Whiz Kid column in Hardcore Computist No. 17. To paraphrase, "...quarter-tracks aren't worth much.". The second reason stems from the fact that nearly every time someone encounters a protection scheme they don't understand, they quickly label it nibble counting or quarter-tracks. Although it will take another author to demystify nibble counting, in this article I'll try to shed a little light on the mysterious quarter-track.

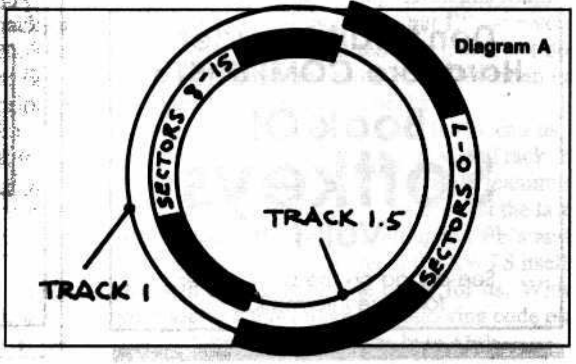
Background

As every experienced computer user should know, DOS 3.3 divides a disk into 35 concentric circles of data called tracks. Because of this, the data on the disk is always one (1) track apart. One day, to no one's surprise, a software manufacturer came up with the idea of putting data in-between the normal DOS 3.3 tracks (called half-tracks). Still, data was placed one (1) track apart and, although data might now be placed on track 1.5, the next nearest track to contain data would be 2.5, then 3.5 etc. The reasoning behind all this madness is that the computer company named after a fruit (No, not banana) warns you that if you place your data less than a full DOS 3.3 track apart, you are asking for trouble. This means that if you have data on tracks 1 and 2, you cannot place data on tracks .5, 1.5 or 2.5 because the poor resolution of the read/write head will cause the data to interact with neighboring half-tracks.

Getting Around It

This data location restriction can be circumvented by using track arcing (see Diagram A) but, with this system, the full track cannot be used for data storage. Fortunately, although known to only a few people, there is another way around the full-track-apart restriction which revolves around the mysterious quarter-track.

Diagram A



Using quarter-tracks (those tracks in between half-tracks) does present some unique problems because the Apple drive is set up to move in half-track increments only. Subsequently, there is a certain amount of imprecision when dealing with quarter tracks. I will discuss later how to position the read/write head over them.

Terminology

A regular sequence of quarter-tracks would be 5, 5.25, 5.5, 5.75, 6 and 6.25. For the sake of argument, I will call 5.75 and 6.25 the adjacent quarter-tracks of track 6. You can expand on this terminology for adjacent half or three-quarter tracks.

A Bit Of Theory

To understand quarter-tracks, you must first accept two empirical laws first recorded in the far distant past by the most powerful Greek god of disk drives, Diskos (long "O" sound):

LAW 1: When a track is written, it will produce exact images of itself on the adjacent quarter-tracks, but data on the adjacent half-tracks will be questionable.

LAW 2: When a track is written, it will not affect data on the adjacent three-quarter tracks.

Note: The above two rules always hold when data is written on full or half-tracks, but they will sometimes fail when applied to quarter-tracks due to the imprecise disk head positioning. However, there are general guidelines that will reduce this inconsistency to a minimum.

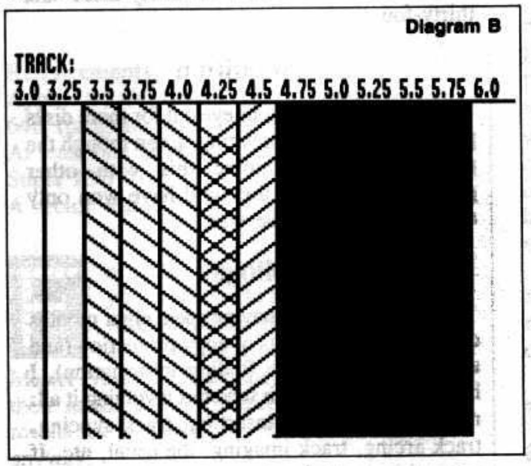
Putting It To The Test

The empirical suggests proof by observation; therefore, get out a nibble editor and we will devise an elegant proof. Use this nibble editor (I used LS 5.0) to examine the adjacent quarter-tracks of track 4.5 (tracks 4.25 and 4.75) of a valid DOS 3.3 disk. If you do a verify of those tracks, you will get images of tracks 4 and 5, respectively, with no errors. This then shows that data is indeed duplicated on adjacent quarter-tracks and that when track 5 was written, it did not zap the data three-quarters of a track away (on track 4.25).

In theory, as a result of this finding, data could be evenly spaced at .75 of a track throughout the entire disk (such a system would yield 46 distinct tracks, a 31% gain). Unfortunately, as stated earlier, the disk drive is not set up for precise positioning of the read/write head on quarter or three-quarter tracks. This is the reason that certain guidlines must be followed.

So What?

After interpreting the above information, I found that by using quarter-tracks, valid data can be written (but once, only) to adjacent half-tracks! As an example of how this can be accomplished, let's assume that you were to write out the first track on track 4, the next track on track 4.75 and another track on track 5.25. By the laws of Diskos, the three tracks would appear on tracks 4.0, 4.5 and 5.0, respectively. See Diagram B below.



But remember Rule 1: in this case, data appears too often to be read reliably, that is, track 4.5 tends to get a few zonks. To acheive the best resolution you must write in either of the following two patterns: 4.25, 5.25, 6.25 etc. or 4.75, 5.75, 6.75, etc. This also would produce a disk with valid images on the full and half-tracks (but they will be the same images that appear on the full tracks).

Copying

It follows that copying this disk by going from $0 to $22 would zap our image on track 4.5. If you "follow" me so far, you will realize that many of the current protection schemes could be applied to this strange track configuration. And, by the way, how do you think you are able to copy those Broderbund disks if the laws of Diskos don't exist? For a little more specific information see the notes on Electronic Arts at the end of this article.

The Demystifying

The ability to do quarter-tracks was probably developed to copy track arcing. Track arcing involves a DOS that interleaves data between half-tracks in such a way that about one-third of the track is written on one track and the next third of the track is written on the next half track and so on. As illustrated in Diagram A, this pattern ensures that valid data is not written so that it zaps the data on adjacent half-tracks.

Accordingly, by the tenets of Diskos, the quarter-tracks of a so encoded diskette would contain images of both the nearest full and half-track. This disk can then be copied by going from $.25 to $22.25 with a track step of one (1). As you might guess, the resulting disk could be subject to errors due to the fact that when you write a track like 1.25 you must not cause any glitches in track .5 which is only three-quarters of a track away. This is usually not a problem.

Depending on the protected disk, these tracks might also have to be written back out to disk in a certain format (in sync, preserving length, etc.). If you are saddened by the fact that track arcing can be copied, you could thwart potential pirates by using quarter-track increments for track arcing which I doubt is used at all (except maybe by the EDD III bootup).

Extraneous Data

With the ability to do quarter-tracks, a new protection scheme was born: the ability to check for the existence of the unblemished half-track. That is, a half track that contains nothing but the correct third of a track. A disk copied in the manner described above would have data from the adjacent half-tracks (in any half or full track) as well as the data it is supposed to have.

At this time I will coin a term for this new protection scheme, the newest line of defense for copy protectors: "TRACK IMAGING". I use this term to describe the protection as relying on the characteristics of how tracks affect each other when they are written. Even though it is just an off-shoot of track arcing, Track Imaging is a distinct type of copy protection. I hope it is now clear why a disk might copy by going from .25 to 34.25 instead of 0 to 34 and correspondingly, that the original disk in no way uses the quarter-track as a protection scheme. Instead: track arcing or track imaging.

Review Of Disk Access

The next part of this article is a review of how to access the disk drive through machine language.

If you have trouble following any of this article please refer to the Whiz Kid column in Hardcore COMPUTIST No's. 15 and 17. In these articles it was explained that DOS 3.3 divides the disk into 70 phases, with tracks being written on even phases and half-tracks on odd phases. Because four magnets control the móvement of the disk drive head, the head is moved by DOS 3.3 by turning these magnets on and off. The following is the way DOS 3.3 would move the head from track zero to track one after $B9A0 (SEEKABS) in DOS 3.3 has been called:

1) Turn on phase 1 (track .5) C083
2) Wait
3) Turn off phase 0 (track 0) C080
4) Wait
5) Turn on phase 2 (track 1) C085
6) Wait
7) Turn off phase 1 (track .5) C082
8) Wait
9) Turn off phase 2 (track 1) C084
10) Wait and return to caller

In this manner, to do quarter-tracks one must turn on the magnets, on either side of the quarter-track, wait, then turn them off. The following shows how to move from track 5.0 to track 5.25:

1) Turn on phase 2 (because there are only four phases, you must go 0, 1, 2, 3, 0, 1, 2, 3)

Most Wanted List

Need help backing-up a particularly stubborn program?

Send us the name of the program and its manufacturer and we'll add it to our Most Wanted List, a column (updated each issue) which helps to keep Hardcore COMPUTIST readers informed of the programs for which softkeys are MOST needed. Send your requests to:

Hardcore COMPUTIST
Wanted List
PO Box 110846-K
Tacoma, WA 98411

If you know how to deprotect unlock, or modify any of the programs below,

let us know. You'll be helping your fellow Hardcore COMPUTIST readers and earning MONEY at the same time. Send the information to us in article form on a DOS 3.3 diskette.

1. Apple Business Graphics
Apple Computer

2. Flight Simulator II
Sub Logic

3. Apple LOGO II
Apple Computer

4. Jane
Arktronics

5. Bookends
Sensible Software

6. Visiblend
Microlab

7. Sundog
FTL Games

8. Lifesaver
Microlab

9. Catalyst
Quark, Inc.

10. Gutenburg Jr. & Sr.
Micromation LTD

11. Prime Plotter
Primesoft Corp.

12. The Newsroom
Springboard Software

13. Robot Odyssey
The Learning Company

14. Zardax
Computer Solutions

15. The Handlers
Silicon Valley Systems

16. Milliken Math Series (NEW)
Milliken Publishing

17. The Apple's Core: Parts 1-3
The Professor

18. Where in the World is Carmen San Diego?
Broderbund

19. Karateka
Broderbund

20. King's Quest
Sierra On-line

21. Hayes Terminal Program
Hayes

22. Fun Bunch
Unicorn

23. Murder on the Zinder neuf
Electronic Arts

2) Turn on phase 3 (track 5.5)
3) Wait
4) Turn off phase 3 (track 5.5)
5) Turn off phase 2 (track 5.Ø)
6) Return

If you missed those two articles, you might need to know how to turn these magnets on and off. All you need to do is address (read from or write to) the pertinent memory locations, which are given in the table below:

Addr.	Label	Function
$CØnØ	MAGØ.OFF	Stepper magnet Ø off
$CØn1	MAGØ.ON	Stepper magnet Ø on
$CØn2	MAG1.OFF	Stepper magnet 1 off
$CØn3	MAG1.ON	Stepper magnet 1 on
$CØn4	MAG2.OFF	Stepper magnet 2 off
$CØn5	MAG2.ON	Stepper magnet 2 on
$CØn6	MAG3.OFF	Stepper magnet 3 off
$CØn7	MAG3.ON	Stepper magnet 3 on
$CØn8	MOTOR.OFF	Drive and motor off
$CØn9	MOTOR.ON	Drive and motor on
$CØnA	SEL.DRV1	Route power to drive 1
$CØnB	SEL.DRV2	Route power to drive 2
$CØnC	IO1	Strobe latch for I/O
$CØnD	IO2	Load data latch
$CØnE	IO3	Prepare latch for input
$CØnF	IO4	Prepare latch for output

Note: In the above table, you must replace "n" with the slot number of the drive to be accessed plus eight (8). For example, to turn on drive 1 in slot 6, you would access $CØE9.

If you really, really have trouble, consult BENEATH APPLE DOS from Quality Software.

Here are a few other locations you may need to know:

```
$B7EA: Current drive in use can be 1 or 2
$47E : Current location of D1 (in slot 6)
       counted in half-tracks. Not updated
       by seek operation
$4FE : Current location of D2 (in slot 6)
       counted in half-tracks. Not updated
       by seek operation.
```

Electronic Arts

I have saved the last part of this article to actually examine a disk that has been protected with track imaging. The most infamous are programs from Electronic Arts. I'll use Pinball Construction Set as an example. First, let me say that when I examined both the boot of EA programs and the code itself, I couldn't find a nibble count being used (self-modifying, really?).

On PCS, this is the way the protection works. First, track 34 is sought. The head then positions itself to 33.5, 33, 33.5 and then 34. The program is checking for two things: 1) that the sectors of data it finds are glitch free (track imaging), and 2) that the sectors found match those sought (track syncing). If you examine the disk and run a sync test, you will find that tracks 33, 33.5, and 34 are exact images of each other (they are all track 33). On a more intensive examination one finds tracks 32.75, 33, 33.25, 33.5, 33.75, and 34 are exact images of each other. The only way this can be done is by placing one head at 33 and one at 33.75 and then feeding both heads with the same signals while formatting. Subsequently, a very precise sync can be obtained along with the check of track imaging. Needless to say, EA disks are formatted with very precise machines and are extremely difficult to duplicate with the Apple drive. On newer disks (Skyfox, Archon,etc.), the protected tracks have changed to five and six instead of thirty-three and thirty-four.

Admonition

I urge you to keep your eye out for more disks protected with track imaging. Even though the protectors may combine this with other protection schemes, they will have won only another battle, not the war.

Epilogue

A final note. After having spent eight months delving into all aspects of copy protection (and at times it was tough finding information), I have written protection schemes involving it all: nibble counting, bit insertion, track syncing, track arcing, track imaging, the usual, etc. If you enjoy the self-torture of deciphering copy protection or would simply like to discuss the topics in this article or topics in general, drop me a line at the address below:

Bruce Jones
P.O. Box 11111
Pueblo, CO 81ØØ1

APT's

Defender
Atarisoft

Use the softkey for Zork I in Hardcore COMPUTIST No. 1 to obtain a deprotected Defender and then, with this APT, you can give yourself 255 men and smartbombs.

BLOAD DEFENDER
CALL -151

Type

43C8:FF

where (FF=255 men/smartbombs)

4ØØØG

(runs game)

Contributed by Mad Max & Opus

Deprotecting

DB Master version 4+

► By Clay Harrell

Requirements:
Apple //e, //c, or][+ with at least 64K (program requirement)
At least one DOS 3.3 disk drive
Super IOB v1.2
A sector editor

For all the DB Master users out there, you'll be glad to know that when the folks at Stoneware provided us with a new version of the infamous DB Master (version 4 Plus), they also changed from their standard copy protection to boot! Sure it means extra work, but that's half the fun anyway!

Stoneware has been famous for their use of half-tracks on all previous versions of DB Master and on their new product, Business Writer. Normally, the protection used is very good and difficult to deprotect. But on this new version, Stoneware has unpredictably changed the protection from half-tracks to all full-tracks.

In my quest to discover the means to deprotect the disk, my first step was to boot normal DOS 3.3 and try copying the disk with COPYA. As expected, COPYA refused to cooperate. My next step was to defeat every error DOS came up with (reading or writing) by putting a CLC intruction ($18) at memory location $B942. This in effect allows COPYA to copy any disks protected by a change in the epilogue bytes. But, if the prologue bytes are altered, you will not get an error and the copy you make with COPYA will be bad. In short, the "B942:18" method, although used frequently, is a somewhat sloppy way to copy disks.

Back to epilogue bytes: You should remember that the epilogue bytes are on every sector of a disk and tell DOS that the "buck stops here". After DOS reads the appropriate data from the disk it expects a two byte sequence ($DE AA) for reassurance that it has read the correct data. If these bytes are not $DE AA, DOS thinks that an error has occurred (unless the error routine is defeated) during the read. Of course it is standard copy protection to change these bytes from the normal $DE AA sequence.

Now both sides of DB Master 4 Plus copy fine, but don't expect them to work yet... we still have to tell DB Master's operating system not to expect the perverted epilogue bytes (since we just converted them to normal DOS format). Because DB Master has a relatively normal DOS image on tracks $0-$2, this is done fairly easily. Once this is done, DB Master will run like a champ... almost! You can create a new input form, but if you try to use your newly created form, the program hangs. Of course, there had to be some secondary protection!

The reason the program hangs is simple: Stoneware has a couple of routines that modify their DOS so that it can read normal disks as well as the DB master disk.

The routine that is partially responsible starts like this:

```
Ø9FD-   AD E9 B7    LDA  $B7E9
ØAØØ-   8D Ø8 ØB    STA  $ØBØ8
ØAØ3-   8D FA ØA    STA  $ØAFA
ØAØ6-   EE 35 B9    INC  $B935
ØAØ9-   EE 91 B9    INC  $B991
ØAØC-   EE 9B B9    INC  $B99B
ØAØF-   EE 3F B9    INC  $B93F
```

Notice that the code increments the values at locations $B935, $B991, $B99B and $B93F. These locations hold the values of the epilogue bytes in DOS, so this routine changes the epilogue bytes from their normal value of $DE AA to $DF AB (upon close examination of the DB master original disk, I found that these were indeed the address and data trailer bytes!). Then, as expected, at $AE4 the routine decrements the same locations in DOS (the epilogue bytes) back to their previous values.

To defeat this routine, disable the "Branch if Not Equal" instruction at $A77 and at $A9A and NOP the increments and decrements of the epilogue bytes (just for assurance). I used a disk search utility to find the above code on track $4, sector $C, side 1 of DB Master version 4 Plus.

In addition, there was some similar code on side 2, on track $12, sectors $C and $D. I defeated this code by NOPing it (replacing the code with NO oPerating instruction).

Finally, I was rewarded with a fully functional deprotected version of the program.

Step-By-Step

1) Type in the controller at the end of this article and save it

SAVE CON.DB MASTER 4+

2) Install the controller into Super IOB and execute it

RUN

3) Answer the questions and make copies for both sides of DB master 4+.

4) Reboot normal DOS 3.3 and run your favorite sector editor and make the following changes to the COPYA copies of DB Master version 4 Plus:

```
Side 1, trk 4, sector $C, bytes $Ø7-$12
 to EA EA EA EA EA EA EA EA EA EA EA EA
 (12 EA's)
Side 1, trk 4, sector $C, bytes $78-79
  from $DØ 25 to $EA EA
Side 1, trk 4, sector $C, bytes $9B-9D
  from $C9 2Ø 9Ø to $A9 ØØ FØ
Side 1, trk 4, sector $C, bytes $E5-FØ
 to EA EA EA EA EA EA EA EA EA EA EA EA
 (12 EA's)
Side 2, trk $12, sct $C, bytes $A5-BC
  to 23 EA's
Side 2, trk $12, sct $C, bytes $CØ-D7
  to 23 EA's
Side 2, trk $12, sct $D, bytes $A5-BC
  to 23 EA's
Side 2, trk $12, sct $D, bytes $CØ-D7
  to 23 EA's
```

And you're all done!

DB Master 4+ Controller

```
1ØØØ REM DB MASTER 4+
1Ø1Ø TK = Ø :ST = Ø :LT = 35 :CD = WR
1Ø2Ø T1 = TK : GOSUB 49Ø : RESTORE : GOSUB 17Ø
1Ø3Ø GOSUB 43Ø : GOSUB 1ØØ :ST = ST + 1 : IF ST <
    DOS THEN 1Ø3Ø
1Ø4Ø IF BF THEN 1Ø6Ø
1Ø5Ø ST = Ø :TK = TK + 1 : IF TK < LT THEN 1Ø3Ø
1Ø6Ø GOSUB 23Ø : GOSUB 31Ø : GOSUB 49Ø :TK = T1
    :ST = Ø
1Ø7Ø GOSUB 43Ø : GOSUB 1ØØ :ST = ST + 1 : IF ST <
    DOS THEN 1Ø7Ø
1Ø8Ø ST = Ø :TK = TK + 1 : IF BF = Ø AND TK < LT THEN
    1Ø7Ø
1Ø9Ø IF TK < LT THEN 1Ø2Ø
11ØØ HOME : PRINT "DONE^WITH^COPY' : END
5ØØØ DATA 223 ,171 ,223 ,171
5Ø1Ø DATA 4^CHANGES ,Ø ,3 ,53 ,222 ,Ø ,3 ,63 ,17Ø
5Ø2Ø DATA Ø ,3 ,145 ,222 ,Ø ,3 ,155 ,17Ø
```

Controller Checksums

1ØØØ	-	$356B	1Ø7Ø	-	$5CAD
1Ø1Ø	-	$3266	1Ø8Ø	-	$Ø4Ø6
1Ø2Ø	-	$7FØA	1Ø9Ø	-	$7922
1Ø3Ø	-	$69ØB	11ØØ	-	$65C7
1Ø4Ø	-	$51E4	5ØØØ	-	$5F7D
1Ø5Ø	-	$FDC1	5Ø1Ø	-	$41FF
1Ø6Ø	-	$54A8	5Ø2Ø	-	$DC93

ProSHADOW: A ProDOS Disk Monitor

By Doni G. Grande

Requirements:
64K Apple with ProDOS

Hardcore COMPUTIST No. 12 contained an article and program called The Armonitor. This program 'shadowed' DOS, as it accessed a disk, showing exactly which track and sector was being accessed and what operation was being done by the little DOS demon, RWTS. This is very important information when trying to determine what the disk arm is doing as it chugs back and forth across the disk.

Unfortunately, there is a major drawback: most protected software uses a custom DOS which is somewhat difficult to patch using the Armonitor.

All of a sudden, along comes ProDOS, an entirely new and mostly uncharted DOS. Utilities such as The Armonitor have been rendered useless because the inner workings of ProDOS are entirely different from those of DOS 3.3.

But look at the wonderful features ProDOS provides the ability to: nest directories, use mass storage devices (hard disks, etc.), use a built in RAM disk (If you've got 128K, just use the volume name /RAM to access a RAM disk about half the size of a normal floppy! But remember: When you turn off the machine, this 'floppy' goes away.), achieve faster disk operations and more. And software vendors now offer a multitude of new programs which use the new ProDOS. One especially nice feature of ProDOS is that it has a standard means to access disk drives and other storage devices. If a software publisher wants to make his software useful with hard disks, he must use plain vanilla ProDOS to access the disk.

The major protection scheme under ProDOS is a nibble counting routine on the original disk but, even so, many programs use ProDOS to position the disk arm to the correct track (Sensible Speller, for example) and then use their own routine to read the nibble count. This makes a program that shadows the disk arm very useful!

Enter ProShadow.

ProShadow 'shadows' ProDOS, reporting on what the RWTS is doing to the disk. ProShadow uses only sixteen bytes of memory within ProDOS (normally reserved for Apple's copyright message). The rest of ProShadow resides within the alternate 4K bank of the RAM card not used by ProDOS. Everytime ProDOS wants to use the disk drives, ProShadow intercepts the message and does these things before passing it on: saves the bottom line of the text screen and reports something similar to the following:

```
CMD=01   S=6   D=1   BUF=1000   BLK=0010
```

This reveals that ProDOS is reading (CMD=1) from slot six, drive one (S=6 D=1) into memory at $1000 (BUF=1000) the data from disk block $10 (BLK=0010). ProShadow then passes command to the normal disk routines. When the disk routines complete their task, command passes back to ProShadow which restores the bottom line of the text screen and exits back to ProDOS. The net effect is that neither ProDOS nor the calling program have any idea that things have gone astray!

Typing It In

First boot up ProDOS and get into BASIC. Now enter the monitor and type the following:

ProShadow Hexdump

```
4000: 2C 81 C0 2C 81 C0 AD 1C   $738A
4008: BF 8D 90 2F AD 1D BF 8D   $91A4
4010: 91 2F A0 E8 B9 38 40 99   $5D71
4018: FF CF 88 D0 F7 A0 0F B9   $885B
4020: D8 40 99 40 BF 88 10 F7   $EE64
4028: A9 40 8D 1C BF 8D 2C BF   $5D08
4030: A9 BF 8D 1D BF 8D 2D BF   $D840
4038: 60 2C 83 C0 AD 53 C0 AD   $FD4D
4040: 54 C0 20 6E D0 A0 04 A5   $1E80
4048: 42 20 82 D0 A0 0B A5 43   $A8ED

4050: 4A 4A 4A 4A 29 07 20 91   $1203
4058: D0 A0 11 A5 43 2A 2A 29   $FCE6
4060: 01 18 69 01 20 91 D0 A0   $8204
4068: 19 A5 45 20 82 D0 A5 44   $F6C3
4070: 20 82 D0 A0 24 A5 47 20   $2C3D
4078: 82 D0 A5 46 20 82 D0 A0   $BBEE
4080: 0F B9 50 BF 99 40 BF 88   $53D4
4088: 10 F7 4C 40 BF 48 98 48   $B3E4
4090: 2C 83 C0 20 6E D0 A0 0F   $E050
4098: B9 9F D0 99 40 BF 88 10   $D199

40A0: F7 68 A8 68 4C 47 BF A0   $3492
40A8: 27 B9 D0 07 48 B9 BF D0   $3D63
40B0: 99 D0 07 68 99 BF D0 88   $11AD
40B8: 10 EF 60 48 4A 4A 4A 4A   $523A
40C0: 20 91 D0 68 29 0F 20 91   $4350
40C8: D0 60 09 B0 C9 BA 90 03   $B258
40D0: 18 69 07 99 D0 07 C8 60   $92FC
40D8: D8 2C 83 C0 4C 00 D0 2C   $576B
40E0: 8B C0 2C 8B C0 28 60 EA   $219E
40E8: 2C 8B C0 2C 8B C0 20 00   $830A
```

ProShadow Source Code

```
*----------------------------------------------------------------------*
*                                                                      *
*                            ProShadow                                 *
*                        ProDOS Disk Monitor                           *
*                                                                      *
*                               By                                     *
*                         Doni G. Grande                               *
*                                                                      *
*----------------------------------------------------------------------*

*----------------------------------------------------------------------*
*                   RAM card bank switch locations                     *
*                 Two accesses to the same location                    *
*                        write enables RAM.                            *
*----------------------------------------------------------------------*

C08B- RAM1RD   .EQ $C08B   Read bank 1
C083- RAM2RD   .EQ $C083   Read bank 2
C081- RAM2WRT  .EQ $C081   Write bank 2 w/o read

*----------------------------------------------------------------------*
*                   Soft switches for text screen                      *
*----------------------------------------------------------------------*

C053- MIXED    .EQ $C053   Mixed graphics/text
C054- PAGE1    .EQ $C054   Page one

*----------------------------------------------------------------------*
*                  Patch location for bank switching                   *
*                 routines. Normally used for ProDOS                   *
*                         copyright message.                           *
*----------------------------------------------------------------------*

BF40- PATCHLOC .EQ $BF40

*----------------------------------------------------------------------*
*                  Last line on screen starts at $7D0                  *
*----------------------------------------------------------------------*

07D0- SCREEN   .EQ $7D0

*----------------------------------------------------------------------*
*                   Disk Device Driver Parameters                      *
*                                                                      *
*                COMMAND     =  $00 for status                         *
*                               $01 for read                           *
*                               $02 for write                          *
```

Continued on next page

```
40F0: D0 08 2C 83 C0 4C 54 D0   $58DD
40F8: C3 CD C4 BD A0 A0 A0 A0   $3312
4100: A0 D3 BD A0 A0 A0 A0 C4   $BA82
4108: BD A0 A0 A0 A0 C2 D5 C6   $CA40
4110: BD A0 A0 A0 A0 A0 A0 A0   $8491
4118: C2 CC CB BD A0 A0 A0 A0   $6B16
```

Now save ProShadow to the disk with:

BSAVE PROSHADOW,A$4000,L$0120

Using ProShadow

Using ProShadow is quite simple: BRUN PROSHADOW. This will install ProShadow into the RAM card and install the patches required to be used with any version of ProDOS. (Note: Do not BRUN PROSHADOW if ProShadow is already installed! Since ProShadow patches itself using the contents of certain ProDOS pointers and then points these pointers to itself, an infinite loop will be generated if ProShadow is installed twice!) To access some 'protected' program, first set the prefix to the name of the disk that the protected program is on (using PREFIX /name command) and then execute the program's starting SYS file (this is always the first pathname.SYSTEM file on the disk) by using the 'dash' command:

-pathname.SYSTEM

The program will usually start running, and you will be able to watch what is happening to the disk!

Inner Workings: Where ProShadow Gets The Data

ProShadow easily takes advantage of the great flexibility of ProDOS (ProDOS provides a standard method to interface to storage devices).

First you must write a machine language device driver to control the device (such as a disk drive). ProDOS already has device drivers built-in for the Disk II, a /RAM disk, and a Thunderclock. Once that has been done, you must notify ProDOS that this device exists. This is done by placing the address of the device driver and the total number of drivers present in a special part of memory- the ProDOS System Global Page (PSGP). Apple guarantees that the PSGP will stay in the same place in memory ($BF00-$BFFF) and that certain areas within the PSGP will always be used for certain purposes.

The area we are interested in is as follows:

Address range	Used for
BF10.BF2F	Device Driver address table
BF10.BF11	Slot 0 - reserved
BF12.BF13	Slot 1, Drive 1
BF14.BF15	Slot 2, Drive 1
BF16.BF17	Slot 3, Drive 1
BF18.BF19	Slot 4, Drive 1
BF1A.BF1B	Slot 5, Drive 1
BF1C.BF1D	Slot 6, Drive 1
BF1E.BF1F	Slot 7, Drive 1
BF20.BF21	Slot 0 - reserved
BF22.BF23	Slot 1, Drive 2
BF24.BF25	Slot 2, Drive 2
BF26.BF27	Slot 3, Drive 2
BF28.BF29	Slot 4, Drive 2
BF2A.BF2B	Slot 5, Drive 2
BF2C.BF2D	Slot 6, Drive 2
BF2E.BF2F	Slot 7, Drive 2
BF30	Slot/Drive of last device
BF31	Count-1 of active devices
BF32.BF3F	List of active devices (ID)
BF40.BF4F	Copyright Notice

Because we are only interested in changing a device driver instead of adding a new one, I won't go over how a new driver (different slot and drive) is inserted into the table. If you are interested, see 'Beneath Apple ProDOS' by Worth and Lechner, published by Quality Software.

Notice that ProDOS must have the address of the Disk II device driver in the table at the entries for slot six, drives one and two. These are in the table at $BF1C-1D and $BF2C-2D and both point to the same place: $F8ØØ for ProDOS versions 1.Ø.x and $DØØØ for versions 1.1.x. Because both point to the same place, ProDOS must have to tell the Disk II device driver which drive to use as well as the desired command (read, write, seek, format), the block number to read, and the address of the block buffer to store the data read. This data is passed to any device driver through the following zero page locations:

```
Addr.                Description
-----------------------------------------------
$42   Command Code
          $ØØ = status request
          $Ø1 = read
          $Ø2 = write
          $Ø3 = format (not used by Disk II)
$43   Unit number - which slot and drive
                    to use in the format
                    DSSSØØØØ
        D = drive # (Ø=D1, 1=D2)
        SSS = slot # Ø-7
$44   Buffer for I/O - memory address of
                    data or where to
                    store data after
                    read
$45   Block Number - device block to
                    use for I/O
```

Upon exit, the carry flag is set by the device driver if an error occurred and the accumulator contains the return code:

```
A-reg      Meaning
-----------------------------------------------
$ØØ        No error
$27        I/O error
$28        No device connected
$2B        Write protect error
```

Also, upon return, some device drivers return the total number of blocks available in the X-register (low byte) and the Y-register (high byte). Since this information is just left in zero page, it is a rather simple matter to write a program that:

1) Replaces the normal Disk II device driver address for slot 6, drives 1 and 2 with its own address
2) Prints the device driver parameters whenever slot 6, drive 1 or 2 is accessed
3) Calls the original device driver
4) Covers its tracks (no pun intended)
5) Returns to the ProDOS calling routine

This is exactly what ProShadow does.

Where to Put The Program

The hard part of writing ProShadow was trying to find a place to put it (ProShadow uses slightly less than one page (256 bytes) of memory). Page 3 ($3ØØ-$3FF) is one of the few free pages in the Apple's memory; however, everyone and his brother uses page 3 for something. Another possible candidate is some memory within ProDOS. Unfortunately, this is

Continued from previous page

```
*                     $Ø3 for format                          *
*                 UNIT  =  DSSSØØØØ                           *
*                D is drive # Ø=1, 1=2                        *
*                  SSS is slot # Ø-7                          *
*         BUFFER  =  address of block buffer                  *
*             BLOCK  =  block # for I/O                       *
*      On return, carry flag is set if error                  *
*             and error # is in A-reg:                        *
*            A-reg  =  $ØØ for no error                       *
*                      $27 for I/O error                      *
*                      $28 for no device connected            *
*                      $2B for write protect error            *
*-------------------------------------------------------------*

ØØ42- COMMAND  .EQ $42
ØØ43- UNIT     .EQ $43
ØØ44- BUFFER   .EQ $44
ØØ46- BLOCK    .EQ $46

*-------------------------------------------------------------*
*           Disk driver starts at $DØØØ in V1.1.1             *
*          The program automatically points to the            *
*                 correct location when run.                  *
*-------------------------------------------------------------*

DØØØ- DRIVER   .EQ $DØØØ

*-------------------------------------------------------------*
*         Disk driver pointers in ProDOS Global Page          *
*-------------------------------------------------------------*

BF1C- DEVADR61 .EQ $BF1C    Slot 6, drive 1
BF2C- DEVADR62 .EQ $BF2C    Slot 6, drive 2

*-------------------------------------------------------------*
*           PROSHADOW will be installed in the RAM            *
*            card at $DØØØ in bank 2. This area is            *
*            not used by ProDOS. However, a patch             *
*            will have to be made at $BF4Ø to jump            *
*          to this code since ProDOS normally uses            *
*           bank 1 memory. $BF4Ø normally contains            *
*            the copyright message for ProDOS.                *
*-------------------------------------------------------------*

                   OR $4ØØØ
         START
4ØØØ: 2C 81 CØ              BIT RAM2WRT  Enable bank2 for write
4ØØ3: 2C 81 CØ              BIT RAM2WRT  but not for read

*-------------------------------------------------------------*
*             Patch PATCH2 to point to correct                *
*             disk driver location. This is in                *
*            case driver location changes in future           *
*           ProDOS versions. For previous versions:           *
*                Version      Driver Address                  *
*                 1.Ø.1           $F8ØØ                       *
*                 1.Ø.2           $F8ØØ                       *
*                 1.1.1           $DØØØ                       *
*-------------------------------------------------------------*

4ØØ6: AD 1C BF              LDA DEVADR61
4ØØ9: 8D 9Ø 2F              STA REALSTRT+PATCH2+JSRDRVER-PATCHLOC+1-BEGIN
4ØØC: AD 1D BF              LDA DEVADR61+1
4ØØF: 8D 91 2F              STA REALSTRT+PATCH2+JSRDRVER-PATCHLOC+2-BEGIN
```

```
*----------------------------------------------------------*
*          Move PROSHADOW to RAM bank 2 at $DØØØ           *
*----------------------------------------------------------*

4Ø12: AØ E8           LDY #END-BEGIN+1 Number of bytes to move
      MOVELOOP
4Ø14: B9 38 4Ø        LDA REALSTRT-1,Y Get byte to move
4Ø17: 99 FF CF        STA BEGIN-1,Y and move it
4Ø1A: 88              DEY
4Ø1B: DØ F7           BNE MOVELOOP Loop until done

*----------------------------------------------------------*
*              Now move PATCH1 to PATCHLOC                 *
*----------------------------------------------------------*

4Ø1D: AØ ØF           LDY #$ØF     Length of PATCH1
      MOVE2LP
4Ø1F: B9 D8 4Ø        LDA REALSTRT+PATCH1-BEGIN,Y Get byte to move
4Ø22: 99 4Ø BF        STA PATCHLOC,Y and move it
4Ø25: 88              DEY
4Ø26: 1Ø F7           BPL MOVE2LP  Loop until done

*----------------------------------------------------------*
*        Patch device pointers to point to this driver     *
*----------------------------------------------------------*

4Ø28: A9 4Ø           LDA #PATCHLOC Get lo byte
4Ø2A: 8D 1C BF        STA DEVADR61 atch 6,1
4Ø2D: 8D 2C BF        STA DEVADR62 Patch 6,2
4Ø3Ø: A9 BF           LDA /PATCHLOC Get hi byte
4Ø32: 8D 1D BF        STA DEVADR61+1
4Ø35: 8D 2D BF        STA DEVADR62+1

*----------------------------------------------------------*
*                    Done, so return                       *
*----------------------------------------------------------*

4Ø38: 6Ø              RTS

*----------------------------------------------------------*
*                 Start routine at $DØØØ                   *
*----------------------------------------------------------*

4Ø39- REALSTRT .EQ *
               .OR $DØØØ
               .TA REALSTRT

*----------------------------------------------------------*
*          Start out by write enableing RAM                *
*            bank 2. Only one access to                    *
*            RAM2RD is required because                    *
*           calling routine has already                    *
*                 accessed it once.                        *
*----------------------------------------------------------*

      BEGIN
DØØØ: 2C 83 CØ        BIT RAM2RD

*----------------------------------------------------------*
*               Turn on bottom line of text.               *
*----------------------------------------------------------*

DØØ3: AD 53 CØ        LDA MIXED
DØØ6: AD 54 CØ        LDA PAGE1
```

Continued on next page

not possible since Apple continues to modify ProDOS (notice that there have already been several 1.Ø.x and 1.1.x versions). A third possibility is to use the space between BASIC.SYSTEM and Applesoft HIMEM, but this will work only if the utility is to be used with BASIC.

After investigating each of these possibilities, I finally found a bit of memory that remains unused by ProDOS. ProDOS resides in the RAM card from $DØØØ-$FFFF (bank 1) and $D1ØØ-$D??? (bank 2). However, $DØØØ-$DØFF is not used by any version of ProDOS (yet?); therefore, a short utility like ProShadow can reside at $DØØØ in the RAM card bank 2. The major disadvantage to this is that whenever ProDOS is active, bank 1 of the RAM card is also active. This means that ProShadow must be called (in bank 2) from ProDOS (in bank 1).

I discovered that I could use the sixteen bytes that Apple reserves for a copyright notice at $BF4Ø-BF4F as a 'switchbox'. The following patch is placed there when ProShadow is installed:

```
PATCH1 CLD        ;All device drivers must
                   begin w/CLD
       BIT $CØ83 ;Turn on bank 2
       JMP BEGIN ;Jump to beginning of
                   ProShadow

EXIT              ;ProShadow exit routine
       BIT $CØ8B ;Turn on bank 1
       BIT $CØ8B ;twice write enables
       PLP        ;Get saved P-register
       RTS        ;Return to ProDOS caller
       NOP        ;A filler byte
```

Note that this routine is exactly sixteen bytes long. When ProShadow is installed, this routine is placed at $BF4Ø and the device driver addresses for slot 6 drives 1 and 2 are patched to point to $BF4Ø. There is now only one other minor (?) problem. For ProShadow to call the original device driver, it must first call a routine in RAM card bank 1. (In ProDOS versions 1.Ø.x the Disk II device driver is at $F8ØØ and this then is not entirely true. However, the device driver might try to access the $DØØØ-$DFFF area of memory so bank 1 must be turned on anyway). This necessitates another 'switching' routine. ProShadow installs the following switching routine at $BF4Ø so that the real device driver may be safely called:

```
PATCH2 BIT $CØ8B ;Turn on bank 1
       BIT $CØ8B ;twice to write enable
       JSR Driver;Call the device driver
       PHP        ;Save P-reg so BIT
                   doesn't disturb
       BIT $CØ83 ;Turn on bank 2
       JMP Back  ;Jump back to ProShadow
```

Note that this is again exactly sixteen bytes long. The 'JSR Driver' instruction is patched to point to the original Disk II device driver address (so thoughtfully provided by ProDOS) when ProShadow is installed, so this will work with any version of ProDOS!

The Mechanics of Operation

When ProShadow is run, the following occurs (addresses in parenthesis reference the actual program code):

1) The RAM card bank 2 is enabled for writing but not for reading ($4000-$4005).
2) PATCH2 described above is patched to do a JSR to the device driver installed for slot 6 drive 1 (drive 2 should be the same for the Apple Disk II) ($4006-$4011).
3) ProShadow is moved to the RAM card bank 2 at $DØØØ ($4012-$401C).
4) PATCH1, the ProShadow switching routine described above, is moved to $BF40-$BF4F ($401D-$4027).
5) The device driver addresses for slot 6, drives 1 and 2 are pointed to $BF40 ($4028-4037).
6) Control is returned to the caller ($4038).

ProShadow In Operation

When the device driver is called, ProShadow intercepts the call and does the following:

1) Write enables bank 2 (this was not done in the switching routine for lack of space) ($DØØØ-$DØØ2).
2) Turns on the text screen. This is optional and can be replaced with NOP's ($EA) if desired. Some programs make you look at graphics while they do the dirty work, and this just frustrates them ($DØØ3-$DØØ8).
3) Swaps the bottom line of the text screen with the screen mask for ProShadow. A swap subroutine ($DØ6E-$DØ81) is used for this ($DØØ9-$DØØB).
4) Prints the command number, slot number, drive number, buffer location and block number on the screen- a special hexadecimal printing routine ($DØ82-$DØ9Ø) since the monitor cannot be accessed from within ProDOS ($DØØC-$DØ45).
5) Installs PATCH2 (described earlier) to call the Disk II device driver ($DØ46-$DØ5Ø).
6) Calls the Disk II device driver via PATCH2 at $BF40 ($DØ51-$DØ53).
7) Saves all registers and write enables the RAM bank 2 ($DØ54-$DØ59).
8) Restores the last line of the text screen by swapping with the screen mask ($DØ5A-$DØ5C).
9) Re-installs PATCH1 at $BF40 to enable ProShadow to exit and be called next time ($DØ5D-$DØ67).
1Ø) Restores all registers ($DØ68-$DØ6A).
11) ProShadow exits via PATCH1 exit routine ($DØ6B-$DØ6D).

Possible Modifications

One modification mentioned above is to not allow ProShadow to display the text screen everytime it is entered. To do this, make the following modification to the original code:

4Ø3C: EA EA EA EA EA EA
(was AD 53 CØ AD 54 CØ)

This merely replaces the code which turns on the text screen with NOP's.

Continued from previous page

```
*------------------------------------------------------------------*
*            Preserve text screen by swapping last                 *
*          line of screen with mask for bottom line.               *
*------------------------------------------------------------------*

DØØ9: 2Ø 6E DØ          JSR SWAPEM

*------------------------------------------------------------------*
*                      Print COMMAND number                        *
*------------------------------------------------------------------*

DØØC: AØ Ø4             LDY #$Ø4
DØØE: A5 42             LDA COMMAND
DØ1Ø: 2Ø 82 DØ          JSR PRINTHEX

*------------------------------------------------------------------*
*                        Print slot number                         *
*------------------------------------------------------------------*

DØ13: AØ ØB             LDY #$ØB
DØ15: A5 43             LDA UNIT     Form DSSSØØØØ
DØ17: 4A                LSR          ;ØDSSSØØØ
DØ18: 4A                LSR          ;ØØDSSSØØ
DØ19: 4A                LSR          ;ØØØDSSSØ
DØ1A: 4A                LSR          ;ØØØØDSSS
DØ1B: 29 Ø7             AND #$Ø7     ØØØØØSSS
DØ1D: 2Ø 91 DØ          JSR DIGITOUT

*------------------------------------------------------------------*
*                       Print drive number                         *
*------------------------------------------------------------------*

DØ2Ø: AØ 11             LDY #$11
DØ22: A5 43             LDA UNIT
DØ24: 2A                ROL              ;D--SSSØØØØØ
DØ25: 2A                ROL              ;S--SSØØØØØD
DØ26: 29 Ø1             AND #$Ø1     ØØØØØØØD
DØ28: 18                CLC
DØ29: 69 Ø1             ADC #$Ø1     Make 1-2
DØ2B: 2Ø 91 DØ          JSR DIGITOUT

*------------------------------------------------------------------*
*                      Print BUFFER location                       *
*------------------------------------------------------------------*

DØ2E: AØ 19             LDY #$19
DØ3Ø: A5 45             LDA BUFFER+1 Get hi byte
DØ32: 2Ø 82 DØ          JSR PRINTHEX
DØ35: A5 44             LDA BUFFER   Get lo byte
DØ37: 2Ø 82 DØ          JSR PRINTHEX

*------------------------------------------------------------------*
*                       Print BLOCK number                         *
*------------------------------------------------------------------*

DØ3A: AØ 24             LDY #$24
DØ3C: A5 47             LDA BLOCK+1  Get hi byte
DØ3E: 2Ø 82 DØ          JSR PRINTHEX
DØ41: A5 46             LDA BLOCK    Get lo byte
DØ43: 2Ø 82 DØ          JSR PRINTHEX

*------------------------------------------------------------------*
*             Install PATCH2 so that disk driver                   *
*                       may be called.                             *
*------------------------------------------------------------------*
```

```
DØ46: AØ ØF            LDY #$ØF      Length of patch
     PATCH2LP
DØ48: B9 5Ø BF         LDA PATCH2,Y Get patch byte
DØ4B: 99 4Ø BF         STA PATCHLOC,Y put in patch area
DØ4E: 88               DEY
DØ4F: 1Ø F7            BPL PATCH2LP Continue until done
*-----------------------------------------------------------*
*                   Now call DISK II driver                 *
*                      as a subroutine                      *
*-----------------------------------------------------------*

DØ51: 4C 4Ø BF         JMP PATCHLOC

*-----------------------------------------------------------*
*                 Back from driver, so restore              *
*                 bottom screen line. However,              *
*              processor registers do contain info          *
*            this time around, so must preserve those       *
*               that are used. Note that the P-reg          *
*                was saved by the bank switching            *
*             routine and will be restored on exit.         *
*-----------------------------------------------------------*

     CLEANUP
DØ54: 48               PHA          ;Save A-reg
DØ55: 98               TYA          ;Save Y-reg
DØ56: 48               PHA

*-----------------------------------------------------------*
*                  Write enable RAM bank 2                  *
*-----------------------------------------------------------*

DØ57: 2C 83 CØ         BIT RAM2RD

*-----------------------------------------------------------*
*                Now swap back bottom screen line           *
*-----------------------------------------------------------*

DØ5A: 2Ø 6E DØ         JSR SWAPEM

*-----------------------------------------------------------*
*                Install PATCH1 so that PROSHADOW           *
*               may exit now and be called later.           *
*-----------------------------------------------------------*

DØ5D: AØ ØF            LDY #$ØF      Length of patch
     PATCH1LP
DØ5F: B9 9F DØ         LDA PATCH1,Y Get patch byte
DØ62: 99 4Ø BF         STA PATCHLOC,Y put in patch area
DØ65: 88               DEY
DØ66: 1Ø F7            BPL PATCH1LP Continue until done
*-----------------------------------------------------------*
*                      Restore registers                    *
*-----------------------------------------------------------*

DØ68: 68               PLA          ;Get Y-reg
DØ69: A8               TAY
DØ6A: 68               PLA          ;Get A-reg

*-----------------------------------------------------------*
*                   Done. Return to caller.                 *
*-----------------------------------------------------------*

DØ6B: 4C 47 BF         JMP EXIT
```

Continued on next page

Another possible modification is to have ProShadow leave its information on the screen. The original program does not do this in order to allow operation with programs which use screen memory for program space. But this causes the information to be rapidly displayed. A modification can be made to the original code by doing the following:

```
4Ø93:EA EA EA       (was 2Ø 6E DØ)
4ØA9:EA EA EA EA    (was B9 DØ Ø7 48)
4ØB3:EA EA EA EA    (was 68 99 BF DØ)
```

I'm sure that someone out there will think of other patches which also can be used with ProDOS. If you do, share them with the rest of us! Remember, since ProDOS uses the RAM card, you cannot call any ROM routines (monitor, Applesoft, etc.) without using a switch routine. Also, make sure that any additions made to ProShadow do not make it extend past $DØFF because ProDOS keeps its Quit code starting there.

Interested in SNOOPING?

If you don't have a ProDOS disk snooping routine you will have to make do with a DOS 3.3 snoop program. To make sense out of the block numbers you have to know whether your disk editor uses logical or physical sector numbers. DOS numbers the sectors on the disk in numerical order; however, for faster access, DOS uses a lookup table to determine which logical sector corresponds to which physical sector. If you are confused, check one of the references given at the end of this article.

ProDOS uses two 256-byte sectors to make each 512-byte block. The disk sectoring is exactly the same as that used by Pascal. To determine the sectors to examine for a given block, use the following table:

SECTORING		1ST & 2ND HALF OF BLOCK							
PHYSICAL	-	Ø&2	4&6	8&A	C&E	1&3	5&7	9&B	D&F
DOS 3.3	-	Ø&E	D&C	B&A	9&8	7&6	5&4	3&2	1&F
PRODOS	-	Ø&1	2&3	4&5	6&7	8&9	A&B	C&D	E&F

With this table, and your favorite disk snooping program, you can look into what ProDOS is reading from the disk using the information given you by ProShadow.

I'm confident that ProShadow will open new doors (and disks) for you, and I hope that you find it as powerful and useful as I have. Remember, if you come up with any good modifications, let the rest of us know!

References:

Worth Don & Peter Lechner, *Beneath Apple DOS, Beneath Apple ProDOS, Supplement to Beneath Apple ProDOS.* Chatsworth, California: Quality Software.

Road Maps to Apple II Disks: DOS 3.3, CP/M, Pascal, and ProDOS. Skillman C. Kim Hunter. Call-A.P.P.L.E., V8, #2 (February 1985), pp. 10-21.

The Armonitor. Nick Galbreath. Hardcore COMPUTIST No. 12, (1985), p. 23.

APT's

Hitchhiker's Guide to the Galaxy
Infocom

To get the babel fish:

1. Put the coat on the hook
2. Put the towel over the grate
3. Get satchel (Ford's)
4. Cover panel with satchel
5. Put mail on satchel
6. Push button

and voila!, you now have a babel fish in your ear.

Contributed by Gary Wu

Continued from previous page

```
*---------------------------------------------------------------*
*                      SWAPEM subroutine                        *
*                                                               *
*            This subroutine swaps the bottom line              *
*                of the text screen with MASK.                  *
*---------------------------------------------------------------*

       SWAPEM
DØ6E:  AØ 27              LDY #39
       LOOP1
DØ7Ø:  B9 DØ Ø7           LDA SCREEN,Y
DØ73:  48                 PHA
DØ74:  B9 BF DØ           LDA MASK,Y
DØ77:  99 DØ Ø7           STA SCREEN,Y
DØ7A:  68                 PLA
DØ7B:  99 BF DØ           STA MASK,Y
DØ7E:  88                 DEY
DØ7F:  1Ø EF              BPL LOOP1
DØ81:  6Ø                 RTS

*---------------------------------------------------------------*
*                    PRINTHEX subroutine                        *
*            This subroutine prints the value of               *
*            the A-reg as two hexadecimal digits.               *
*            DIGITOUT is used to print the digits.              *
*---------------------------------------------------------------*

       PRINTHEX
DØ82:  48                 PHA          ; Save A-reg
DØ83:  4A                 LSR          ; Get hi-nibble
DØ84:  4A                 LSR
DØ85:  4A                 LSR
DØ86:  4A                 LSR
DØ87:  2Ø 91 DØ           JSR DIGITOUT Print hi-nibble
DØ8A:  68                 PLA
DØ8B:  29 ØF              AND #$ØF     Get lo-nibble
DØ8D:  2Ø 91 DØ           JSR DIGITOUT Print lo-nibble
DØ9Ø:  6Ø                 RTS

*---------------------------------------------------------------*
*                    DIGITOUT subroutine                        *
*                                                               *
*            DIGITOUT prints a single hex digit at              *
*              SCREEN,Y and increments Y-reg.                   *
*---------------------------------------------------------------*

       DIGITOUT
DØ91:  Ø9 BØ              ORA #$BØ     Convert to high ASCII
DØ93:  C9 BA              CMP #$BA     Check for Ø-9
DØ95:  9Ø Ø3              BLT LOWDIGIT go if yes
DØ97:  18                 CLC
DØ98:  69 Ø7              ADC #$Ø7     Convert to A-F
       LOWDIGIT
DØ9A:  99 DØ Ø7           STA SCREEN,Y Put on screen
DØ9D:  C8                 INY
DØ9E:  6Ø                 RTS

*---------------------------------------------------------------*
*           These are the bank switching routines               *
*           used by PROSHADOW to access ProDOS.                 *
*           The appropriate routine is moved to                 *
*                $BF4Ø before being called.                     *
*     The routines CANNOT be larger than 16 bytes!!!            *
*                                                               *
*                                                                              *
*                      PATCH1 is used as an entry point to                     *
*                      PROSHADOW as well as it's exit point.                   *
*------------------------------------------------------------------------------*

      PATCH1
                 .OR PATCHLOC
                 .TA $40D8
BF40: D8                   CLD          ;All drivers begin w/CLD
BF41: 2C 83 C0             BIT RAM2RD   Turn on bank2
BF44: 4C 00 D0             JMP BEGIN    Call PROSHADOW
      EXIT
BF47: 2C 8B C0             BIT RAM1RD   Turn on bank1
BF4A: 2C 8B C0             BIT RAM1RD   and write enable
BF4D: 28                   PLP          ;Get P-reg
BF4E: 60                   RTS
BF4F: EA                   NOP
*------------------------------------------------------------------------------*
*                       PATCH2 is used to call the disk                        *
*                               device driver.                                 *
*------------------------------------------------------------------------------*

      PATCH2
                 .OR PATCHLOC
                 .TA $40E8
BF40: 2C 8B C0             BIT RAM1RD   Turn on bank1
BF43: 2C 8B C0             BIT RAM1RD   and write enable
      JSRDRVER
BF46: 20 00 D0             JSR DRIVER   Call the driver
BF49: 08                   PHP          ;Save P-reg
BF4A: 2C 83 C0             BIT RAM2RD   Turn on bank2
BF4D: 4C 54 D0             JMP CLEANUP  and re-enter PROSHADOW
                 .OR PATCH1+$20
                 .TA $40F8

*------------------------------------------------------------------------------*
*                         Mask for last screen line                            *
*------------------------------------------------------------------------------*

D0BF: C3 CD C4
D0C2: BD A0 A0
D0C5: A0 A0 A0
D0C8: D3 BD A0
D0CB: A0 A0 A0
D0CE: C4 BD A0
D0D1: A0 A0 A0
D0D4: C2 D5 C6
D0D7: BD A0 A0
D0DA: A0 A0 A0
D0DD: A0 A0 C2
D0E0: CC CB BD
D0E3: A0 A0 A0
D0E6: A0         MASK      .AS -"CMD=     S=    D=    BUF=       BLK=    "
D0E7- END        .EQ *
```

Need Back Issues?

See the ad on page 31.

APT's

Bruce Lee
Datasoft, Inc.

Here is an easy (but not quick) way to get extra men in this very difficult game. Note that at every **30,000**, **50,000** and **100,000**, the player will receive an extra man...this cycle repeats every **100,000** points.

When your man appears on the first street scene, go up to the second (middle) floor by climbing the ladder and running right to the second screen. When you reach the second screen, move about three steps from the left edge of the picture, face left, and hold the punch button down so that Bruce Lee stands in the punching position. When the Ninja and the Sumo men appear, they will be immediately punched out and you will receive approximately 3,000-4,000 points.

Next, go back to the first screen, face right, and repeat the same process. Once you get the hang of it, you can get **200,000** points in about 2 minutes! If you do this and stop at **200,000** points to begin the game, you will have ten men instead of four.

When you have become good enough to reach the final screens, we have one more tip for you. In the final battle scene against the wizard (the one where the face spits killer things at your man), just run from the left to the right without stopping and you will see the victory scene.

Contributed by Francis Wong and Craig Phillippe

APT's

Tubeway
Datamost

To start at any level from 1-9 all you have to do is: press ESCAPE, then press $ (shift 4).

Talon

To get a free man, press CONTROL (don't release it) and then W. This will result in one extra man, but don't be greedy. *Terrible* things may result.

Contributed by Ron Searle.

By Clay Harrell

Deprotecting Dazzle Draw

(from Broderbund Software)

Requirements:
Apple //e or //c with 128K
ProDOS users disk
1 blank INITialized disk
1 blank disk
Super IOB v1.2 (or COPYA and a sector editor)
Dazzle Draw from Broderbund Software

In comparing the software (for the Apple][computer) marketed by various companies, it becomes apparent that some expend considerably more effort to protect their releases than do other companies. The subject of able protection schemes brings to mind publishers like Sirius (RIP), Electronic Arts and Broderbund. Usually deprotecting a program sold by one of these companies is very difficult and requires a great deal of time and ingenuity.

Having written several articles describing the removal of copy protection from these companies' releases, I have come to realize that there is a definite connection between the difficulty in deprotecting the program and writing an understandable description of how to do it.

Dazzle Draw And Its Protection

Broderbund's most recent release is an excellent double hi-res graphic drawing program that looks much like a Macintosh program. It utilizes windows and requires a joystick or a mouse for user input. And like the program itself, Dazzle Draw's protection is also very good.

I pay particular attention to the sound of protected disks as they boot because sometimes you can tell a great deal about the protection, and Dazzle Draw is no exception. During the bootup of Dazzle Draw, you can hear some unusual disk action when the disk head is over track $1F. Closer examination (via nibble editor) reveals that this sound is most probably an indicator of the infamous nibble count of sorts and therefore must be defeated.

In addition to the nibble count, Broderbund has changed the end of address marker from $DE AA to $DE FF. This change is rather easy to circumvent, though. If you make a copy (by ignoring the last byte in the address field epilog) of tracks $00 to $1E and insert this copy in the drive after the nibble count, you will find the program seems to run fine. It can then be inferred that this is the only protection we will have to defeat.

In Detail

The following is a detailed explanation of how I deprotected Dazzle Draw (if you don't wish to wade through the explanations, skip ahead in the article to the section entitled ''Step-By-Step'').

First of all, I copied tracks $00 through $1E by using a Super IOB controller. Another way to produce the same copy would be to use COPYA in the following manner:

1) Load COPYA and its object file

RUN COPYA

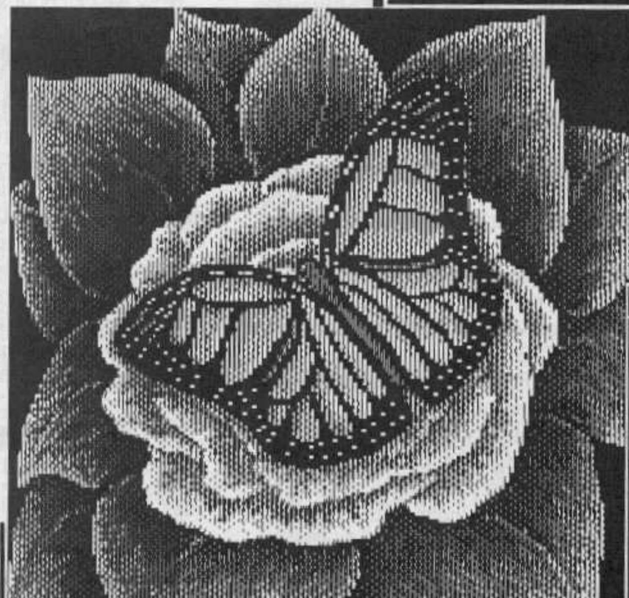
2) After the drive stops and COPYA is ready to accept slot and drive numbers, cause a BASIC break

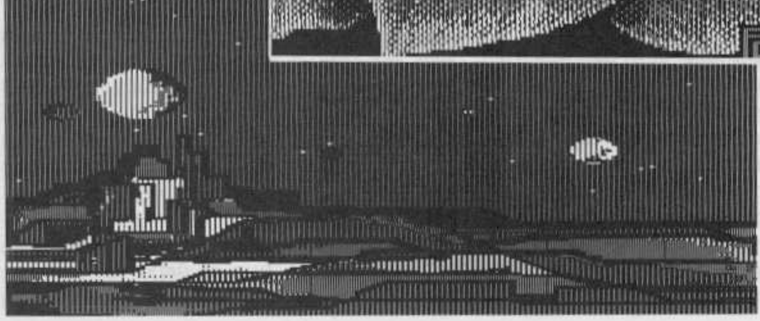
⌃C

3) Enter the monitor and tell COPYA's object file to copy only tracks $00 through $1F

```
CALL -151
302:1F
35F:1F
```

4) Tell DOS to ignore the last byte of the address epilog marker and return to BASIC





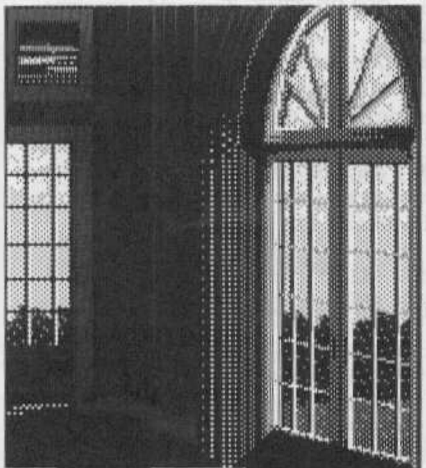
```
B99D:00
3D0G
```

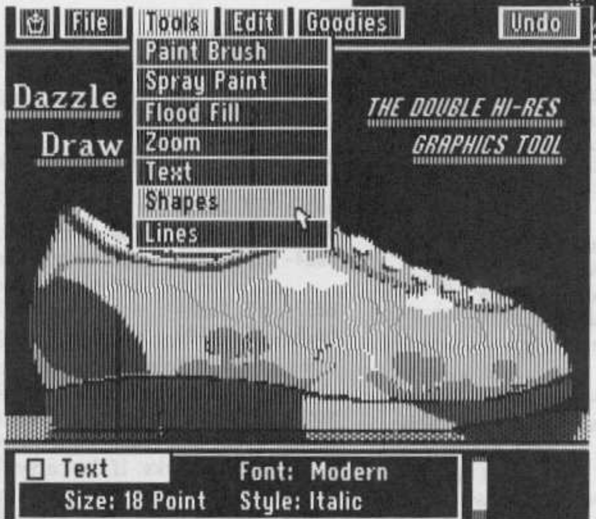
5) Tell COPYA not to reload its object file and make the copy

```
70
RUN
```

Back to the Nibble Count...

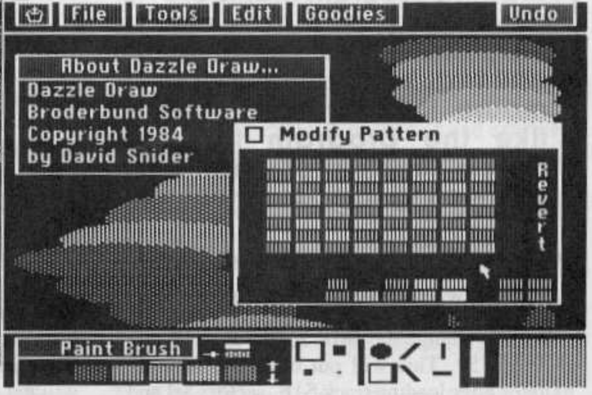
I usually defeat nibble count routines by using an NMI card (I have a Replay][card) which allows me to stop the routine while it is executing. The NMI card then tells me the location in memory at which the nibble count is loaded. This allows me to examine and modify it so that it no longer functions. By applying this change to the disk (first I have to find it with a disk searcher) it is permanent and I have a deprotected version of the program.





□ Text   Font: Modern
Size: 18 Point   Style: Italic

Unfortunately, because I have no hardware of this nature that works on the //e, I was forced to use my second (backup) procedure: ''Boot Code Tracing''.



About Dazzle Draw...
Dazzle Draw
Broderbund Software
Copyright 1984
by David Snider

Boot Code Tracing

The concept behind boot code tracing has been explained in just about every issue of

Hardcore COMPUTIST. To avoid appearing redundant, I will instead refer uninformed readers to the Softkey for Bankstreet Writer (Issue No. 18) for a complete explanation.

After completing the usual boot-code tracing steps ("86ØØ<C6ØØ.C6FFM", "86F9:59 FF", "86ØØG" and "8Ø1L") I was able to examine the first stage of the Dazzle Draw boot. It looked someting like this:

```
Ø8Ø1- STX $43      :
Ø8Ø3- LDA #$FF     :
Ø8Ø5- STA $Ø4FB    :
Ø8Ø8- STA $CØØ8    :Main stack and zero page
Ø8ØB- STA $CØØ4    :write main memory.
Ø8ØE- STA $CØØ2    :read main memory.
Ø811- STA $CØØC    :
Ø814- STA $CØØØ    :8Ø columns off.
Ø817- STA $CØ81    :RAM card off.
Ø81A- STA $Ø3F3    :Reboot on reset.
Ø81D- STA $Ø3F4    :Reboot on reset.
Ø82Ø- JSR $FB2F    :Set up zero page.
Ø823- JSR $FE89    :Set up zero page.
Ø826- JSR $FE93    :Set up zero page.
Ø829- JSR $FC58    :-----
Ø82C- LDX #$Ø3     :This routine loads accum
Ø82E- LDA $Ø845,X:with $845 to $848
Ø831- STA $AØ,X    :and stores it
Ø833- DEX          :at $AØ to $A3.
Ø834- BPL $Ø82E    :Are we done?
Ø836- LDA #$ØØ     :-----
Ø838- STA $FF      :Reset counter.
Ø83A- LDA #$3E     :
Ø83C- JSR $Ø899    :Check for original disk.
Ø83F- JSR $Ø849    :get stage from T1F SB&C
Ø842- JMP $6ØØØ    :Jump to next stage.
```

Interestingly, the above code shows that the next stage of the load is read from Track $1F, sectors $B and $C, which are a totally different format than the rest of the disk (tracks $Ø to $1E).

Broderbund's most recent release is an excellent double hi-res graphic drawing program that looks much like a Macintosh program. It utilizes windows, and requires a joystick or a mouse for user input. And like the program itself, Dazzle Draw's protection is also very good.

Onward Goes The Boot

We can change the JMP $6ØØØ at $842 to a JMP $FF59. This will put us in the friendly monitor after loading track $1F, sectors $B and $C into $6ØØØ-61FF. In order to do this, we also must change the disk controller code at $86ØØ to go through the motions of loading track Ø, sector Ø into $8ØØ but not really do it.

This is accomplished by changing the JMP $FF59 at $86F8 back to a JMP $8Ø1 so at executes that code (and eventually bombs at $842) and we must also change the byte at $8659 from a $Ø8 to a $5Ø. This byte tells the disk controller card where to store track Ø, sector Ø (normally page 8). But since we do not want to overwrite the changes we made at $8ØØ, we tell the disk controller code to read track Ø, sector Ø into page $5Ø ($5ØØØ) and jump to our modified code at $8Ø1! (NOTE: We cannot simply execute the code at $8Ø1 without going through the entire boot process. This is due to some zero page parameters that get set during the boot process.)

Now we can execute the code at $86ØØ by typing "86ØØG". After a moment, we will hear a beep and see the monitor prompt. You may turn off the drive by typing "CØE8".

If you examine the code at $6ØØØ, this is what you will find:

```
6ØØØ- JMP $6Ø1A   :jump to $6Ø1A.
  .     .          .
  .     .          .
  .     .          .
6Ø1A- LDX #$FF    :reset the
6Ø1C- TXS         :stack.
6Ø1D- LDX $2B     :
6Ø1F- STX $Ø8     :
6Ø21- JSR $619A   ://e or c with 128K?
6Ø24- JSR $6Ø2C   :read protected tracks
                  :and nibble count
6Ø27- BCS $6Ø24   :error, go back to $6Ø24
6Ø29- JMP $64ØØ   :exit to next stage.
```

Basically, this routine checks to make sure you have a 128K //e or //c, and then does a nibble count and reads some more info from the protected tracks. If the carry flag is clear, it thinks everything went OK on the nibble count, and continues to the next set of routines at $64ØØ. Here at $6ØØØ is the primary protection, as the code at $64ØØ loads in the remainder of the program in a normal manner from the unprotected tracks $ØØ to $1E.

Now for the final step of the trace! We change the code at $842 from the jump to monitor back to jump to $6ØØØ. Since we are about to change the newly loaded code at $6ØØØ, we have to tell the code at $8Ø1 not to overwrite the code at $6ØØØ during the boot, much like we did to the code at $86ØØ. Therefore, we change byte $85Ø from a $6Ø (page $6Ø or $6ØØØ) to a $4Ø (page $4Ø or $4ØØØ). This will cause the code at $8Ø1 to read track $1F, sectors $B and $C into $4ØØØ-41FF, instead of $6ØØØ-61FF. Then we can jump to our modified code at $6ØØØ.

Finally, we change the code at $6ØØØ to execute and halt before running away from us. This will read the remainder of the protected tracks and stop before jumping to $64ØØ. To do this we change byte $6Ø2B from a $64 to $ØF, which causes the program to jump to $FØØ instead of $64ØØ. Now you ask, "Why $FØØ?". We will put our own routine at $FØØ, which will get executed instead of the code at $64ØØ.

The code $6ØØØ does the final nibble count, and there is really no more protection past that. So the idea is to let Dazzle Draw read the original disk and do the nibble counts and reading from the protected tracks. Then just after it finishes, we halt the code and save it. Then we can re-execute this code where it stopped, and use a COPYA version of Dazzle Draw instead of the original, and let the code continue loading the program.

So we need to put a "capture" routine to save memory from $ØØ to $8FF so we can re-start the Dazzle Draw load after the nibble count. I used $FØØ to store this routine at because it is a memory area that is unused by Dazzle Draw (at least to this point in the load). Here is the routine:

```
ØFØØ- STA $ØFFC   :store Accum.
ØFØ3- STX $ØFFD   :store X-reg.
ØFØ6- STY $ØFFE   :store Y-reg.
ØFØ9- TSX         :transfer stack ptr to X.
ØFØA- STX $ØFFF   :store X-reg.
ØFØD- LDX #$ØØ    :
ØFØF- LDA $ØØØØ,X:move $ØØ-8FF
ØF12- STA $8ØØØ,X:to $8ØØØ-88FF.
ØF15- INX         :
ØF16- BNE $ØFØF   :Done with this page?
ØF18- INC $ØF11   :Get next page LDA
ØF1B- INC $ØF14   :Get next page STA
ØF1E- LDA $ØF11   :
ØF21- CMP #$Ø9    :Done with all pages?
ØF23- BNE $ØFØD   :Nope, do next page.
ØF25- JMP $FF59   :Done, exit to monitor.
```

So, after you key in this routine (the hexdump can be found in Step seven under the "Step-By-Step" headline) you type "86ØØG" to start the boot trace over. In a moment, your Apple will beep and the monitor prompt will appear. Now type "FFC.FFF" and write down the values that appear. These are the values of the Accumulator, the X-register, the Y-register and the Stack pointer, respectively. Put the name of the register next to each value also.

Now we can boot a 48K DOS 3.3 slave disk and save our memory piece by typing "BSAVE DD,A$6ØØØ,L$29ØØ". Since this eventually captured startup file is eventually going to be a ProDOS system file (which must start at $2ØØØ), we have to relocate the captured code into $2ØØØ with a "21ØØ<6ØØØ.88FFM".

The purpose of all this was to halt the program after all the protection code was executed and satisfied. Now we can re-run our start up file and use a COPYA back-up of Dazzle Draw. But first, let's put the saved code into a single BRUNable file.

This is accomplished by typing in the following routine (the hexdump can be found under Step 14 of the "Step-By-Step" headline) which will move the memory back where it goes.

```
2ØØØ- LDX #$ØØ    :------
2ØØ2- LDA $41ØØ,X:Move memory from $41ØØ
2ØØ5- STA $ØØØØ,X:through $49FF to $ØØØØ
2ØØ8- LDA $21ØØ,X:Move memory from $21ØØ
2ØØB- STA $6ØØØ,X:through $4ØØØ to $6ØØØ
2ØØE- INX         :
2ØØF- BNE $2ØØ2   :Done with page
2Ø11- INC $2ØØ4   :Next page
2Ø14- INC $2ØØ7   :
2Ø17- INC $2ØØA   :
2Ø1A- INC $2ØØD   :
2Ø1D- LDA $2ØØA   :Examine current page
2Ø2Ø- CMP #$41    :Done with all pages?
2Ø22- BCC $2ØØ2   :Nope, continue
2Ø24- JSR $619A   :Check for 128K
2Ø27- LDA $CØE9   :Turn on Drive
2Ø2A- LDX #$FF    :
```

```
202C- TXS         :reload Stack Pointer.
202D- LDY #$00    :reload Y-reg.
202F- LDX #$60    :reload X-reg.
2031- LDA $C010   :clear keyboard.
2034- LDA #$BA    :reload accum.
2036- JMP $6400   :resume loading!
```

This program moves $4100-49FF back down to $00-8FF, moves $2100-40FF back to $6000-$7FFF, checks if you have a //e or //c with 128k, re-loads the registers, and restarts the programs at the point we stopped it. NOTE: At location $202B should be the value you wrote down for your stack pointer. At location $202E should be the value you wrote down for the Y-register. At location $2030 should be the value you wrote down for the X-register. And at location $2035 should be the value you wrote down for the accumulator, if they are different than the ones noted above.

Next change locations $75C5 and $75C6 from CB 75 to A7 7B by typing "36C5:A7 7B". This will prevent Dazzle Draw from saving your configuration on the COPYA disk. You don't want to do this since it saves the configuration in a non-standard DOS format, thus destroying your COPYA copy.

To save this routine, you would type "BSAVE DAZZLE.SYSTEM, A$2000, L$2A00".

Now For the ProDOS Part

As it now sits, you could boot normal DOS 3.3, BRUN DAZZLE.SYSTEM and put in your copy of Dazzle Draw and it would work fine. But this is a pain. I want a disk that all I have to do is boot it to get Dazzle Draw. At first thought, this might seem simple, all we have to do is put DAZZLE.SYSTEM on the copy of Dazzle Draw as the boot file. However, performing such a feat involves many other steps.

First of all, since the VTOC of the copy of Dazzle Draw shows every track in use and we need some space to put our startup file on the disk, we must edit track $0, sector $3 and put a $FF in bytes $1F-$22 which will free up tracks $1F through $22. Remember that these tracks held the same information as in our startup file anyway (just in a different format).

Next, we have to put a dummy file called PRODOS on the main directory which contains exactly the same data as the file called PRODOS on the utilities directory. This will avoid the "UNABLE TO LOAD PRODOS" message. To do this, we must first boot the ProDOS users disk, exit to BASIC, insert our copy of Dazzle Draw and type "CREATE PRODOS, TSYS".

Next, we have to boot our sector editor and make this file identical to the PRODOS file found on the utilities directory by changing byte $EE of track $00, sector $B of our copy to $26 and copying bytes $3B-$3C ($FF and $09 on my copy) of track $00, sector $01 to bytes $FE and $FF of track $00, sector $0B.

Next, we have to put our startup file on the ProDOS disk by booting the ProDOS users disk, selecting "C" to convert between DOS and ProDOS. After moving the file, we have to make it a system file by exiting to ProDOS BASIC and typing "BLOAD DAZZLE.SYSTEM", "DELETE DAZZLE.SYSTEM", "CREATE DAZZLE.SYSTEM, TSYS", "BSAVE DAZZLE.SYSTEM, A$2000, L$2A00, TSYS".

Finally, we have to get rid of the strange Dazzle Draw boot sequence by copying track 0, sector 0 of the ProDOS users disk to track 0, sector 0 of our copy disk.

Believe it or not, the resulting disk will boot exactly like the original (title page and all). Here is the cookbook method.

Step-By-Step

1) Turn your Apple //e or //c on and hit reset to stop the drive.

2) Enter the monitor

```
CALL -151
```

3) Move the disk controller ROM code to RAM and modify it so that it jumps to the monitor.

```
8600<C600.C6FFM
86F9:59 FF
```

4) Insert your original Dazzle Draw in drive 1 and execute the partial boot.

```
8600G
```

5) Your Apple will beep and the monitor prompt will appear. Now turn off the drive and make some mods so that the next stage of the boot can be halted and start up this boot.

```
C0E8
843:59 FF
86F9:01 08
8659:50
8600G
```

6) The drive will reboot and in a moment, your Apple will beep and the monitor prompt will appear. Now make a couple more modifications to stop the boot and jump to $F00.

```
C0E8
602B:0F
843:00 60
850:40
```

7) Type in this short "capture" routine to save memory $0000 through $08FF into $8000 through $88FF.

```
F00:8D FC 0F 8E FD 0F 8C FE
F08:0F BA 8E FF 0F A2 00 BD
F10:00 00 9D 00 80 E8 D0 F7
F18:EE 11 0F EE 14 0F AD 11
F20:0F C9 09 D0 E8 4C 59 FF
```

8) Execute this last boot.

```
8600G
```

9) The drive will reboot again and in a moment, your Apple will beep and the monitor prompt will appear. At this point, turn off the drive and examine what the 6502 registers were when we JuMPed into the monitor.

```
C0E8
FFC.FFF
```

10) Make a note or the values returned for the locations $FFC to $FFF. Remember, $FFC = Accum, $FFD = X-reg, $FFE = Y-reg, and $FFF = Stack Pointer.

11) Disable the configuration save option by typing:

```
75C5:A7 7B
```

12) Boot a normal DOS 48K slave disk and save the memory pieces.

```
C600G
BSAVE DD,A$6000,L$2900
```

13) Enter the monitor and move the memory pieces to $2100.

```
CALL -151
2100<6000.88FFM
```

14) Type in this routine which reconstructs the memory.

```
2000:A2 00 BD 00 41 9D 00 00
2008:BD 00 21 9D 00 60 E8 D0
2010:F1 EE 04 20 EE 07 20 EE
2018:0A 20 EE 0D 20 AD 0A 20
2020:C9 41 90 DE 20 9A 61 AD
2028:E9 C0 A2 FF 9A A0 00 A2
2030:60 AD 10 C0 A9 BA 4C 00
2038:64
```

15) Now you can enter the values you wrote down for the 4 registers

```
2035:xx  (value for Accumulator)
2030:xx  (value for the X-register)
202E:xx  (value for the Y-register)
202B:xx  (value for the Stack Pointer)
```

16) Save the whole Dazzle Draw startup program

```
BSAVE DAZZLE.SYSTEM, A$2000,
  L$2A00
```

17) Make a copy of your Dazzle Draw disk by either using the Super IOB controller printed at the end of this article or the procedure outlined earlier in this article. If you use the COPYA procedure, then you will need to sector edit bytes $1F-$22 of track 0, sector 3 all to $FF on your copy of Dazzle Draw.

18) Next, boot up the ProDOS users disk, exit to BASIC, insert your copy of Dazzle Draw and create a dummy PRODOS file.

```
CREATE PRODOS, TSYS
```

19) Insert your ProDOS users disk and execute the DOS converter program.

```
-CONVERT
```

20) For Steps 21 through 25, if you have one disk drive, use the step labeled "XX.1" and if you have two drives, then use the steps labeled "XX.2".

21.1) Set the prefix by pathname (Press "P" twice) to "/RAM".

21.2) Set the prefix by pathname (Press "P" twice) to "/DD".

22.1) Insert the disk you saved the system file on and convert "SYSTEM.DAZZLE". This will move "SYSTEM.DAZZLE" from your DOS 3.3 disk to an artificial RAM disk in your extended 80 column card.

22.2) Insert the disk you saved the system file on in drive 2 and your copy of Dazzle Draw

in drive 1 and convert "DAZZLE.SYSTEM"
23.1) Insert the ProDOS users disk and type "Q" to quit. When asked for a filename, use:

/USERS.DISK/BASIC.SYSTEM

23.2) Skip ahead to Step 26.
24.1) Without powering down, startup the ProDOS file transfer program by typing "F".
25.1) Move "DAZZLE.SYSTEM" from your RAM disk to your copy of Dazzle Draw by pressing "F", "C" and using "/RAM/=" and "/DD/=" as pathnames respectively.
26) Make DAZZLE.SYSTEM into a SYStem file.

```
BLOAD DAZZLE.SYTEM,A$2000
DELETE DAZZLE.SYSTEM
CREATE DAZZLE.SYSTEM, TSYS
BSAVE DAZZLE.SYSTEM, A$2000,
   L$2A00, TSYS
```

27) Boot up your sector editor and make the dummy ProDOS file exactly the same as the ProDOS file on the utilities directory.

Track	Sector	Byte	From	To
$00	$0B	$EE	$16	$26
$00	$0B	$FE	$FF	$FF
$00	$0B	$FF	$F8	$09

28) Finally, using your sector editor, copy track 0, sector 0 of your ProDOS users disk to track 0, sector 0 of your copied Dazzle Draw disk.

That's it! Have fun with your deprotected Dazzle Draw.

Dazzle Draw Controller

```
1000 REM DAZZLE  DRAW
1010 TK = 0 :ST = 0 :LT = 31 :CD = WR
1020 T1 = TK : GOSUB 490 : RESTORE : GOSUB 170
1030 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1030
1040 IF BF THEN 1060
1050 ST = 0 :TK = TK + 1 : IF TK < LT THEN 1030
1060 GOSUB 310 : GOSUB 230 : GOSUB 490 :TK = T1
     :ST = 0
1070 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1070
1080 ST = 0 :TK = TK + 1 : IF BF = 0 AND TK < LT THEN
     1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "DONE^WITH^COPY' : END
5000 DATA 222 ,255 ,222 ,170
5010 DATA 4^CHANGES ,0 ,3 ,31 ,255 ,0 ,3 ,32 ,255
     ,0 ,3 ,33 ,255 ,0 ,3 ,34 ,255
```

Controller Checksums

```
1000  - $356B      1070  - $180F
1010  - $32C4      1080  - $A606
1020  - $554E      1090  - $3D08
1030  - $434F      1100  - $6FD7
1040  - $7BF4      5000  - $00CE
1050  - $FFC5      5010  - $8733
1060  - $100A
```

ADVENTURE TIPS

***Cranston Manor**
Sierra On-Line

Music can affect secret doors.
The "Suit of Armor" is scared of something.
Piranha's can not chew through inflatable rafts.
Coins are good for vending machines.

***Zork II**
Infocom, Inc.

To get past the Lizard you need a key and something for the Lizard to eat.
To get the Unicorn's key you must save its master.
The key to the locked cell is in the keyhole on the inside of the room.

***Zork III**
Infocom, Inc.

To get all of your lost belongings in the lake, go under the water.
Do not let the man on the cliff help you up.
Offer bread to an old man for a secret tip.

**Contributed by Chris Windle.*

#Ultima III
Origin Systems

(Make a backup before you go into Exodus' castle. You won't regret it!)
Can't get to Exodus? Check out the 8th level of every dungeon.
Still can't get there? (PRAY) in the circle of light at Yew.
If monsters in Exodus' castle won't die, ready exotics.
If you keep dying while trying to (INSERT) Exodus, visited the Time Lord yet?
Don't leave a horse over a moongate position when it's not active. It will get sucked into oblivion.

#Amazon
Telarium

Lost in the jungle? Forget the map. Try North, East continually.
Parachutes make good hole stuffers.
Harassed by alligators? Use paddle power.
Paco doesn't like bridges. Don't shoot him, just prove that you can.
Don't bother with the rifle unless you're feeling suicidal.

#Contributed by Matthew Arnold

Copy Parameters

Copy][+ 4.3: The Hitchhiker's Guide to the Galaxy (Infocom)
0-22...10=96 (write protect)

Copy][+ 4.4 and EDD v2: Hitchhiker's Guide
0-22 (write protect)

Copy][+ 4.4: Beyond Castle Wolfenstein
0-22

Contributed by Dwayne Claud

The Blade of Blackpoole
Sirius Software

Meat eating plants could easily rid you of your insect enemies.
If you're going on a boat ride, remember to put the potion in the boat with you. And don't forget to go back for the shield.
Got a monster in your path? You can't drown him, but maybe you can get him drunk??!
Don't use too much logic when confronted with a boulder. Use your fork like you would a sword.

Crime Stopper
Hayden Software

Where should you look for a safe? Behind a picture, of course. Use combination L36-R26-L26.
Wait around in Room 209. Soon the phone will ring. And don't forget to look around. You'll find some interesting reading material.
You might have to bribe some seedy character, but at only $5 a shot, it's almost worth it.
If you've taken an unexpected "nap" in the Warehouse, and awake in a "cold" situation, try standing on the desk. Need a smoke? Light up (& be a litter bug.)

Time Zone
Sierra On-Line

Is the pterodactyl giving you trouble? Look for a place to hide.
Maybe you can "light up" the view of the Inca pyramid. Try dropping the torch when you're at the top.
If the stampeding herd is "overhead" you can't be trampled.
Check out the food in Tokyo time. Try selection 2 or 3. Eat hearty!

Softkey For Twerps

By M.M. McFadden

Twerps
Sirius Software
1Ø364 Rockingham Drive
Sacramento, CA 95827

Requirements:
48K Apple][with disk drive
A blank disk

Twerps is a somewhat lame shoot-em-up with eight repetitive levels. Although it is copy protected fairly well, cracking it isn't all that difficult. The procedure is a bit confusing, though, so if you've never cracked a program or used a softkey before, this isn't the one to start with. To help you along, I have included comments and explanations in parenthesis next to some of the commands you need to type in. DO NOT type in any of these comments (your Apple will beep at you angrily).

The first thing you will need to begin this procedure is a blank disk. Put the blank disk in the drive, and type

```
FP
INIT HELLO
```

We are going to crack this disk with the boot-trace method (by following and controlling its boot process), and my favorite utility for this particular occasion is a little program I wrote called RDØ. Part of the program, located at $9ØØ, reads in track Ø, sector Ø (the first part of the boot program) from a copy-protected disk and then stops, allowing examination and modification of the code. Another part of RDØ, located at $AØØ, uses the examined and modified code to continue booting. RDØ is created as follows:

1) Enter the monitor

```
CALL -151
```

2) Move code from $C6ØØ-C6FF to $9ØØ-9FF

```
9ØØ<C6ØØ.C6FFM
```

3) Also move this code to $AØØ-$AFF

```
AØØ<C6ØØ.C6FFM
```

4) Make the following modifications:

```
921:A9 6Ø
923:EA EA EA EA EA EA EA EA
92B:EA
9F8:BD 88 CØ 4C 69 FF
A21:A9 6Ø
A23:EA EA EA EA EA EA EA EA
A2B:EA
AC6:F8 D8
AE6:F8 D8
```

5) Save this code to disk

```
BSAVE RDØ,A$9ØØ,L$2ØØ
```

Now The Real Part Begins

Put the Twerps disk in the drive, and type

```
9ØØG
```

The disk will clatter for a moment and you will be returned to the monitor. Now type

```
8Ø1L
```

Your screen should look like this:

```
Ø8Ø1-  8D 5Ø CØ   STA $CØ5Ø
Ø8Ø4-  8D 52 CØ   STA $CØ52
Ø8Ø7-  8D 54 CØ   STA $CØ54
Ø8ØA-  8D 57 CØ   STA $CØ54
Ø8ØD-  A6 2B      LDX $2B
Ø8ØF-  A9 Ø4      LDA #$Ø4
Ø811-  85 11      STA $11
Ø813-  AØ ØØ      LDY #$ØØ
Ø815-  84 1Ø      STY $1Ø
Ø817-  BD 8C CØ   LDA $CØ8C,X
Ø81A-  1Ø FB      BPL $Ø817
Ø81C-  C9 DD      CMP #$DD
Ø81E-  DØ F7      BNE $817
Ø82Ø-  BD 8C CØ   LDA $CØ8C,X
Ø823-  1Ø FB      BPL $Ø82Ø
Ø825-  C9 AD      CMP #$AD
Ø827-  DØ F3      BNE $Ø81C
Ø829-  BD 8C CØ   LDA $CØ8C,X
Ø82C-  1Ø FB      BPL $Ø829
Ø82E-  C9 DA      CMP #$DA
```

This information was located at track Ø, sector Ø.

The RDØ program (I'll refer to it as the $800 program) requires a short explanaton. The code at $801-$80C turns the hi-res page "on" to hide the text page (the reasons will become apparent in a moment), then sets up some zero page pointers at $80D-$816. Look carefully at $817-$82F. 'LDA $C08C,X' is the command which will read information from the disk. Note the compare commands. When three successive read and compare commands are issued, it can be assumed that the program is looking for either an address field or a data field. In this case, DD AD DA happens to bethe address field of the disk. The address fields in parameter lists are located in this way.

To return to the problem at hand- the $800 program will read four sectors of information for the next stage of the boot process…right on top of the text page at $4ØØ (I'll call it the $4ØØ program). Because Twerps activated the hi-res screen, you are not supposed to notice that the text page is being used by the program. Having code on the text page is just fine, except that as soon as we type something, we alter part of the text page and, with it, part of the $4ØØ program! There are a few things we must do to prevent such errors:

1)Keep everything below $2ØØØ safe

```
853:2Ø
85C:AØ
```

2) Make the $8ØØ program jump to $BØØ instead of $4ØØ

```
87D:4C ØØ ØB
```

3) Place a program at $BØØ which will move the $4ØØ program to $CØØ (we can now look at it without erasing it)

```
BØØ:AØ ØØ B9 ØØ Ø4 99 ØØ ØC
BØ8:C8 DØ F7 EE Ø4 ØB EE Ø7
B1Ø:ØB AD Ø4 ØB C9 Ø8 DØ EA
B18:4C 65 FF
```

Verify that you typed it in correctly by typing

```
BØØ.B1A
```

DOS will be erased from memory when the Twerps disk boots, so we will deactivate it now. If your computer is connected to DOS, when it isn't in memory, you will suffer a rather unpleasant experience.

Set the reset vector to jump to the monitor

```
3F2:65 FF 5A
```

and press CTRL RESET.

The Moment Of Truth

Run the $8ØØ program

```
AØØG
```

The disk will clatter and the speaker will beep. When this happens, type

```
CØ51 CØE8
```

(You won't be able to see what you're typing because you're looking at the hi-res page. Accessing $C051 and $C0E8 will set text mode and turn off the drive, respectively). If all has gone well, you should see a screen with garbage and inverse @ signs on it. Now type

```
CØØL
```

Your screen should look like this:

```
ØCØØ-  AØ ØØ      LDY #$ØØ
ØCØ2-  59 ØØ ØØ   EOR $ØØØØ,Y
ØCØ5-  C8         INY
ØCØ6-  DØ FA      BNE $ØCØ2
ØCØ8-  A8         TAY
ØCØ9-  FØ Ø3      BEQ $ØCØE
ØCØB-  4C 4Ø Ø5   JMP $Ø54Ø
ØCØE-  A9 4Ø      LDA #$4Ø
ØC1Ø-  8D F2 Ø3   STA $Ø3F2
ØC13-  A9 Ø5      LDA #$Ø5
ØC15-  8D F3 Ø3   STA $Ø3F3
```

```
ØC18-   49 A5       EOR #$A5
ØC1A-   8D F4 Ø3    STA $Ø3F4
ØC1D-   86 2B       STX $2B
ØC1F-   EA          NOP
ØC2Ø-   AD 81 CØ    LDA $CØ81
ØC23-   AD 81 CØ    LDA $CØ81
ØC26-   AØ ØØ       LDY #$ØØ
ØC28-   84 ØØ       STY $ØØ
ØC2A-   A9 DØ       LDA #$DØ
```

This is the $4ØØ program that was loaded on the text page at $4ØØ, but has been moved to $CØØ by the routine at $BØØ. Sound confusing? You ain't seen nothin' yet...

The program at $CØØ is the last stage of the boot program. The first bit of code at $CØØ-$CØD is known as a "spoilsport" routine because, if conditions are not exactly as they should be (they were altered by us), then the program dies because it thinks that someone is trying to copy it ("If I can't load my program, then you can't either"). Fortunately, such routines are simple to eliminate (Take that, brat!):

CØB:EA EA EA

The program now proceeds to set the reset vector at $CØE-$C1C and, at $C2Ø-C38, it clears out the language card.

Type "L" again to see the rest.

```
ØC2C-   85 Ø1       STA $Ø1
ØC2E-   B1 ØØ       LDA ($ØØ),Y
ØC3Ø-   91 ØØ       STA ($ØØ),Y
ØC32-   C8          INY
ØC33-   DØ F9       BNE $ØC2E
ØC35-   E6 Ø1       INC $Ø1
ØC37-   DØ F5       BNE $ØC2E
ØC39-   A9 4Ø       LDA #$4Ø
ØC3B-   8D FC FF    STA $FFFC
ØC3E-   A9 Ø5       LDA #$Ø5
ØC4Ø-   8D FD FF    STA $FFFD
ØC43-   AØ 8Ø CØ    LDA $CØ8Ø
ØC46-   A9 A2       LDA #$A2
ØC48-   85 36       STA $36
ØC4A-   85 38       STA $38
ØC4C-   A9 Ø5       LDA #$Ø5
ØC4E-   85 37       STA $37
ØC5Ø-   85 39       STA $39
ØC52-   A9 ØØ       LDA #$ØØ
ØC54-   BA          TSX
```

At $C39-$C42, the program sets locations $FFFC and $FFFD to $4Ø and $Ø5, respectively. If you thought that the vector at $3F2 was the last word in reset control, forget it. When you hit reset, the Apple immediately looks at locations $FFFC and $FFFD for directions. Because these locations are in ROM, they can't normally be set. But by using the Language Card, they can be changed. In this case the change occurs at $54Ø, which happens to be a routine in the Twerps program which will erase memory and reboot the disk! We can't let this happen, for obvious reasons. To rectify the situation, type:

C3A:65
C3F:FF

Your Apple will now jump to the monitor whenever **<RESET>** is pressed... no matter *WHAT* $3F2 is set to. The program then sets up zero page pointers at $36-$39 so that the second we type anything or print out a single letter, the program will crash (someone had a field day with these spoilsport routines!). Fix that by typing:

DA2:2Ø 89 FE 2Ø 93 FE 4C 65 FF

Instead of crashing, it will jump to the monitor. Now enter:

D3D:4C 65 FF

This will make the $4ØØ program stop when it is done loading. This brings us to a small problem. Twerps is really *loaded* with spoilsport routines, and this one is the worst. Because it can't be eliminated, we must go around it.

The problem here is that the $8ØØ program wants to load the $4ØØ program onto the text page and jump straight to it. However, we stopped the $4ØØ program before it had a chance to execute and, in doing so, mangled a number of zero page pointers (horrors!). The only thing to do now is to execute the $8ØØ program again but, instead of moving the $4ØØ program from $4ØØ to $CØØ, we are going to move the modified $4ØØ program from $CØØ, put it back at $4ØØ and execute it. We accomplish this great (and confusing) feat by modifying the move routine so that it is essentially reversed and jumps to $4ØØ instead of $CØØ

BØ4:ØC
BØ7:Ø4
B12:Ø7
B18:4C ØØ Ø4

Now type

AØØG

This will read the $4ØØ program off the disk, and then promptly ignore it by jumping to $BØØ, which will move the $4ØØ program at $CØØ to $4ØØ, and execute it (huh?).

The drive will chug away, the screen will change to hi-res, and the speaker will beep when it's all over. You might now begin to wonder if you should have begun this softkey at all, but don't worry. The worst is over. To see what you're doing, type

CØ51 CØ54

Because you are looking at the hi-res page, you won't be able to see this as you type. When you are on text mode again, you should see the familiar garbage 'n @ signs combination on the screen.

When you boot a DOS 3.3 disk, it overwrites most of the memory from $9ØØØ-$BFFF. Therefore, we will need to move portions of Twerps which would be overwritten by DOS if we tried to boot our blank disk. Fortunately, the folks at Sirius provided two large expanses of blank space, located from $8ØØ to $1FFF and $4ØØØ to $5FFF (everything below hi-res page 1 and all of hi-res page 2). To move the endangered portions of Twerps to a safe location, type

1ØØØ<9ØØØ.9FFFM
4ØØØ<AØØØ.BFFFM

Now insert your blank disk and type

6 CTRL P

The disk will boot and leave you with the Applesoft prompt (]). Now enter the monitor and type in this program which restores everything

CALL -151

```
ØF8Ø: AØ ØØ B9 ØØ 1Ø 99 ØØ 9Ø   $B364
ØF88: C8 DØ F7 EE 84 ØF EE 87   $3BBØ
ØF9Ø: ØF AD 84 ØF C9 2Ø DØ EA   $2AØ2
ØF98: B9 ØØ 4Ø 99 ØØ AØ C8 DØ   $9FB7
ØFAØ: F7 EE 9A ØF EE 9D ØF AD   $185D
ØFA8: 9A ØF C9 6Ø DØ EA EA A9   $9A2A
ØFBØ: 4C 8D ØØ Ø3 A9 ØØ 8D Ø1   $ØBBB
ØFB8: Ø3 A9 BC 8D Ø2 Ø3 2C 5Ø   $1AED
ØFCØ: CØ 2C 54 CØ 2C 57 CØ 2C   $AF9C
ØFC8: 52 CØ 4C ØØ Ø3            $33A1
```

You may have noticed that Twerps loads a logo off the disk. The logo loader program uses a special routine which will only work on the copy-protected orignal and, therefore, must be eliminated. So, instead of loading the logo, let's clear the hi-res screen at the start of the program.

The loader is loaded at $BCØ3 (which was moved by us to $5CØ3 to keep it safe when we booted the blank disk) so, let's put our little routine there:

5CØ3:AD 81 CØ A2 2Ø
5CØ8:86 Ø1 A9 ØØ 85 ØØ A8 91
5C1Ø:ØØ C8 DØ FB E6 Ø1 CA DØ
5C18:F6 4C 41 Ø4

This code erases the primary hi-res screen and then jumps to $441, which checks for a keypress and shows a demonstration.

One last point: Twerps will erase the entire memory of the Apple when RESET is pressed, which is disappointing if you are trying to obtain a hi-res dump of the game. Fix it by entering

5C88:2C
5C8B:2C
5C9Ø:2C
5C93:2C
5C98:2C

Because DOS won't normally allow us to save anything over $7FFF bytes long, to make it save up to $FFFF bytes, type

A964:FF

Save the program with

BSAVE TWERPS,A$F8Ø,L$8Ø8Ø

This will leave you with a 131-sector file.

And that's it. The program will start when you hit the spacebar.

If you are having trouble with your controls, don't forget to type

CTRL P CTRL V

if you have a joystick. Have fun, and...sorry about the floating eyeballs!

Are you a NEW SUBSCRIBER?

BACK ISSUES of Hardcore COMPUTIST and ★CORE are PACKED with information that you won't want to miss!

Hardcore COMPUTIST 20: Softkeys for Sargon III, Wizardry: Proving Grounds of the Mad Overlord and Knight of Diamonds, The Report Card v1.1 and Kidwriter / Apple][Boot ROM Disassembly / The Graphic Grabber v3.0 / Copy II + 5.0: A Review / The Know-Drive- A Hardware Evaluation / An Improved BASIC/Binary Combo

Hardcore COMPUTIST 19: Softkeys for Rendezvous With Rama, Peachtree's Back To Basics Accounting System, HSD Statistics Series, Arithmetickle, Arithmekicks and Early Games for Children / Double Your ROM Space / The Games of 1984: In Review-Part II / Towards a Better F8 ROM / The Nibbler: A Utility Program to Examine Raw Nibbles From Disk

Hardcore COMPUTIST 18: Softkeys for the Scholastic Version of Bank Street Writer, Applewriter //e, SSI's Non-RDOS Disks, BPI Accounting Programs and DesignWare Programs / Installing a Free Sector Patch Into Applewriter //e / The Games of 1984: In Review / 65C02 Chips Now Available / Checksoft v2 / Simple Copy Protection

Hardcore COMPUTIST 17: Softkeys for The Print Shop, Crossword Magic, The Standing Stones, Beer Run, Skyfox, and Random House Disks / A Tutorial For Disk Inspection and the Use Of Super IOB / The Graphic Grabber For The Print Shop / The Lone Catalog Arranger Part Two / S-C Macro Assembler Directives (Reprint)

Hardcore COMPUTIST 16: Softkeys for Rescue Raiders, Sheila, Basic Building Blocks, Artsci Programs, Crossfire, Sensible Speller for ProDOS and Sideways / Secret Weapon: RAMcard / The Controller Writer / A Fix For The Beyond Castle Wolfenstein Softkey / The Lone Catalog Arranger Part 1

Hardcore COMPUTIST 14: Super IOB v1.2 Update / Putting Locksmith 5.0 Fast Copy Into a Normal Binary File / Softkeys for Seadragon, Rocky's Boots, Knoware, PFS Software, Computer Preparation SAT & MatheMagic / Batman Decoder Ring / REVIEW: Boulder Dash / A Fix For DiskEdit

Hardcore COMPUTIST 13: Softkeys for Laf Pak, Beyond Castle Wolfenstein, Transylvania and The Quest, Electronic Arts, Snooper Troops (Case 2), DLM Software, Learning With Leeper, & TellStar / CSaver: The Advanced Way to Store Super IOB Controllers / Adding New Commands to DOS 3.3 / Fixing A ProDOS 1.0.1 BSAVE Bug / REVIEW: Enhancing Your Apple / Locksmith 5.0 and the Locksmith Programming Language.

Hardcore COMPUTIST 11: Copy II Plus 4.4C Update / PARMS for Essential Data Duplicator / Ultimaker III / Mapping of Ultima III / Ultima II...The Rest of the Picture / Softkeys for Sensible Speller, Ultima III, Softporn Adventure, The Einstein Compiler v5.3, & Mask of the Sun

Hardcore COMPUTIST 10: Controller Saver / Softkeys for The Arcade Machine, Bankstreet Writer, Minit Man, Sensible Speller IV, Essential Data Duplicator III, Krell LOGO, & Canyon Climber, ApplEar / REVIEWS: The 65SC802 & 65SC816 Chips and Dino Eggs / Examining Protected Applesoft Basic Programs / Crunchlist II / Parms for DB Master v4

Hardcore COMPUTIST 7: Softkeys for Zaxxon, Mask of the Sun, Crush, Crumble & Chomp, Snake Byte, DB Master, & Mouskattack / Making Liberated Backups That Retain Their Copy Protection / S-C Assembler: Review / Disk Directory Designer / COREfiler: Part 1 / Upper & Lower Case Output for Zork.

Hardcore COMPUTIST 6: Softkeys for Pandora's Box, Donkey Kong, Caverns of Freitag, & Visifile / Program Enhancements: Quick.Bug / Personalizing A Program / Modified ROM's / Data Bases / Data Base Management Systems / Crunchlist / Essential Data Duplicator: Review / The CIA: Review.

Hardcore COMPUTIST 4: Ultima II Character Editor / Softkeys for Ultima II, Witness, Prisoner II, & Pest Patrol / Adventure Tips for Ultima II & III / Copy II Plus PARMS Update

Hardcore COMPUTIST 1: Softkeys for Data Reporter, Multiplan & Zork / PARMS for Copy II Plus / No More Bugs / APT's for Choplifter & Cannonball Blitz / Reviews: Replay, Crackshot, Snapshot & Wildcard copy cards

CORE 3 Games: Constructing Your Own Joystick / Compiling Games / GAME REVIEWS: Over 30 of the latest and best / Pick Of The Pack: All-time TOP 20 games / Destructive Forces / EAMON / Graphics Magician and GraFORTH / and Dragon Dungeon

CORE 2 Utilities: Dynamic Menu / High Res: Scroll Demo / GOTO Label: Replace / Line Find / Quick Copy: Copy

CORE 1 Graphics: Memory Map / Text Graphics: Marquee, Boxes, Jagged Scroller / Low Res: Color Character Chart / High Res: Screen Cruncher, The UFO Factory / Color / Vector Graphics:Shimmering Shapes, A Shape Table Mini-Editor / Block Graphics: Arcade Quality Graphics for BASIC Programmers / Animation.

(★CORE is no longer published as an independent quarterly magazine. Back issues not listed are no longer available.)

Send me the back issues indicated below:

HC 20 .... $4.75 □	HC 14 .... $4.75 □	HC 6 .... $4.75 □
HC 19 .... $4.75 □	HC 13 .... $4.75 □	HC 4 .... $4.75 □
HC 18 .... $4.75 □	HC 11 .... $4.75 □	CORE 3 Graphics .... $4.75 □
HC 17 .... $4.75 □	HC 10 .... $4.75 □	CORE 2 Utilities .... $4.75 □
HC 16 .... $4.75 □	HC 7 ..... $4.75 □	CORE 1 Games .... $4.75 □

Name ______________________ ID# __________

Address ______________________

City ______________ St ________ Zip ________

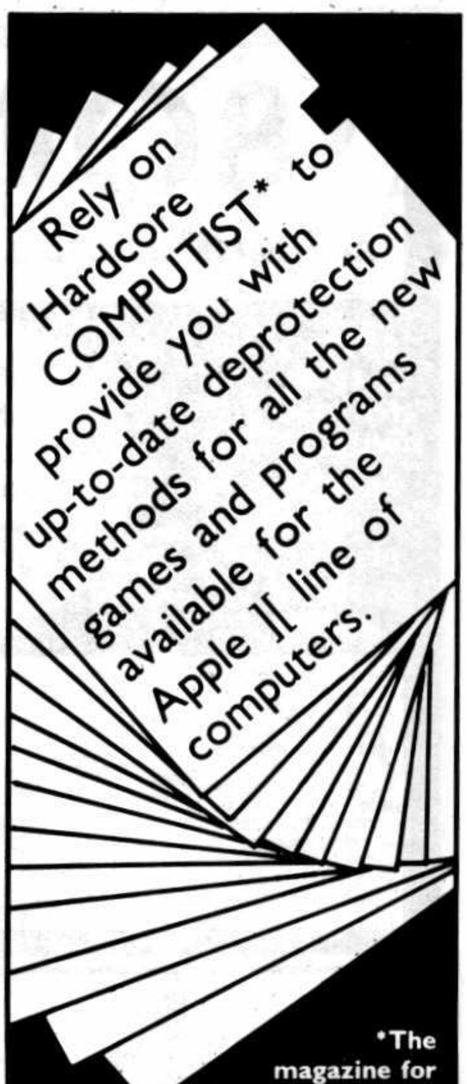
Phone ______________________

VISA/MC ______ - ______ - ______ Exp ________

Signature ______________________

Send check or money order to: Hardcore COMPUTIST, PO Box 110846-B, Tacoma, WA 98411. Most orders shipped UPS. Please use street address. Washington residents add 7.8% sales tax. Foreign orders add 20% shipping and handling. US funds drawn on US bank.

Void after September 30, 1985



Rely on Hardcore COMPUTIST* to provide you with up-to-date deprotection methods for all the new games and programs available for the Apple][line of computers.

*The magazine for the serious user of Apple][computers.

Subscription rates (for six issues):

Please check one

□ US	$20
□ US 1st Class	$24
□ Canada	$34
□ Mexico	$39
□ Foreign	$60
□ SAMPLE- US	$4.75
□ SAMPLE- Foreign	$5.75

( ) Yes! Start my subscription now.

Name ______________________

Address ______________________

City ______________ St ________ Zip ________

Country ______________________

VISA/MC ______ - ______ - ______ Exp ________

Signature ______________________

Send check or money order (US funds drawn on US bank) to: Hardcore COMPUTIST, PO Box 110846-T, Tacoma, WA 98411.

80¢ per disk?

*And that **INCLUDES** the Tyvek sleeve, too?*

SS/DD 5 1/4" for use with:

Apple
Commodore
Atari
Epson
TRS-80

25 Verbatim disks For only $20.00

- Sold in multiples of 25 ONLY
- 100% guaranteed, with reinforced hubs
- Tyvek sleeves & write protect tabs included

Send your order to: Disk Offer, SoftKey Publishing, PO Box 110816, Tacoma, WA 98411. Please enclose $3 shipping and handling for the first set PLUS $1 for each additional set. Foreign orders add 20% shipping & handling; US funds drawn on US bank. Washington state residents add 7.8% sales tax. Orders shipped via UPS; please use street address.

Please send me ________ sets (25 disks per set) of 5 1/4" SS/DD diskettes. I have enclosed $20.00 PLUS shipping & handling for each set.

Name ________
Address ________
City ________ St ________ Zip ________
Country ________
VISA/MC ________ Exp ________
Signature ________

Moving Soon?

Let us know your NEW ADDRESS **at least 30 days in advance** so you won't miss a single issue of Hardcore COMPUTIST (issues missed due to non-receipt of change of address may be obtained at the current back-issue rate).

Fill out your new address on the form provided, paste your present address label in the space below and send it to:
Hardcore COMPUTIST,
Subscription Dept., PO Box
110846-T, Tacoma, WA 98411.

My new address is:

Address ________
City ________ St ________ Zip ________
Phone ________

Paste **present address label here** or fill in old address:

Name ________
Address ________
City ________ St ________ Zip ________
Country ________
Phone ________

MORE BACK ISSUES!

For Hardcore COMPUTIST No's. 5, 8, 9 & 12!

And we thought they were all sold-out!

Hardcore COMPUTIST 5: Softkeys for Bill Budge's Trilogy, Bag of Tricks (fix), Egbert II Communications Disk, Starcross, Hard Hat Mack, The Home Accountant, Dark Crystal, Screenwriter II, Visifile, Lancaster, Amper-Magic, Sammy Lightfoot and Buzzard Bait.

Hardcore COMPUTIST 8: Softkeys for Robotron, Legacy of Lyllgamyn, The Artist, Data Factory v5.0, Essential Data Duplicator, The Spy Strikes Back, Hayden Software and Apple LOGO / Corefiler Formatter / ProDOS Data Encryptor / Best of the Bit Copiers: A Comparison.

Hardcore COMPUTIST 9: Softkeys for Sensible Speller, Sierra On-Line Software, The Visible Computer: 6502, Visidex, Music Construction Set, Gold Rush, Visiterm, & Cosmic Combat / Super IOB / Pro-DOS to DOS: Single Drive Conversion Technique / Using ProDOS on a Franklin Ace / CORE Word Search Generator / The Visible Computer vs. Apple][- 6502 ALT: REVIEW.

Hardcore COMPUTIST 12: Softkeys for Zoom Grafix, Flip Out, Lion's Share, Music Construction Set, Hi-Res Computer Golf II, Suicide, Sabotage, Millionaire, Time Is Money, & Type Attack / Psychedelic Symphony / The CORE Disk Searcher / The Armonitor / Pseudo-ROMS on the Franklin Ace.

ORDER NOW!

First come/first serve while they last.

(See our ad on page 31 to order other back issues)

Please rush me the BACK ISSUES marked below. I have enclosed $4.75 (plus applicable sales tax and shipping) for each issue requested.

□ No. 5 □ No. 8 □ No. 9 □ No. 12

Name ________
Address ________
City ________ St ________ Zip ________
Phone ________
VISA/MC ________ Exp ________
Signature ________

Send check or money order to: Hardcore COMPUTIST, PO Box 110846-B, Tacoma, WA 98411. Most orders shipped UPS; please use street address. Washington residents add 7.8% sales tax. Foreign orders add 20% shipping & handling. US funds drawn on US bank. (Offer good for a limited time only.)

NEW DON LANCASTER RELEASES

Available ONLY from Synergetics. All software open and unlocked.

ABSOLUTE RESET (IIc/old IIe/new IIe) $19.50

Get back in total control. No more hole blasting! Access any part of any program at any time for any reason. Passes all diagnostics. Invisible till activated. Includes EPROM burner adapter details, service listings, and a free bonus book.

APPLEWRITER™ IIe TOOLKIT $39.50

EIGHT diskette sides include a complete disassembly script, self-prompting glossaries, Diablo microjustify and proportional space, patches for NULL, shortline, IIc detrashing, answers to lots of help line questions, source code capturing, two columns, WPL secrets, space on disk, keyword indexing, more.

ProDOS APPLEWRITER™ 2.0 TOOLKIT $39.50

Another EIGHT diskette sides crammed full. Includes a neat automatic source code capturer, a complete disassembly script, dozens of patches, prefix hassle fixes, Grappler and shortline repairs, NULL diversion, two columns, auto indexing, plus Don's unique self-prompting glossaries, Diablo microjustify, etc.

Both TOOLKITS $59.50

APPLEWORKS™ DISASSEMBLY SCRIPT $49.50

TEN diskette sides chock full of Applework's innermost secrets, using Don's unique and powerful "tearing method". Primarily for gonzo hackers and definitely NOT for the faint of heart.

COMPANION DISKETTES:

For Enhancing Your Apple II, Volume I	$19.50
For Enhancing Your Apple II, Volume II	$19.50
For Don Lancaster's Assembly Cookbook	$19.50

ASSORTED GOODIES:

VAPORLOCK instant sync package	$19.50
Old Fangled Animation demo	$ 9.50
The Incredible Secret Money Machine (autographed)	$ 7.50
Laserwriter/Applewriter Utilities Package	$39.50

SYNERGETICS
746 First Street
Box 809-HC
Thatcher, AZ, 85552

FREE
VOICE HELPLINE
(602) 428-4073

VISA and MASTERCHARGE accepted. Please - no COD, foreign, or purchase orders.
Appleworks, Applewriter, and ProDOS are registered trademarks of Apple Computer.

At the dead center of the
Milky Way galaxy lies a lone planet.
It does not orbit a sun--
ALL SUNS ORBIT IT.

The multitude of stars exert
powerful forces upon the planet-
twisting light, tides . . . even the
laws of science itself!

WELCOME to the world of

EAMON*

* A text-adventure in the public domain

With the EAMON Introductory Offer you get six text-adventures for the Apple: *the Beginner's Cave, the Orb of Polaris, Operation Crab Key, the Feast of Carroll, EAMON Utility 1 and the Dungeon Designer v6.0.* Public Domain software is not commercial quality and is supplied as-is. (The Computer Learning Center fills orders on double-sided disks.)

To order, send check or money order (US funds drawn on US bank) for $20.00 to: Computer Learning Center, Eamon Special, PO Box 110876-HC, Tacoma, WA 98411. Washington residents add 7.8% sales tax. Foreign orders add 20% shipping and handling.

Computer Learning Center
Tacoma, WA

warning

You are entering DANGEROUS territory.

The

Book Of Softkeys

TEARS INTO **copy-protection,**
RIPS AWAY **the mystery of**
making backups for
commercial software and
BEATS THE COMPETITION
when it comes to
providing information for the
serious Apple][computer user!

If you want to make BACKUPS,
then YOU WANT
The Book Of Softkeys Vol. I

(compiled from 8 previous issues of Hardcore Computing and COMPUTIST magazines)

Send $20.00 (US funds only) TODAY to: The Book Of Softkeys Vol. I, PO Box 110846, Tacoma, WA 98411. Washington state residents add 7.8% sales tax; foreign orders add 20% shipping and handling. VISA/MC also accepted (signature and expiration date required).

The ONLY resource that teaches you
how to BACKUP your expensive Apple][software!