For The Serious User Of Apple][Computers

Hardcore

COMPUTIST

Issue No. 25 $3.75

Softkeys For:

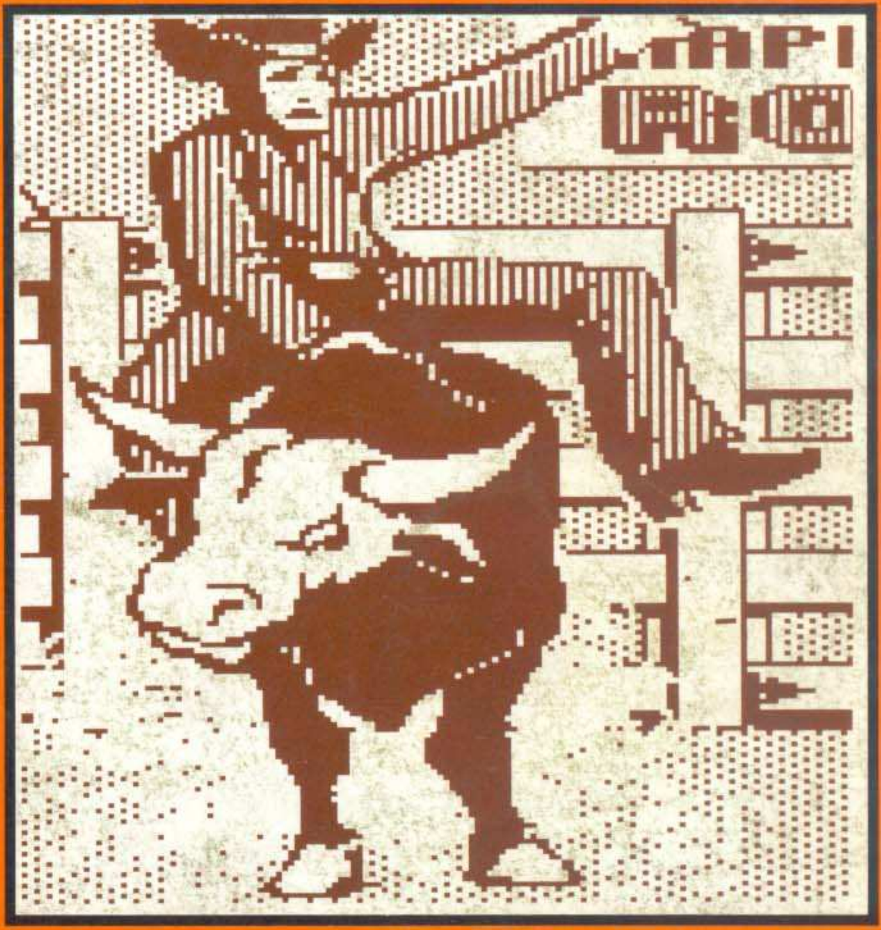
DB Master 4.2 & 4+
Computer SAT
Take 1
Bank Street Speller
Carmen Sandiego
Bank Street Writer //c
Word Challenge

Core:

DOS to ProDOS and back

Feature:

Adding "If Then Else" to Applesoft



Hardcore COMPUTIST
PO Box 110846-T
Tacoma, WA 98411

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.

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 special 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$\boxed{\square\text{P}}$

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 has most recently appeared in Hardcore COMPUTIST No. 22. 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. To get the latest version of this program, you may order Hardcore COMPUTIST No. 22 as a back issue or order Program Library Disk No. 22.

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 (Synergetics; 746 First Street; Box 809-HC; Thatcher, AZ 85552; free voice HelpLine 602-428-4073) 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 (▵). 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 $\boxed{\square\text{Y}}$.

xxx.xxx$\boxed{\square\text{Y}}$

And correct the lines at which the checksums differ.

this is... PAGE ONE

coming soon...

DEAD OR ALIVE

The Most Wanted List comes alive as separated posters scattered through the pages of the COMPUTIST. We're not just wasting space, we're offering bounties! You'll see lots of **WANTED posters with publicly announced REWARDS for softkeys to certain not-yet-unlocked software! Be a bounty hunter and earn your rewards now by becoming a hard-core COMPUTIST softkey author.**

TIME TO RENEW?

Check your mailing label to see if you need to renew your subscription. And if you think you might forget when that fatal time arrives, renew right now. Just use the order blank below.

BACK IT UP OR BUST

If you aren't a subscriber, then maybe you need to find out more about the plague of copy-protection that is turning your valuable software library into a collection of inpenetrable, mysterious black boxes that stop working at the wrong times when the nearest backup is weeks and dollars away.

NEW SUBSCRIBERS!

You've just joined the ranks of concerned software users who are ensuring that the software they use is copy-corrected, copyable, and user-useful. No need to depend on the inflated back-up costs charged by the makers of your software, make your own back-up, and even remove the often time-consuming boot-up delays caused by copy-protection. In other words, Welcome aboard. You'll find every issue packed with softkeys, hardcore user and programmer articles, and even some relaxing entertainment.

IF YOU'RE MOVING...

Let us know right away or at least 30 days in advance so that you won't miss a single issue. Just Write your new address here, and paste your present address label in the order form below and send it to us.

My new address is: ____________________

City ____________ State ________ Zip ________

Phone ____________________

USER SURVEY

☐ I Love Copy-protection and I hope that you naughty hackers stop publishing your rag!

☐ I want to make legitimate copies of the software that I own but I am unable to do so.

☐ I want to get rid of any and all computer code that prevents me from using my property the way I want to use it.

☐ I hate Copy-protection and will do anything in my power to get rid of this scourge.

☐ I find that locked-up disks present an intellectual challenge and user-barrier that I must overcome.

☐ I am a software pirate.

☐ ____________________

____

____

____

____

All purpose Subsciption, Renewal, change-of-address, and survey form. HC25

Name ____________________

Address ____________________

City ____________ State ________ Zip ________

Phone ____________________

Signature ____________________

VISA/MC ______ - ______ - ______ - ______ Exp. ________

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

☐ RENEW subscription
☐ NEW subscriber

SUBSCRIPTION RATES:
(for 6 issues)

☐ US $20
☐ US 1st class $24
☐ Canada, Mexico $34
☐ Foreign $60

SAMPLE COPY:
☐ US $4.75
☐ Foreign $5.75

Christmas Specials

$5 OFF $5

The Book of Softkeys Volume I

This coupon is valid only for current subscribers of Hardcore COMPUTIST. This coupon can be applied to the purchase of the product named, resulting in $5 off the current price only. Expires December 31, 1985

see advertisement on the back cover

$5 OFF $5

The Best of Hardcore Computing

This coupon is valid only for current subscribers of Hardcore COMPUTIST. This coupon can be applied to the purchase of the product named, resulting in $5 off the current price only. Expires December 31, 1985

see advertisement on the inside of the back cover

$2 OFF $2

The Diskbusters T-shirt

This coupon is valid only for current subscribers of Hardcore COMPUTIST. This coupon can be applied to the purchase of the product named, resulting in $2 off the current price only. Expires December 31, 1985

see the advertisement on the inside of the back cover

3 for $20

Hardcore Computist Library Disks

This coupon is valid only for current subscribers of Hardcore COMPUTIST. This coupon allows you to purchase three (3) Hardcore Computist Library disks for only $20. Expires December 31, 1985

see advertisement on page 32

IF

you're a subscriber,

THEN

you can

SAVE

lots of $$$

ELSE

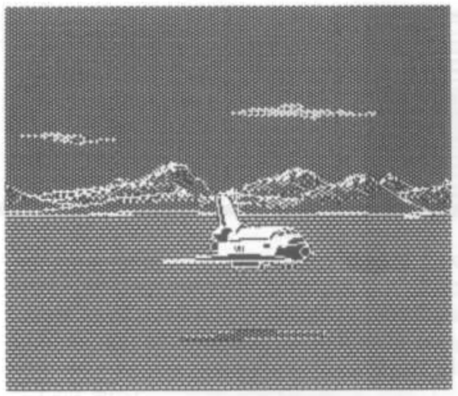
sorry, no savings

OR

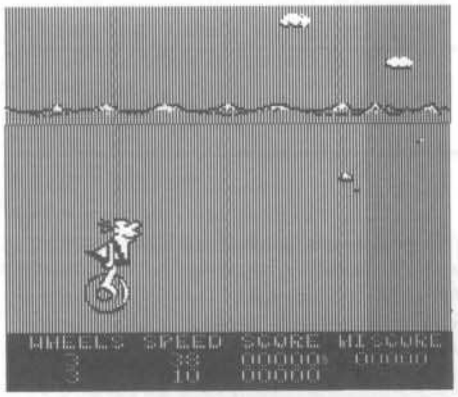
You can subscribe!

INSTRUCTIONS TO SUBSCRIBERS

Just include these coupons with your regular order (see appropriate advertisement in this or last issue) along with your subscriber number or mailing sticker and deduct the appropriate savings. These coupons are good till the end of 1985, so if you're not yet a subscriber, or you forgot to renew, use the form on *page one* and you'll be a subscriber, too. Then you, too, can take advantage of these holiday savings!



Kigali, the largest city in the east central African country of Rwanda, has a population of 150,000.



This month's cover: Graphics from DLM's Shape & Color Rodeo.

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.

SUBSCRIPTIONS: Rates (for 6 issues): U.S. $20, U.S. 1st Class $24, Canada & Mexico $34, 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.

Hardcore Issue 25

COMPUTIST

Publisher/Editor: Charles R. Haight **Managing Editor:** Ray Darrah
Technical Editor: Robert Knowles **Circulation:** Michelle Frank, Debbie Koval
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

input

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.

Squire Meets Millionaire

Mr. A. L. Head's letter on deprotecting *Millionaire* in Hardcore COMPUTIST No. 21 gave me the clue I needed to deprotect *Squire* and *Baron*. The necessary information is contained in the Basic file CDIINC. The file CDIINC contains inverse characters after each non-inverse letter. (Inverse C follows C, Inverse D follows D, etc.) By reading this file with *Copy][Plus*, I discovered that *Squire* requires a MAXFILES 5 command, followed by BRUN SQROOT.X, and that *Baron* uses MAXFILES 6, followed by RUN BARINIT.BAS.

Baron Step By Step

1) Init 2 sides of a disk with a Basic program named HELLO.

2) Load the system master, type NEW, and enter the following program:

```
10 PRINT CHR$(4) "MAXFILES 6"
20 PRINT CHR$(4) "RUNBARINIT.BAS"
```

3) Insert the front side of your previously INITted disk.

4) SAVE HELLO.

5) Using *Fid* or *Copy II+*, copy all files except ENTRA, CON, and CDIINC.

6) Turn the initted disk over and copy all files from the second side.

Squire Step by Step

To deprotect *Squire* follow steps 1-6, except that the HELLO program in step 2 should read as follows:

```
10 PRINT CHR$(4) "MAXFILES 5"
20 PRINT CHR$(4) "BRUNSQROOT.BAS"
```

Also, most EDUWARE programs can be deprotected using SUPER IOB 1.5 with the NewSwap controller installed. Except for *Rendezvous*, the copied programs will run without further alteration.

Charles Taylor
Los Angeles, CA

Spiradisk Info

Responding to my article in Hardcore COMPUTIST No. 21 on quarter tracks, M. M. McFadden informed me of the existence of the so-called "spiradisk" that uses quarter-track increments for track arcing. By obtaining a copy of such a disk (*Frogger* by Sierra ON-LINE) I was able to verify that he was right. Since parameters using quarter tracks can only copy disk data that is a half-track apart, the "spiradisk" cannot be copied by the current bit-copiers. Fortunately, the new Apple //c hardware implements a delay between commands to the stepper magnets which prevent one from accessing quarter tracks accurately. Thus the "spiradisk" has gone the way of DOS 3.2, thank goodness. Please refer to my article in Hardcore COMPUTIST No. 21 for more information on quarter tracks.

Bruce W. Jones
Pueblo, CO

Setting a Mountain Clock

A year and a half ago, before I moved to Brazil, I purchased a Mountain Clock Card for my Apple //e. I got this on sale and it came with no software to set the date and time. I have had it all this time with the wrong date and no way to set it. Does anyone know how to set this clock? And, if so, put a letter in this column with a short program or something to set the clock. Any help or information would be greatly appreciated. Thanks for any help. Keep up the good work with the best magazine available to the Apple user.

Greg Poulos
Detroit, MI

Mr. Poulos: *While I cannot offer you a program for setting your clock, I can offer some information. First of all, to turn off the other cards and turn on your mountain card, do an "X = PEEK (53247) + PEEK (49152 + SLOT *256) where SLOT is equal to the slot where your card is. You must then POKE the values for the time you want into memory addresses 38304 ($95A0) through 38315 ($95AB) and do a "CALL 38316".*

Handling the New Handlers

In partial reply to Michael Mullin's problems with *The Handlers*; a letter to ALS at 1195 E. Arques Ave, Sunnyvale, CA 94086 and a check for $15.00 sent with his old *Word Handler* disk will get him an updated version of that great little word processor. The new version is copyable with *Copy][Plus v5.0* with the built-in auto-copy routine. He may want to call technical assistance at 1-408-730-0307 between 9:00 and 3:00 to get authorization to have the whole *Handlers* package updated.

If he gets the new package he will find a new "Tutorial Disk". When he boots the backside called, "Tutorial Documents", he will get a utility to convert the proprietary files to DOS 3.3 text files for transmission by modem.

Keep the magazine coming. Right now I am especially enjoying the *Graphic Grabber v3.0* from Hardcore COMPUTIST Issue No. 20. Lots of great programs and tips.

P.S. I am still looking for help making backup copies of *Milliken Comprehensive Power Programs* and *Borg Warner's College Entrance Exam Preparation*

D. J. Ward
So. Williamsport, PA

Mr. Ward: *Joel Huse's article on backing up the College Entrance Exam Prep in Hardcore COMPUTIST No. 21 might be what you're looking for.*

input

A Nutritional Softkey

The following is a softkey for *The Dine System*, by D. Dennison, 724 Robin Rd, West Amherst, NY 14228, (716)688-2492.

This is a nutritional plan which is designed to improve personal eating behavior, with an extensive database of over 3500 foods including many brand name items. It is an excellent program, but I desired to modify the basic program to include more factors in determining daily nutrient requirements. The program disk is copyable with a bit copier, but the program cannot be accessed normally.

Using Softmove, a utility which comes with the Replay II card, I was able to move the Applesoft program to standard DOS 3.3. Listing this program allowed me to discover the protection used. There is an additional catalog on track 10, which is accessed by 3 POKE statements:

```
POKE 44033,10
POKE 47445,170
POKE 47455,213
```

After making these pokes in DOS 3.3, LOAD each file from the program disk, return the addresses to their normal values:

```
POKE 44033,17
POKE 47445,213
POKE 47455,170
```

and SAVE the file onto another formatted disk with a normal DOS 3.3 or fast DOS on it.

Then, as you make your modifications to *Dine*, modify all pokes to these addresses to the normal values given above.

I hope this may be useful to someone who needs to modify this already good program.

Jason Chao, MD
Cleveland Hts., OH

A Mickey Mouse Protection

Mickey's Space Adventure by Walt Disney and Sierra On-line is a game designed for beginning adventurers (especially kids), giving the player options to select instead of the standard, two-word input style.

The disk can be copied normally except that the program doesn't quite run. After going through a title page and quite a bit of disk reading, a copy of the program will jump into the monitor at $1702. It sure takes its time before deciding that it is a copy.

However, the copy protection is really quite minimal. By booting Dos 3.3 from a normal disk, *Mickey's Space Adventure* can be CATALOGed. (Which is only the beginning of the simple-mindedness of their protection.)

In the catalog is an Applesoft program with the very suspicious title of BOOT. By loading this program I found that it ran MICKEY-CODE at $4000.

I typed MONICO (so that DOS commands will be echoed to the screen) and ran BOOT. The computer obligingly displayed each loaded program module and, in most cases, even showed where in memory it was loaded. Since MICKEY-CODE was the only BRUN program (all others were BLOADed) I started my search there.

At $4000 there was a JSR (jump subroutine) to a routine that displayed a "PLEASE WAIT" type message, BLOADed the other modules and kindly returned to where it came from. The main routine then set up some pointers and such, called a routine at $1572 (which I considered very close to where the whole thing blew up when booting) and then jumped somewhere.

I removed the JSR $1572 with NOP (no operation) instructions and gave it a 4000G from the monitor. The program loaded without any trouble. Apparently the subroutine at $1572 did some nibble counting on the data that normally brought the boot process to a grinding halt.

Step By Step

1) Copy Mickey's Space adventure with COPYA.

2) Boot DOS 3.3.

3) LOAD BOOT from the copied *Mickey's Space Adventure.*

4) CALL -151 to enter the monitor.

5) 4014:EA EA EA   to remove the subroutine call.

6) BSAVE MICKEY-CODE, A$4000, L9728.

That's it!

Gary Kowalski
Anaheim, CA

P.S. My vote is to keep Hardcore COMPUTIST for the Apple][family only. Too many magazines allocate space for the MAC. There already are magazines for the MAC and more will spring up.

A Quick Pop Into the Monitor

I love your magazine. I'm sending money to renew my subscription. You can reset into the monitor on any apple by using a 100 ohm resistor and connecting it to pins 26 and 29 of any slot. This causes an NMI.

Brian Snook
Riverside, CA

Running the CIA on a //c

We advertise a product called the CIA with your magazine and have done so, for quite some time now. It has come to our attention that a small patch is necessary to run the CIA on the Apple //c.

The patch is given below in the hope that you may print it as a press release in Hardcore COMPUTIST. This would benefit many of your readers who have ordered the CIA from us over the past year or so.

TRICKY DICK:

If program is in memory, change:

B1 to **08** at **$1302**

If program is on disk, change:

B1 to **08** at Track **15** Sector 05 Byte **03**

TRACER:

If program is in memory, change:

50 to **70** at **$8980**

If program is on disk, change:

50 to **70** at Track **1B** Sector **09** Byte **87**

Many thanks in advance.

Golden Delicious Software
Chelsea, England

input

Know of any Knowdrive Prospects?

In Hardcore COMPUTIST No. 2Ø, you have a review of a 128K memory board called "The Know Drive". In this review it was stated that you can get them for less than half-off by getting a user group buy together. Well, I tried to do this at my local user group, but only a few were interested. Therefore, I was wondering if you knew of anybody who is getting a large buy together? If so, please give me their names and addresses, or you can give them mine. If you don't know of any, then please put my name and address in your magazine and I will try to put a buy together.

Also, I have been looking for information on X and K modem protocols but haven't seen a thing. (These are standards in which computers send files over modems). If you people know or can find out about them, why don't you write an article about them in the core section?

Finally, here are a few adventure tips for you on *The Dark Crystal*:

Locked gate giving you a problem, Fizgig does more than grrrr.

Riddle troubles, moss covered rocks help.

Sharp rocks are a big help to get things floating along.

Richard Blair
12252 Manning Place
Medway, OH 45341

Various Cracks

Here's a few programs that I have cracked. You might want to pass them on to your readers.

DINOSAUR DIG by CBS

Use SWAP controller or do a CALL-151, B942:18 and copy with COPYA.

OPERATION FROG by SCHOLASTIC

Same as DINOSAUR DIG.

STELLAR DEFENSE by RAINBOW COMPUTING

To copy this popular *Star Trek* game, first copy nomally with COPYA or LS fast copy.

1) Boot the copy, immediately hit $\boxed{\text{ctrl}\ C}$ or reset.

2) LOAD the program called "DRIVER" into memory.

3) LIST line 5Ø6 of DRIVER, it should read: CALL 15953.

4) Type in 5Ø6 and hit return to erase this line from the program.

5) SAVE DRIVER back to the disk.

You are now the proud holder of an unprotected copy of *Stellar Defense*.

MAGIC SLATE by SUNBURST

To back up this very good word processor, use *Copy][Plus v5.Ø*. Now go back and BIT COPY Track 1 keeping the track length.

William Forsyth
New York, NY

The Quicker Aux

A few issues ago, a method was presented for using the Apple extended 8Ø-column card on an Apple //e for cracking. The article used several small programs to manipulate the auxilary memory. However, it is much easier to do this:

1) Boot Dos 3.3

2) Type

```
CALL -151
Ø:8D Ø3 CØ 8D Ø5 CØ 4C ØØ C6
```

3) Insert disk to boot.

4) Type

```
ØG
```

When the disk is done booting insert a normal DOS disk and hit $\boxed{\text{ctrl}}$ reset. Next, drop into the monitor (CALL -151) and type this:

```
3ØØ:18 4C 11 C3 3F8:4C ØØ Ø3
```

This causes the $\boxed{\text{ctrl}\ Y}$ to move auxilary memory into main memory. Now type

```
8ØØ<96ØØ.BFFF ctrl Y
```

This moves the DOS and RWTS into main memory starting at $8ØØ.

Steve Dietz
Arlington, TX

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.

Mouse Calc	Apple Computer
Apple Business Graphics	Apple Computer
Flight Simulator II	Sub Logic
Factory	Sunburst Communicating
Jane	Arktronics
Bookends	Sensible Software
Visiblend	Microlab
Sundog	FTL Games
Sundog, V2	FTL Games
Lifesaver	Microlab
Catalyst	Quark, Inc.
Gutenburg Jr. & Sr.	Micromation LTD
Prime Plotter	Primesoft Corp.
Zardax	Computer Solutions
The Handlers	Silicon Valley Systems
Milliken Math Series (NEW)	Milliken Publishing
The Apple's Core: Parts 1-3	The Professor
King's Quest	Sierra On-line
Hayes Terminal Program	Hayes
Fun Bunch	Unicorn

readers' softkey & copy exchange

Danny Pollak's Softkey for...

Spy's Demise

Spy's Demise
Penguin Software
830 Fourth Ave.
P.O. Box 311
Geneva, IL 60134

Requirements:
Apple][Plus or equivalent
A blank initialized DOS 3.3 disk
Spy's Demise

Spy's Demise is a game from Penguin Software in which you, the spy, must traverse the floors of the diplomatic mission in Pyongyang and gather pieces to a message which is the key to a fortune in valuable computer data.

The program occupies tracks 0 through 12 with alternating address headers of D5 AA 96 and D4 AA 96. We can use a boot trace to load in the entire file and then exit to the Monitor. Here's how it is done.

The first thing we need to do is to load in Boot1 from the Spy's Demise disk and then re-enter the monitor. Insert the Spy's Demise disk into drive 1 and then enter the following:

```
CALL-151
9600<C600.C700M
96FA:98
9801:4C 59 FF
9600G
```

Boot1 has now been loaded into page 8 of memory. We will now alter our code at $9600 so that it will load Boot1 at $1800 and jump to the Boot1 at $800.

```
9659:18
96FA:08
```

If we examine the code starting at $801, we see that it loads some code into memory, sets up the reset vector, loads the x-register with the value stored at $2B (slot number times 16) and then jumps to the load routine at $B700. We can alter this code so that it will exit to the Monitor before jumping to the routine at $B700.

```
80C:90
84D:FF
84F:59
8A4:4C 59 FF
9600G
```

If we now examine the code at $B700, we see a jump to $6000 at $B734. Alter this code

so that it will exit to the Monitor instead of jumping to $6000 by entering

```
B734:4C 59 FF
```

Load the x-register with $60 and execute the code at $B700 by entering

```
CTRL E
:00 60
B700G
```

The rest of the program should have been loaded and you should now be in the Monitor. The program resides in memory from $4000 to $8DFF. We will fill memory from $8E00 to 8EFF with $00's and save this extra page of memory with the program. The extra sector that results in our binary file will be used to store the high scores. Boot the initialized disk and then enter the following:

```
CALL-151
3F04:AD EC B7 8D
3F08:EE 80 AD ED B7 8D EF 80
3F10:AD 50 C0 AD 52 C0 AD 55
3F18:C0 AD 57 C0 4C 00 60

8E00:00
8E01<8E00.8EFEM
80D4:20 00 BD 60
BSAVE SPY'S DEMISE,A$3F04,L$4FFB
```

And there you have it. A completely unprotected copy of Spy's Demise. Here are a couple of APTs you can use.

Start with any number of spies:

```
60AB:(number of men)
```

Infinite number of spies:

```
6D8A:EA EA
```

These are neither all of the APTs, nor are they the best. Have fun finding your own and good luck on decoding the message.

Michael A. Coffey's Softkey for...

Mind Prober

Mind Prober
Human Edge Software Corp.
2445 Faber Place
Palo Alto, CA 94303
$49.95

Requirements:
Apple][
COPYA or Super IOB
Both sides of a blank disk
Mind Prober disk

Mind Prober is a fascinating research based program that provides you with a profile of a person. The outcome of the profile depends on your accuracy in answering the questions. The results can leave you confounded in their preciseness or seem so far out of line so as to be laughable. In using the program, my subjects felt that most of the conclusions were correct, sometimes to the point of being frightening. I suspect that it is the use of the program on those you don't know well, or those who lie when answering the questions that accounts for the occasional "it's only a party game" attitude.

One fault I have with the program is the $9.95 charge to provide you with a backup. Since I'd rather make my own anyway, I set about to do so. The program appears to be protectionless, and none of the popular bit copiers seem to work. Nevertheless a backup can be made can be made with little trouble following these steps:

1) Make a copy of both sides of Mind Prober using either COPYA or Super IOB with the standard controller.

2) The turnkey program is called HELLO and it is this we must modify. Under normal DOS and side one of the copy of the drive, type

```
UNLOCK HELLO
LOAD HELLO
```

3) Get rid of all the lines before 4000 by typing

```
DEL 3012,3060
```

4) Save and lock the new program.

```
SAVE HELLO
LOCK HELLO
```

You now have an enjoyable and deprotected disk. Happy profiling.

readers' softkey & copy exchange

Danny Pollak's Softkey for...

BC's Quest For Tires

B.C.'s Quest For Tires
Sierra On-Line, Inc.
10398 Rockingham Dr., Ste. 12
Sacramento, CA 95827
$34.95

Requirements
48K Apple][
B.C.'s Quest For Tires
A blank disk

B.C.'s Quest For Tires is a game from Sierra On-Line. Unfortunately, like their other software, B.C.'S Quest For Tires is copy protected. I have not been fortunate enough to see the "Sierra On-Line softkey" which I have seen mentioned in several of the latest issues of Hardcore COMPUTIST. For that reason, here is my method of breaking Quest For Tires.

The copy protection used on B.C.'s Quest For Tires is the same as that on the other games by Sierra On-Line I have seen. Hopefully, this method can be used to unlock other Sierra On-Line software.

1) Prepare a slave disk with a deleted HELLO program. If you want, use the name BC'S QUEST FOR TIRES.

```
INIT HELLO
DELETE HELLO
```

2) Boot the original B.C.'s Quest For Tires disk. When the prompt appears, hit the reset key (you have to be fast). If this does not put you into Applesoft, then repeat step 2 until you do get in.

3) Get some of the code from the disk and move $800-8FF out of the way.

```
BLOAD X,A$4000,L$2000,B$0B00
BLOAD Y,A$0800,L$3800,B$0708
CALL-151
6000<800.8FFM
```

4) Insert your initialized disk in drive one and reboot.

```
PR#6
```

5) Restore $800-8FF and save to the new disk.

```
CALL-151
800<6000.60FFM
BSAVE XY,A$800,L$5800
```

6) Repeat step 2.

7) Get some more code from Quest For Tires.

```
BLOAD Z,A$6000,L$3A00,B$0300
CALL-151
1600<9600.99FFM
BLOAD  W,A$2000,L$4000,B$0D00
```

8) Insert your initialized disk in drive one again.

```
PR#6
```

9) Type in the following:

```
BSAVE W,A$2000,L$4000
MAXFILES1
CALL-151
9600<1600.19FFM
BLOAD XY
7FD:4C F5 19
```

```
97A0:AD 55 C0 20 A9 97 4C CA
97A8:97 A0 00 B9 B7 97 20 ED
97B0:FD C8 C0 10 D0 F5 60 8D
97B8:84 C2 CC CF C1 C4 A0 C2
97C0:C3 AE C8 C9 C7 C8 8D

97CD:4C EB 97

97E8:EA EA EA
```

10) Tell DOS to allow binary files up to $FFFF in length so you can save a huge file.

```
A964:FF
BSAVE BC.OBJ,A$7FD,L$9203
```

11) Hang in there, you're nearly done. Type in the following:

```
BLOAD W
2031:4C DF 20
```

```
20AA:20 B0 20 4C DF 20
20B0:A0 00 B9 BE 20 20 ED FD
20B8:C8 C0 1C D0 F5 60 84 C2
20C0:D3 C1 D6 C5 C2 C3 AE C8
20C8:C9 C7 C8 AC C1 A4 B2 B0
20D0:B0 B0 AC CC A4 B4 B0 B0
20D8:B0 8D

20EC:A9 19 8D 86
20F0:1C A9 1A 8D 06 1C
```

```
BSAVE BC.HIGH,A$2000,L$4000
```

11) Enter this Applesoft program.

```
FP
```

```
10 D$ = CHR$( 13) + CHR$( 4)
20 HGR : POKE -16302 ,0
30 PRINT D$ "MAXFILES1"
40 PRINT D$ "BRUN^ BC.OBJ"
```

```
SAVE BC'S QUEST FOR TIRES
```

12) Delete the extra files you made, leaving only BC'S QUEST FOR TIRES, BC.OBJ, and BC.HIGH.

```
DELETE XY
DELETE W
```

Now you are off and rolling with a totally unprotected version of B.C.'s Quest For Tires. You can also put the files on your favorite game disk. A fast DOS will help immensely.

readers' softkey & copy exchange

Greg Prior's Revisted Softkey for...

DLM Software

Development Learning Materials
1 DLM Park
Allen, TX 75ØØ2

Requirements:
Apple][Plus or equivalent
Minus Mission or Meteor Multiplication
Super IOB v1.2
A blank disk

In Hardcore COMPUTIST No. 13, a DLM softkey appeared that doesn't seem to work on the new releases of DLM software. Could it be that DLM might have a subscription to Hardcore COMPUTIST and changed the protection scheme on at least some of their disks? Specifically, the ones I found changed were Minus Mission and Meteor Multiplication.

It would seem that they have dropped the 13 sector format and have altered the address and data epilogues from the normal DE AA to DA AA. Now, This would seem to be an alteration that wouldn't be worth the trouble, until a slick little utility like Bag of Tricks returns "unable to interpret data" on at least half of the tracks. This is caused by the first byte of either the address field or data field prologues appearing to be sync bytes. Hmm, it seems that they're a bit more devious than appears at first glance.

Anyway, type in the Super IOB controller that follows (by the way, it was generated by the Controller Writer from Hardcore COMPUTIST No. 16), and proceed as follows:

1) Install the Controller into Super IOB using your favorite method.

2) INITialize a blank disk with a normal or fast DOS using the name of the original disk as the hello file. (ex. for Minus Mission, type INIT MINUS MISSION)

3) Run Super IOB to make the copy onto the freshly INITialized disk and presto-chango! a COPYAable DLM program.

If you look at the controller, you will notice that tracks Ø through 2 are not copied from the original disk. We don't really want to copy their DOS, do we?

Note: if this method does not work for these or other DLM programs, you can try using the Swap controller and the RWTS from the offending disk.

controller

```
1000 REM NEW DLM CONTROLLER
1010 TK = 3 : LT = 35 : CD = WR : MB = 151 : ONERR GOTO
     550
1020 ST = 0 : T1 = TK : GOSUB 490 : RESTORE : GOSUB
     190 : GOSUB 210 : GOSUB 170
1030 GOSUB 430 : GOSUB 100 : ST = ST + 1 : IF ST <
     16 THEN 1030
1040 IF BF THEN 1060
1050 ST = 0 : TK = TK + 1 : IF TK < LT THEN 1030
1060 GOSUB 230 : TK = T1 : ST = 0 : GOSUB 490
1070 GOSUB 430 : GOSUB 100 : ST = ST + 1 : IF ST <
     16 THEN 1070
1080 ST = 0 : TK = TK + 1 : IF BF = 0 AND TK < LT THEN
     1070
1090 IF TK < LT THEN 1020
1100 HOME : A$ = "ALL^ DONE" : GOSUB 450 : END
5000 DATA 213 , 170 , 150 , 213 , 170 , 173 , 218 , 170
     , 218 , 170
```

controller checksums

```
1000  - $356B    1060  - $6AE6
1010  - $5E3F    1070  - $22FD
1020  - $B92C    1080  - $54D8
1030  - $E2AA    1090  - $7FC2
1040  - $2463    1100  - $F952
1050  - $E2BC    5000  - $EF9F
```

Nick Galbreath's softkey for...

Homeword Speller

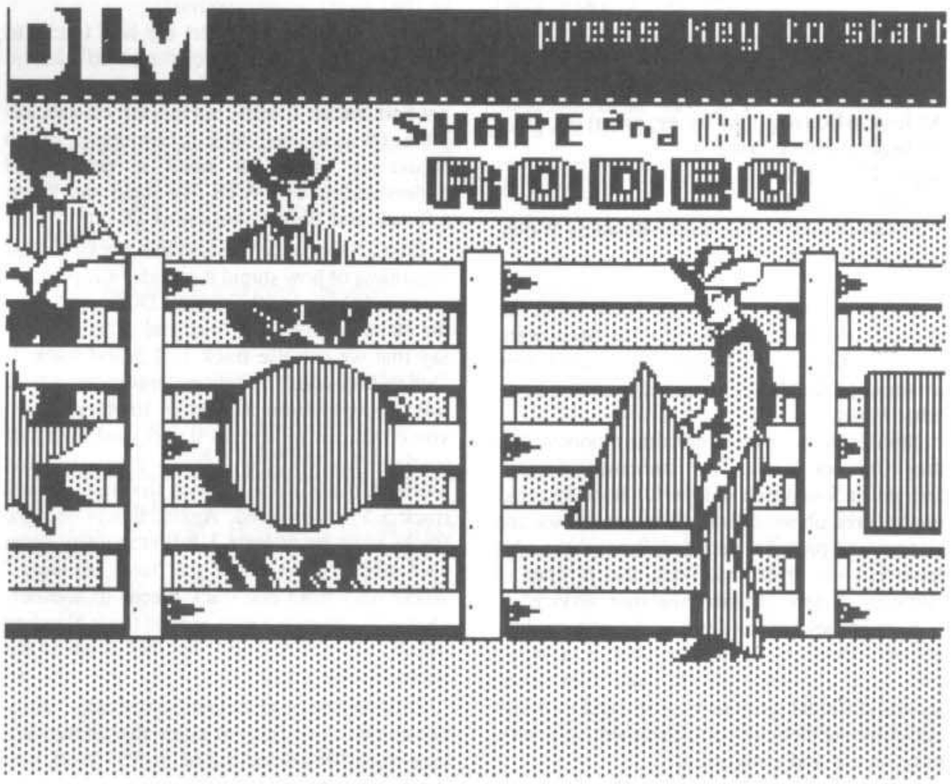
Homeword Speller
Sierra On-Line Systems
36575 Mudge Ranch Rd.
Coarsegold, CA 93614

Requirements:
Apple][
COPYA or equivalent sector copier
A sector editor

To crack the spelling checker, copy the disk with COPYA, then get out a sector editor and change the following:

```
TRK SEC BYTE FROM  TO

 1   7  $C8  $20  $4C
 1   7  $C9  $12  $DC
```



Beating Stoneware's...

DB Master v4.2

By Clay Harrell

DB Master 4.2
DB Master Business Writer
Stoneware, Inc.
5Ø Belvedere St.
San Rafael, CA 949Ø1

Requirements:
At least 64K (required by program)
At least one disk drive
Super IOB v1.5
Some blank disks
DB Master version 4.2 and/or Business Writer from Stoneware

OK, it's about time someone put to paper the deprotection method used for the Stoneware series of software! If you have used (the infamous) DB Master, you will know exactly what I mean...

Stoneware offers several great products, but they are not known for customer service. Stories of four-week delays for backups, and unanswered phone calls to customer service are all true, and probably very familiar to those who have had any problems. Well, this is where I come in... Here is your customer service!

Now Stoneware has developed a new product called "Business Writer". This is a word processor allowing you to take files from several different word processors (and formats) and combine them into one document. It is very easy to use and convenient.

The protections used on DB Master version 4.2 and Business Writer are exactly the same. This should be no great shock since the protection used is good, and for quite some time remained untouched by prying eyes. The main protection used is half tracking, and Stoneware was one of the first (if not THE first) publisher to use half tracking.

The only other protection used by Stoneware is modified data and address epilogue bytes (you know, the three byte sequence found on every sector of a disk that tells DOS where the data to read ends).

So our job is to convert the half tracks to normal DOS whole tracks and to change Stoneware's DOS to read just whole tracks and not half tracks. It turns out this isn't a difficult job as far as manual labor is concerned. But to figure it out, you need a real good understanding of RWTS (the Read and Write a Track and Sector portion of DOS).

First a little background on half tracks... Yes, regardless of how stupid it sounds, it is possible to write in between the normal DOS track, thus the name "half tracks" evolved. This is not to say that we can use track 1, 1.5 and track 2. Due to hardware limitations, tracks must be 1 full step away from each other. In other words, you could use full tracks Ø to 5, and then half tracks 6.5 to 34.5, and this is precisely what Stoneware has done. Notice that track 6 and track 5.5 are not used. Again, this is because tracks must be at least 1 full step away from each other, or you could get "track bleeding" where data from one track bleeds to another. (Actually, there is a way around track bleeding called "spiral tracking", but that is out of the scope of this article).

Now as any good student would ask, "Holy bat-tracks, how did you know that Stoneware uses tracks Ø to 5 and 6.5 to 34.5?" Well, you can not tell by listening, so the only way left is by watching. This involves stripping your drive of its skin and watching the stepper motor (as described in Hardcore COMPUTIST No. 5, pg. 11), or by using a "track monitor" such as Track Star by Midwest Microsystems.

Whatever your weapon, Stoneware knows that only some extremely small percentage of Apple users would **ever** be able to figure this out, much less find a way to convert and defeat this protection. Well, we can't let them get away with it!

Now that we have a diagnosis on the mutation, we have to understand how it works. On a hardware level, the way your Apple tells the drive to move the read/write head is by pulses. Timing is critical, and there are a couple of ways to create pulses to move the head. Each pulse moves the head a quarter track, but the way DOS is written you can only move the head in half track increments from standard RWTS. Stoneware uses a fairly standard RWTS portion of DOS, so they can only deal in half and whole tracks.

The routine in DOS that carries out the job of moving the head is at $B9AØ. It issues two pulses to move the head a full track. But if we tell it to do one more pulse it pushes the head to a half track beyond the desired track. This is easy to do from normal DOS. At $B9AØ you can put a JSR (Jump Subroutine) to the following routine:

```
B43Ø-   86 2B       STX  $2B
B432-   85 2A       STA  $2A
B434-   C9 XX       CMP  #$XX
B436-   9Ø Ø2       BCC  $B43A
B438-   E6 2A       INC  $2A
B43A-   AD 78 Ø4    LDA  $Ø478
B43D-   C9 XX       CMP  #$XX
```

& Business Writer

```
B43F-   90 03       BCC  $B444
B441-   EE 78 04    INC  $0478
B444-   A5 2A       LDA  $2A
B446-   60          RTS
```

This is the routine that Stoneware uses to read half tracks (note that the XX represents twice the value of the track you want to read). When Stoneware wants to read whole tracks they put a $60 (RTS, Return From Subroutine) at $B434. When they want to read half tracks they put a $C9 back at $B434.

It was easy to find the routine that did the switching between whole and half tracks by searching for the byte sequence "A9 C9 8D 34 B4", which represents "load the accumulator with the value $C9 and store it at $B934". To fix it so it just reads whole tracks, merely involves changing the "load accumulator with the value $C9" to "load the accumulator with the value $60". This way the half track routine is never accessed.

Now we must copy tracks Ø to 5 to another disk and convert the half tracks 6.5 to 34.5 to whole tracks by reading in half tracks and writing out whole tracks.

Super IOB can handle it with ease. Examination of the disk reveals that both epilogues have been changed to DF AB. The controller reads the Stoneware disk from tracks Ø to 5 (using the Fast routine in Super IOB v1.5) and writes them onto the copy normally. Simple so far. Each following track is read by setting the variable CD to Ø (seek only) and going to the R/W Sector routine to move the head to track x.Ø, then using the Move S Phases subroutine to move forward one half track to track x.5. This is done for each track.

A sector edit is performed on the following locations:

```
--------------------------------------
  Track   Sector   Byte   From   To
--------------------------------------
   $00     $03     $91    $DF    $DE
    00      03      9B     AB     AA
    00      03      35     DF     DE
    00      03      3F     AB     AA
    00      0C      E3     C9     60
 add this for Business Writer:
    05      0A      13     C9     60
--------------------------------------
```

Instructions

1) Install the controller with this article in Super IOB v1.5. It takes advantage of the Fast routine in version 1.5.

2) If you are copying Business Writer, add these lines:

5010 DATA 6 CHANGES
5070 DATA 5,10,19,96

3) RUN Super IOB and copy the disk.

4) Don't forget to copy *both* sides of DB Master.

Congratulations, you now are finished!

controller

```
1000 REM STONEWARE CONTROLLER
1010 TK = 0 :LT = 6 :ST = 15 :LS = 15 :CD = WR :FAST
     = 1
1020 GOSUB 490 : GOSUB 170 : GOSUB 610 :TK = 5 :
     GOSUB 310 :TK = 0
1030 GOSUB 490 : GOSUB 230 : GOSUB 610 :LT = 35
     :TK = 6 :ST = 0
1040 T1 = TK : GOSUB 490 : RESTORE : GOSUB 170
1050 CD = 0 : GOSUB 100 : POKE BUF , PEEK (BUF )
     - 1 :CD = RD :PH = TK * 2 :S = 1 : GOSUB 130
1060 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1060
1070 IF BF THEN 1090
1080 ST = 0 :TK = TK + 1 : IF TK < LT THEN 1050
1090 GOSUB 230 : GOSUB 490 :TK = T1 :ST = 0
1100 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1100
1110 ST = 0 :TK = TK + 1 : IF BF = 0 AND TK < LT THEN
     1100
1120 IF TK < LT THEN 1040
1130 HOME : PRINT "COPYDONE!" : END
5000 DATA 223 ,171 ,223 ,171
5010 DATA 5^ CHANGES
5020 DATA 0 ,3 ,145 ,222
5030 DATA 0 ,3 ,155 ,170
5040 DATA 0 ,3 ,53 ,222
5050 DATA 0 ,3 ,63 ,170
5060 DATA 0 ,12 ,227 ,96
5070 REM DATA5 ,10 ,19 ,96
```

controller checksums

```
1000  - $356B    1110  - $2028
1010  - $F776    1120  - $8AB5
1020  - $7B8D    1130  - $74C1
1030  - $0AA8    5000  - $0659
1040  - $F4E7    5010  - $69C9
1050  - $840B    5020  - $1734
1060  - $8B0F    5030  - $ABED
1070  - $19F5    5040  - $6CFE
1080  - $5D14    5050  - $C398
1090  - $2981    5060  - $71B6
1100  - $1881    5070  - $43E8
```

Another DB Master 4 +

By Anthony L. Barnett

Stoneware, Inc.
50 Belvedere St.
San Rafael, CA 94901

Requirements:
Apple //e, //c, or 64K][Plus
Super IOB v1.2
A sector editor
Previous DB Master softkeys from Hardcore COMPUTIST Nos. 7 and 21.

When DB Master Version 4+ disappeared from the Most Wanted list in Hardcore COMPUTIST No. 20, I thought a softkey might be on the way. When I saw Clay Harrell's softkey in Hardcore COMPUTIST No. 21, I thought at last I would be able to back up this program with four copies which I usually make of all my valuable software. (Stoneware has done some amazing feats to fool all the bit copiers but not the Super IOB technique on this software.)

Unfortunately, although Super IOB copied the disk without an error message the resultant copy would not boot. In fact, it crashed into the monitor at $B100. As nothing had been put into memory at $B100, it appeared as though the third stage of the boot was failing to load.

I then looked at $BF75 and saw that a volume mismatch error ($20) had occurred. Fortunately DB Master uses a fairly standard RWTS routine. The volume number found in location $B7F6 was $FF. However, checking the code which caused the jump to $B100, I saw that $FF was loaded into $BF76 by the program.

I decided the simplest way to check the actual volume number was to use a nibble viewer (like The Nibbler) on the original disk. Sure enough, a volume number of 1 showed up. So, this time I used Super IOB with a disk formatted with volume number of 1.

At last the familiar DB Master logo appeared. This time, however, the program would not read the Utility disk. It was then I remembered Dan Lui's earlier softkey for version 3 (Hardcore COMPUTIST No. 7). With a sector editor, I modified track 0, sector E, bytes $A,$B to D0 12.

With this last modification I could not boot program disk #1 and read data and utility disks. However, The program would not recognize Program disk #2. Even after modifying track 0, sector E on disk 2 it was still not readable. Finally I decided to check the volume number again. Eureka! The volume number of disk 2 is 2. So by using Super IOB and formatting disk 2 to volume 2 plus the sector edit of track 0 sector E, I had a working, unprotected copy of DB Master 4+... well, almost.

There is a bug in the label printer program which does not allow you to justify a label with spaces in the comment fields. This works fine, however, on the label printer on Utility Pak #2. In addition, one may wish to use Stats Pak or the transaction file merge facility of Utility Pak #2.

To get to the Utility or Stats Pak one uses option 9 from the Utility Pak menu. The unprotected version of DB master could not read the Stats Pak or Utility Pak disks. To overcome this I used Dan Lui's version 3 softkey on these disks, editing track 0, sectors 3 and E only. However, the program would still not read the unprotected Utility or Stats Pak.

I then began to search Program disk No. 2 for some code which turned on the half-tracking of the Utility Paks. In fact, on track 8 sector 9, I found the BASIC program which gives the menus for file maintenance. To get into the program it is necessary to use a sector editor to modify a statement with a CALL-151 (the sequence 8C C9 31 35 31 3A) so the program crashes into the Monitor. Then do a D6:00 to turn off the run time flag, and ©C to Applesoft.

After listing the program, I found a line that said

```
L1 = PEEK(106) * 16^2 + PEEK(105) - 48:
  CALL L1
```

I suspected this call to be the one which turned on the half tracking so I set my "break point" (CALL-151) to the line before it and ran the program again. I thus found the value of L1 and the starting point of the routine I wished to avoid.

I changed this line with a sector editor to read

```
POKE PEEK(106) * 16 ^ 2 + PEEK(105) -
  48, 96:REM
```

At last, I had a completely deprotected version of DB Master 4+ and Utility Paks.

As a final fling, I have always been peeved that DB Master would not find the date from my CCS 7424B clock in spite of what the manual says. Several letters to Stoneware have proved fruitless on this issue. However, by modifying the BASIC program which begins on track 4 sector A of Program disk No. 1 so that it recognizes the signature bytes of my clock I no longer have to enter the date.

Step By Step

1) Use Clay Harrell's softkey (Hardcore COMPUTIST No. 21) with Super IOB v1.2 for DB Master program disks. When asked whether you wish to format the disks, respond Yes and use Volume No. 1 for disk 1 ,and Volume No. 2 for disk 2.

2) In addition to the sector edits listed by Clay Harrell, edit Track 0, Sector E, bytes $A and $B to D0 12 on both program disks.

3) On disk 2 edit track 8, sector 9, bytes $86-$A0 (27 bytes) to

```
B9 E2 28 31 30 36 29 CA
31 36 CC 32 C8 E2 28 31
30 35 29 C9 34 38 2C 39
36 3A B2
```

4) Use Dan Lui's softkey (Hardcore COMPUTIST No. 7) to make unprotected copies of Utility Paks or Stats Pak editing Track 0 only. To be safe, modify line 250 of COPYA to ensure the copy is formatted with a volume number of 1.

5) Put a write-protect tab on all disks.

softkey for...

Barron's Computer SAT

by Glenn Schmottlach

Barron's Computer SAT Study Program
Barron's Educational Series, Inc.
113 Crossways Park Drive
Woodbury, New York 11797
$89.95

Requirements:
48K Apple][or equivalent
Super IOB v1.2
Six blank disks
One or two disk drives

This computer preparation package by Barron has to be considered one of the best programs available for preparation before taking the SAT. Along with three double sided disks comes three hefty study manuals which you use along with the program. Essentially you read the SAT questions out of the study manuals and choose the correct answer on the computer. If you choose the wrong answer the computer responds by giving you hints for the correct answer, and if you are still incorrect the right answer and an explanation of why it is the correct choice appears on the screen. The program uses actual hi-res pictures on the mathematic section to convey an easier understanding of the problems given. Of course the program also keeps track of your score and gives you some guidelines on what you should study more.

The Protection

When the Barron study program disks boot they sound like normal DOS 3.3 disks and the familiar bracket appears. After browsing through the raw track nibbles using Copy II+ 4.4D (another good program for this task is the Nibbler published in Hardcore COMPUTIST NO. 19, page 25), I noticed two things different from normal DOS 3.3 disks. The first thing I noticed was that the start of address marks were changed on all of the disks from the normal values of D5 AA 96 to BB AA 96. For those out there who are unaware of what these marks are let me tell you. When you initialize a disk DOS places these certain values on a disk so it can find the beginning of valuable information such as where a sector begins and where valid data can be found. If standard DOS 3.3 tries to read or write to one of these protected disks it won't be able to find valid data because of the changed marks and therefore an error will be produced. This is a very common technique and it is used on almost all protected disks and often in conjunction with other protection methods. The second major difference I noted was the fact that on all six disk sides track $ØA was unformatted. The only explanation I can come up with this is that the protected DOS might use some sort of nibble count on track $ØA. We won't have to worry about that because the deprotected versions of these disks work smoothly with standard DOS 3.3. The only thing that I had to change because of the unformatted track was programming the controller to skip track $ØA so Super IOB wouldn't get hung up on it.

How to Deprotect the Disk

The following steps are used to deprotect the Barron program disks:

1) Boot a standard DOS 3.3 disk and then clear memory

FP

2) Initialize each of the six blank disks using HELLO as your boot program.

INIT HELLO

repeat for each disk.

3) Load up Super IOB Ver. 1.5 and type in the controller program.

4) Follow the prompts and when the program asks you if you want to initialize the duplicate disk tell it no because we just formatted all six blank disks.

5) Copy all six program sides of the Barron SAT disks using the same contoller just like in step 4.

Closing Comments

You have completely deprotected all of the Barron SAT program disks. You should now feel at ease knowing that you aren't using the original disks. Sit down now and study, study, study. Maybe you'll be able to raise those scores just a few more points; and then again maybe not.

controller

```
1000 REM BARRONS COMPUTER SAT CONTROLLER
1010 TK = 3 :ST = Ø :LT = 35 :CD = W
1020 T1 = TK : POKE 47445 ,187 : GOSUB 490
1030 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1030
1040 IF BF THEN 1070
1050 ST = Ø :TK = TK + 1 : IF TK = 1Ø THEN TK = 11
1060 IF TK < LT THEN 1030
1070 GOSUB 490 :TK = T1 :ST = Ø : GOSUB 230
1080 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1080
1090 ST = Ø :TK = TK + 1 : IF TK = 1Ø THEN TK = 11
1100 IF BF = Ø AND TK < LT THEN 1080
1110 IF TK < LT THEN 1020
1120 HOME : PRINT : PRINT "DONE^ WITH^ COPY" :
     END
```

controller checksums

```
1000  - $356B     1070  - $D588
1010  - $4CFF     1080  - $CF82
1020  - $A7C5     1090  - $24F9
1030  - $B1C4     1100  - $97D8
1040  - $15AA     1110  - $ED2F
1050  - $9CE7     1120  - $C7ØØ
1060  - $482C
```

Softkey for...

By Clay Harrell

Baudville
1001 Medical Park Dr. S.E.
Grand Rapids, MI 49506
$49.95

Requirements
64K Apple][Plus, //e or //c
One disk drive with DOS 3.3
A sector editor
Take 1 from Baudville Software

Take 1 is an excellent graphics program! Just looking at the demo makes you want Christmas to come early, so you can get one. The graphics are extremely smooth, flowing and detailed. This program is a "10" on my list.

The protection isn't too shabby either, and Baudville does a trick or two to keep us from making easy copies. The protection can't be too wild though, or the users would have problems interfacing their graphics routines with normal DOS 3.3.

The disk seems pretty much normal, and can be copied with COPYA without too much difficulty. Use COPYA to copy Take 1 to another disk. This will be our work disk and we can examine the protection from here.

Go ahead and boot the COPYA version of Take 1. Notice it reads the disk and prints "Take 1" and your serial number on the screen. Then it just hangs there, looking for something on the disk!

Since (after many tries of reading the disk and not finding what it wanted) it doesn't clear memory and reboot, we can interrupt the program at our leisure and examine the code. The best way to do this is to use a Replay II card or a Know-Drive. These tools will allow you to interrupt the program and let you know the current location a program is running at.

Running in the RAM Card

But we encounter our first problem with Take 1: they are using the memory within the RAM card for the RWTS. To access this memory we have to use some tack, and refer to our 16K RAM card manual.

Looking at our RAM card manual tells us that if we want to turn on the RAM card, we type C080 from the monitor. If you try this, your Apple will lock up and the only way to recover is to power down and cold boot. The reason for this is simple.

The RAM card occupies the same memory as your Applesoft ROMs and the monitor ROM ($D000-$FFFF). It is accessed by soft switches, much like the way you turn on and off the hi-res pages. But when you turn on the RAM card, you are also turning off the motherboard monitor ROM, which overlooks all your Apple's functions. Since there isn't a monitor ROM in the RAM card, your Apple locks up. To prevent this, we can set the RAM card to "write enabled," and read from the motherboard ROMs. Now we can copy an image of the monitor ROM into the RAM card using the monitor's memory-move command. Finally, we can turn on the RAM card and examine the memory in it!

To do this type:

```
C081 C081
F800<F800.FFFFM
C080
```

From our Replay card we know that the program was stuck reading the disk around $DC4F. Here is the code:

```
DC43- RTS            :End of previous routine.
DC44- LDY #$FC       :start
DC46- STY $26        :
DC48- INY            :
DC49- BNE $DC4F      :
DC4B- INC $26        :
DC4D- BEQ $DC42      :
DC4F- LDA $C08C,X    :
DC52- BPL $DC4F      :
DC54- CMP #$D5       :First byte of Address
DC56- BNE $DC48      :prologue marker
DC58- NOP            :
DC59- LDA $C08C,X    :
DC5C- BPL $DC59      :
DC5E- CMP #$AA       :Second byte of Address
DC60- BNE $DC54      :prologue marker
DC62- LDY #$03       :
DC64- LDA $C08C,X    :
DC67- BPL $DC64      :
DC69- CMP #$96       :Last byte of Address
DC6B- BNE $DC54      :prologue marker
          .
          .
DC8B- LDA $C08C,X    :
DC8E- BPL $DC8B      :
DC90- CMP #$DE       :First byte of Address or
DC92- BNE $DC42      :Data epilogue
DC94- NOP            :
DC95- LDA $C08C,X    :
DC98- BPL $DC95      :
DC9A- CMP #$AA       :Last byte of Address or
DC9C- BNE $DC42      :Data epilogue
DC9E- CLC            :
DC9F- RTS            :End of routine
```

As you can see, this routine checks the disk for the prologue address marker of $D5 AA 96. Note that a prologue address marker is on every sector of a normal DOS disk. It's a road sign to tell DOS the next few bytes designate what track, sector, and volume it is reading (hence "soft sectoring").

Likewise, the routine also checks for an epilogue byte of $DE AA. Epilogue bytes are on every sector of a normal DOS disk too. They tell DOS that "the data ends here'.

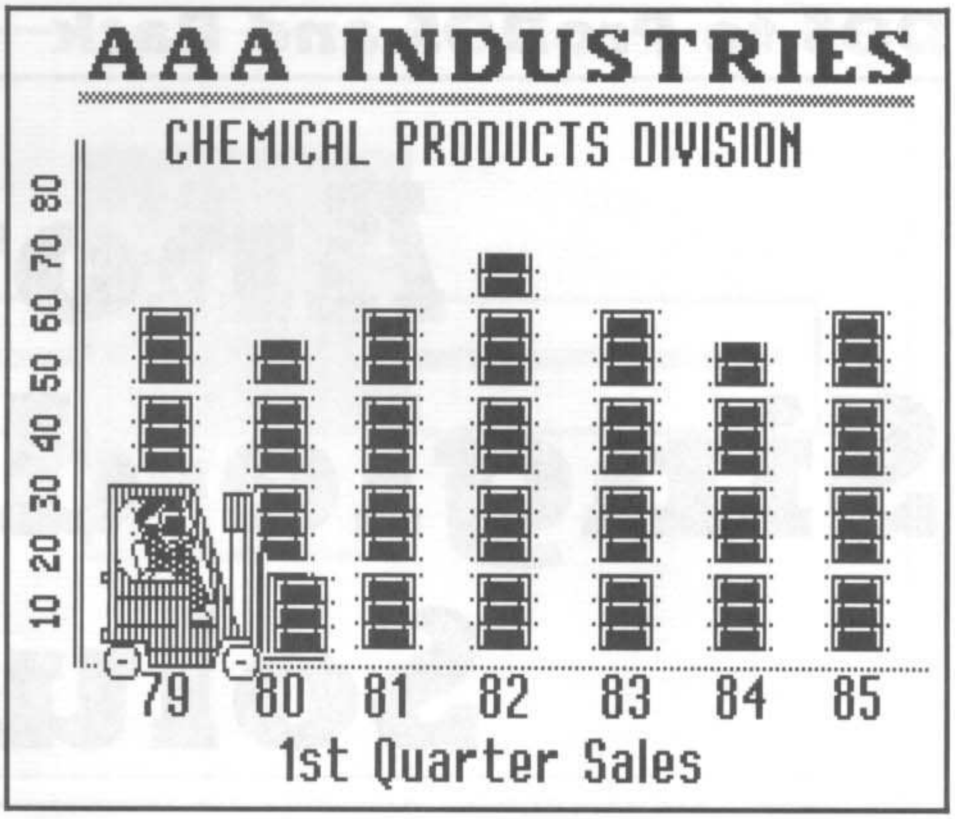
There is really nothing funny about this routine. The address and epilogue bytes this routine is looking for is on every normal DOS disk. So we must dig deeper, since the program is hanging here for some reason. First, we should find where this routine is called from (we know the routine starts at $DC44 since there is an RTS at $DC43).

To make this easier, I copied the RAM card memory down to motherboard RAM with the monitor move command

4000<D000.FFFFM

Then I used the Inspector's search command to look for a JSR $DC44. Sure enough, I found one at $500F (really $E00F). Here is the code I found:

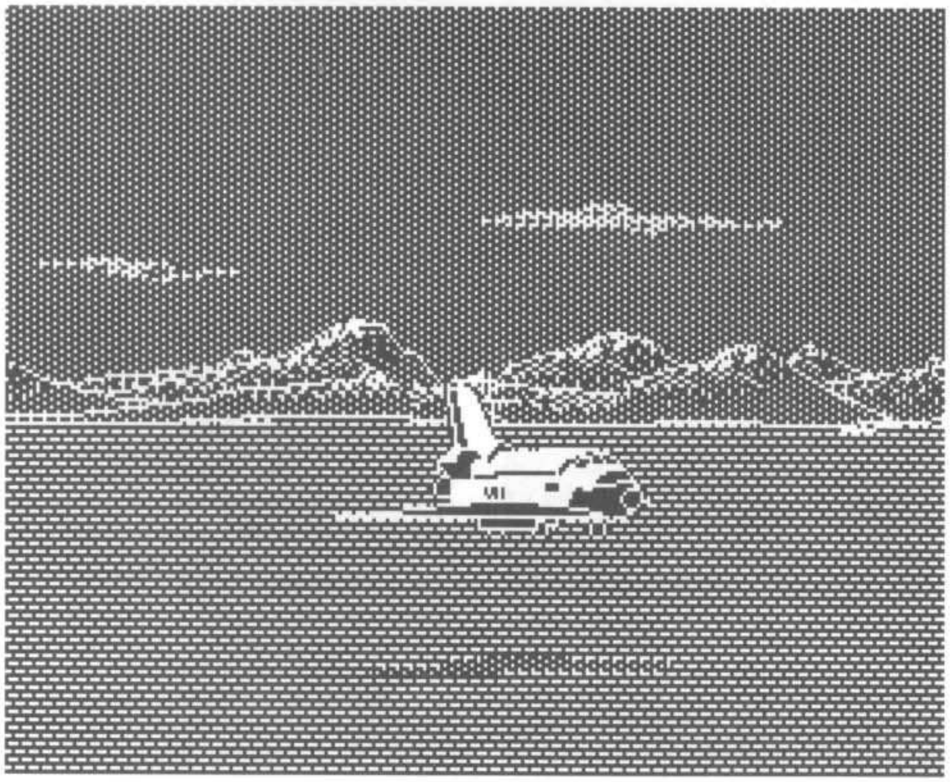
```
E00F- JSR $DC44 :go to check disk.
E012- LDA $2D   :load A with location $2D.
E014- CMP #$FF  :compare it to $FF.
E016- BNE $E00F :If not equal, go to $E00F
```



Here is the offending routine! Notice it jumps to a subroutine at $DC44 to check the disk. After it returns, it checks location $2D, and if it is not $FF, it goes back and starts over. Hence our disk keeps running in this infinite loop and never finishes loading the program. Obviously, somewhere in the code at $DC44 it reads the disk, and if something is not up to benchmarks, it loads location $2D with something other than $FF.

So to defeat this routine we must find it on the disk, and change the code with a sector editor from $D0 F7 (BNE $E007) to $EA EA, which stands for "no operation" (NOP). This removes the "branch back to start if location $2D is not equal to $FF". The code lives on track 0, sector 6, byte $16, and also on track 0, sector $F, byte $08. I used the Inspector's Locate command to find the code on the disk.

Lo and behold, Take 1 is now deprotected. Note that this routine lived in two places on the disk. If you only found the first one on sector 6 of track 0, the program would die about a third of the way through the boot.



Cookbook Version

1) Boot your system master disk and run COPYA.

PR#6
RUN COPYA

2) Run your favorite sector editor and make the following changes to the COPYA version of Take 1:

```
Track 0, Sector 6, Byte $16
   Change from $D0 F7
            to  $EA EA

Track 0, Sector $F, Byte $08
   Change from $D0 F7
            to  $EA EA
```

3) Write the sectors back out.
And you're all done!

Another Single-drive Solution

By Brian K. Chin

Requirements:
64K Apple][Plus and up
One disk drive
A DOS 3.3 disk
A blank disk
A ProDOS disk formatter program
A sector editor or block editor (optional)

This procedure for single disk DOS-to-ProDOS conversion is presented not only as an alternative to the procedure presented by Mr. Eubanks (Hardcore COMPUTIST No. 9, p.15), but also as a simpler and more flexible solution which will attempt to shed a little light on the subject of how disk formats work. The process involves the creation of a disk on which both the ProDOS and DOS 3.3 environments can peacefully co-exist. The inspiration for this article was to find an easier way to convert files with only one drive. Being an avid AppleWorks user, I have found that even occasionally working with only one drive can be a real hassle, not to mention being unable to convert text files back and forth without the second drive. The reason behind the creation of the Dual DOS Disk was to see if it could even be done. The notion that my new disk could be used to solve the single disk conversion problem did not come to my attention until after it had spent a month on my shelf doing nothing.

As with all changes come both advantages and disadvantages. The same holds true here as well. You are no longer confined to converting only Applesoft and binary files. You are no longer confined to a maximum file size of 59 sectors (30 blocks) in one direction, and 193 sectors (86 blocks) in the other. You now possess the ability to transfer a file of ANY type, and up to a maximum size of 258 sectors (129 blocks). The only drawback which exists is that you only have the available space of half of a ProDOS disk and half of a DOS 3.3 disk, which reduces your operating and storage space by a factor of two. While this may eliminate any need for disk swapping during the conversion process, it may require copying files onto and off of the special disk both before and after the conversion is made. In light of the benefits gained, this becomes a minor problem at worst. Lastly, and most important, it permits you to use the CONVERT program, without forcing you to make all of those nasty little RESET breaks, memory moves or forcing you to learn how to think in hexadecimal.

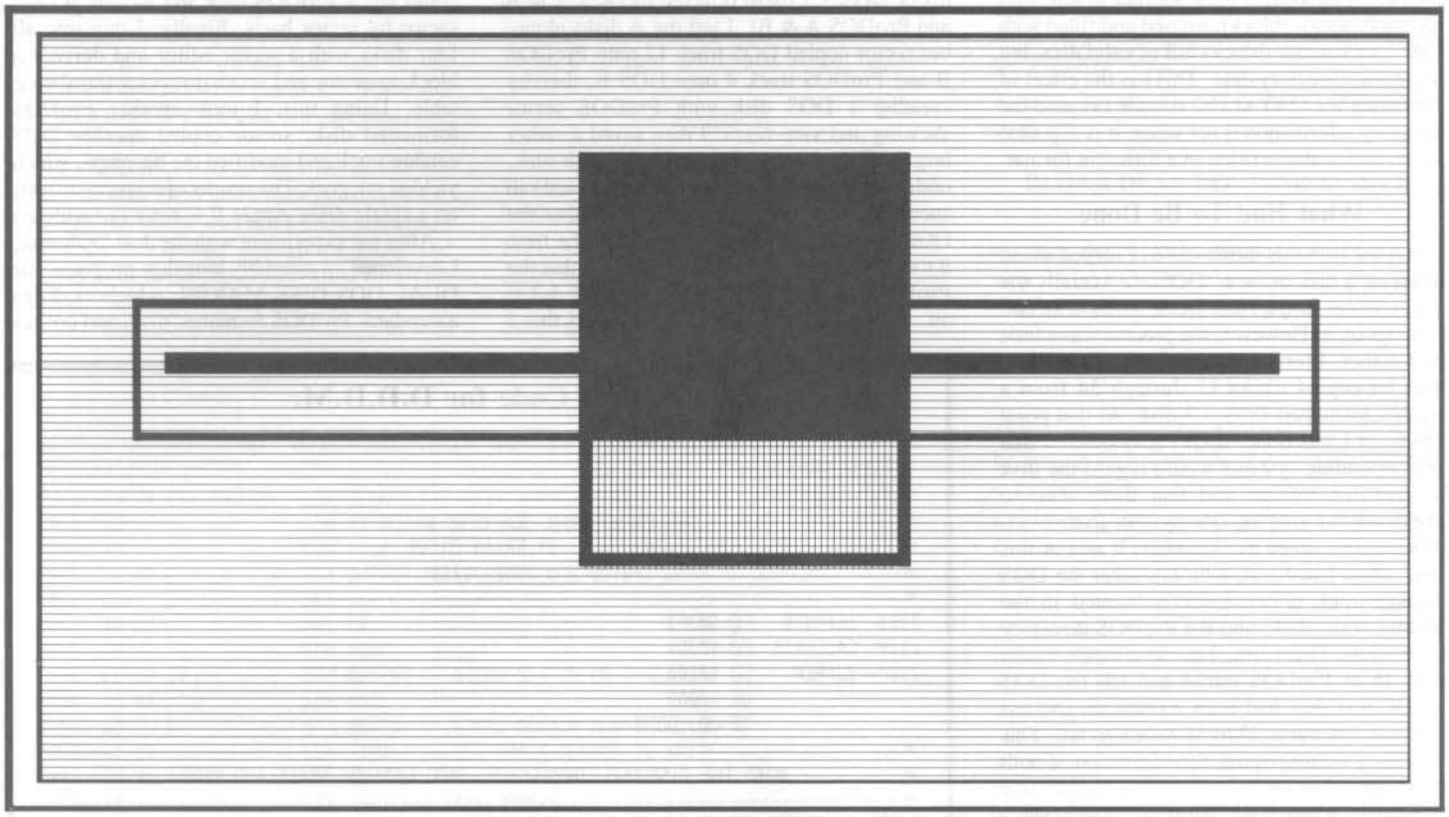
But Aren't the Formats Different?

At one time or another, it was the general concensus (I think) that only a disk formatted under DOS 3.3 could contain DOS 3.3 files. The same held true with Apple CP/M, Pascal and ProDOS. This is true to a certain degree. The major dissimilarities between the four operating systems (we'll only be dealing with two of them) are the location of the directory data and volume bit-map, the placement and numbering of logical sectors, and how the operating system defines the minimum amount of data that can be read at any one time.

Let's start with the simplest of them, the last one. As you already know, DOS 3.3 reads and writes data in 256 byte sectors, while ProDOS uses 512 byte blocks. However, as we shall see in a minute, ProDOS does not really allocate eight blocks per track when it formats, in the way DOS 3.3 would allocate 16 sectors, but *defines* a block as a specified sector, plus the physical sector located two sectors over. Thus, ProDOS does not read a full block, it makes a sequential read of two noncontiguous sectors. If this is confusing, please bear with me. It should become a little clearer very soon.

In order for a disk (with the exception of DOS 3.2) to be Apple-readable, its format must be consistent with Apple's sixteen-sector format. This is why sector editor utilities, such as DiskEdit, are able to read any of the four operating system formats. However, the sectors read from a non-DOS 3.3 disk will appear to be in an unorganized order because each system utilizes a different *logical* sector layout when accessing the disk.

When a disk is formatted, under any of the four operating systems, it divides each track into 16 equally sized subdivisions. Each of these subdivisions are known as *physical* sectors because they are numbered (physically) by their



orientation on the disk. Sector Ø is next to sector 1, which is next to sector 2, and so on. Sector Ø is determined rather arbitrarily and is used to denote the start of that particular track. Within the boundaries of each physical sector are two fields, the first containing the address information, and the second the actual data.

Within the address field of each physical sector is a pair of bytes indicating the physical sector number. With the exception of sector Ø, the logical sectors are numbered by their respective operating systems, independently of their location on the disk (their physical sector numbering).

All the operating systems have their own way of keeping track of the sectors internally. These are referred to as *logical* sectors. When the time comes to access the disk, the logical sector number is cross-referenced in a table to find the matching physical sector number. The way in which a physical sector is logically numbered is called sector skewing. The following table shows how the different operating systems logically number each of the 16 sectors.

```
Physical   Ø 1 2 3 4 5 6 7 8 9 A B C D E F
CP/M       Ø B 6 1 C 7 2 D 8 3 E 9 4 F A 5
Pascal     Ø 8 1 9 2 A 3 B 4 C 5 D 6 E 7 F
DOS 3.3    Ø 7 E 6 D 5 C 4 B 3 A 2 9 1 8 F
ProDOS     Ø 8 1 9 2 A 3 B 4 C 5 D 6 E 7 F
P.DOS blk  Ø 4 Ø 4 1 5 1 5 2 6 2 6 3 7 3 7
           ↑   ↑This sector is the second
           ↑    half of ProDOS block Ø.
           ↑This sector is the first half
            of ProDOS block Ø.
```

Thus, when DOS 3.3 looks for a sector $ØF (its logical number), it first reads the 16th byte (don't forget sector Ø) in its sector translate table, which yields the corresponding physical sector number (in this case 2). DOS then looks for physical sector 2, and reads the following data field. When the sector is in memory, DOS will think that it got sector $ØF and continue on its way. You will note that disk access under ProDOS (and Pascal) is dramatically faster than DOS 3.3. This is due less to changes in the operating system and more to the changes in sector skewing.

A Word About Directories

The main reason why any one operating system cannot read another operating system's files is the location and format of the volume bit-map and directory sectors. The directory sectors (blocks) act as the the disk's table of contents, holding the file names and their locations on the disk. The volume bit-map is responsible for keeping the status of each sector (block) on the disk; whether the sector (block) is available for storing data, or is currently in use. On a ProDOS disk the directory lies in blocks 2 through 5, and the bit-map in block 6. Under DOS, the directory lies on track 17, sectors 1 through 15 ($ØF), and the bit-map, or volume table of contents (VTOC) on sector Ø.

The directories of both DOS and ProDOS function in much the same way. They both contain filenames and vital file accounting information such as links to the other directory sectors (blocks), file-size, file-type, and file location, or more appropriately, the location to the DOS track/sector list and the ProDOS index block. These two packets of data contain the locations of all track/sector pairs (blocks) of data relevant to that particular file. ProDOS directory blocks also contain other data. Among them are, volume name, file creation times and dates, last modification time and date, starting file address, file length, and text record length.

The differences between bit-map formats are more evident. The DOS bit-map begins at byte 56 on track 17, sector Ø, the VTOC. The first quarter of the sector is reserved for miscellaneous accounting information such as disk volume number, DOS version number, location of first directory sector, and track allocation direction. Each track is represented by a sequence of 4 bytes. Of the first two bytes, each bit, either 1 or Ø, represents a sector which is either free or allocated, respectively. The second two bytes are not needed and left at $ØØ. Thus, an empty track is represented by 'FF FF ØØ ØØ'. Under ProDOS, since a volume may be of any size, allocates one byte to represent 8 blocks, or in this case, a single track. Here too, it bit-maps with 1 and Ø so that $FF indicates free range of 8 blocks.

Bearing this in mind, the only reason that DOS cannot read ProDOS files, and vice-versa, is that both DOS and ProDOS expect the directory to be in a certain position on the disk. Thus, when the directory sector (block) is read, the operating system must attempt to interpret an empty sector (block), unused and filled with "00", or a sector (block) full of valid data, but useless as directory data. This has the effect of confusing the CATALOG routine because the directory information is not where it is expected to be, and is also written in a different format.

What Had To Be Done

Armed with this information, I forged ahead to create a disk of two "DOSs". Actually the solution was simple, just create a disk with two catalogs on it. However, the process was a little bit tougher. I formatted a disk under ProDOS, then bit-copied tracks 17 through 34 from a freshly formatted DOS 3.3 disk. At that point I thought I was done, until I remembered that both operating systems would regard the disk as free for storage and that there was no provision for keeping one system from over-writing data saved by the other. It was at this time when I realized, a bit late, that the DOS catalog track is conveniently located in the middle of the disk, and the ProDOS directory on track 0. Therefore, I could allocate tracks 0 to 16 as ProDOS tracks and tell the DOS VTOC that they had been written to, and tell ProDOS that tracks 17 to 34 were also full. This however, would require a modification of both bit-maps.

By manipulating the information in a disk's bit-map, one can define sections of the disk as being are off-limits to the operating system. The operating system interprets those areas as "written to" and therefore cannot de-allocate them, whether there is valid data there or not (in this case, it's the latter). The only way that DOS or ProDOS can 'free up' a used sector is by deleting the file which last resided there. Since there will be no DOS files in the ProDOS allocated section and vice versa, DOS will not be able to do anything to the ProDOS allocated tracks, as well as the reverse. Be warned however, that improperly modifying a bit map after data is stored on the disk could result in allocation of already used sectors as usable, which could lead to the accidental over-writing of valuable data or even critical DOS information.

At the time I made the proper modifications, I used a sector editor for the DOS bit-map and a block editor for the ProDOS bit-map. By accident however, I discovered that the required procedure need not be as long and complicated, and that I could complete the entire operation in the DOS 3.3 environment. As I was scanning through track 0, my finger brushed the key which instructs the program to read the previous block of data. Since I was on block 0, it read block 279 (track 34, sectors 1 and 15). I was rather surprised to see that it did not return with an 'I/O ERROR' message. At that point, I tried reading and writing to a regular DOS 3.3 disk with the block editor and to a ProDOS disk with a sector editor, with mostly successful results. The only problem was that the logical sector numbering (skewing) was fouled up.

Sensing that the discovery was only beginning, I tried another experiment. I formatted two disks under ProDOS and two under DOS 3.3 (here referred to DOS A & B and ProDOS A & B). I left the A disks alone, but sector copied DOS track 17 onto ProDOS B and ProDOS track 0 onto DOS B, thereby creating a DOS disk with ProDOS sector skewing and vice-versa. I then saved a rather large ($7000 bytes) binary file on each disk, under the appropriate operating system and ran speed tests on loading them. I found that the DOS skewing dramatically increased the time it took for ProDOS to load the file, but that the ProDOS skewing had no effect on DOS 3.3 at all. Upon determining this, I surmised that I could use a ProDOS disk and modify it on a sector by sector basis. Finally, I dug into all four disks with a sector editor and derived a block-to-sector and sector-to-sector translation table. Using this, I took another ProDOS formatted disk, sector copied another DOS catalog track and modified the bit-maps, which yielded my goal. The results of conversion tests on a single drive Apple][+ were miraculous.

After my experiment with the dual DOS disk, I developed an assembly language program, the DUAL DOS DISK MAKER, which modifies a standard, ProDOS formatted disk, and creates

Source Code for D.D.D.M.

```
* ------------------------------------------------------------------------------- *
*                             DUAL DOS DISK MAKER                                 *
*                              BY BRIAN CHINN                                     *
*                                JUNE, 1985                                       *
* ------------------------------------------------------------------------------- *
03E3- SETRWTS  .EQ $03E3
03D9- CALLRWTS .EQ $03D9
2100- BUFFER   .EQ $2100
               .OR $2000
               .TF OBJ.DDDM
* ------------------------------------------------------------------------------- *
*         MAKE THE DISK DOS COMPATIBLE - ADD CATALOG TRACK AND VTOC               *
* ------------------------------------------------------------------------------- *
2000: 20 76 20  START    JSR CLEARBUF
2003: A2 80     DOSMAP   LDX #$80     ;SET T/S BITMAP BUFFER OFFSET
2005: A9 FF              LDA #$FF     ;SET T/S BITMAP MARKING VALUE
2007: 9D 00 21  L1       STA BUFFER,X ;ALLOCATE A TRACK AS FREE
200A: 9D 01 21           STA BUFFER+1,X
200D: E8                 INX          ;SKIP THE NEXT FOUR BYTES
200E: E8                 INX
200F: E8                 INX
2010: E8                 INX
2011: D0 F4              BNE L1
2013: A2 00     MAKEVTOC LDX #$00     ;BUILD REMAINING VTOC AREA
2015: BD A9 20  L2       LDA DATATBL1,X ;FETCH DATA FROM FIRST TABLE
2018: F0 0A              BEQ WRTVTOC  ;TEST FOR 'EOD'
201A: BC AA 20           LDY DATATBL1+1,X ;FETCH DATA OFFSET
201D: 99 00 21           STA BUFFER,Y ;STORE DATA IN BUFFER
2020: E8                 INX
2021: E8                 INX
2022: D0 F1              BNE L2
2024: A9 00     WRTVTOC  LDA #$00
2026: 8D ED B7           STA $B7ED    ;STORE SECTOR NUMBER TO WRITE
2029: A9 11              LDA #$11     ;SET TRACK NUMBER TO WRITE
202B: 20 82 20           JSR GORWTS   ;WRITE VTOC
202E: 20 76 20           JSR CLEARBUF
2031: A9 01              LDA #$01     ;WRITE 1ST DIRECTORY SECTOR (ALL 00'S)
2033: 8D ED B7           STA $B7ED    ;STORE SECTOR NUMBER TO WRITE
2036: A9 11              LDA #$11     ;SET TRACK NUMBER TO WRITE
2038: 20 82 20           JSR GORWTS
203B: A9 11     BUILDDIR LDA #$11
203D: 8D 01 21           STA BUFFER+1 ;STORE TRACK OF NEXT DIRECTORY SECTOR
2040: A2 0F              LDX #$0F
2042: 86 FF              STX $FF      ;SAVE SECTOR# FROM RWTS DECTRUCTION
2044: 8E ED B7  L3       STX $B7ED    ;STORE SECTOR NUMBER TO WRITE
2047: CA                 DEX          ;POINT  TO PREVIOUS SECTOR
2048: 8E 02 21           STX BUFFER+2 ;STORE SECTOR POINTER IN BUFFER
204B: 86 FF              STX $FF      ;SAVE SECTOR VALUE FOR NEXT WRITE
204D: A9 11              LDA #$11     ;SET TRACK NUMBER TO WRITE
204F: 20 82 20           JSR GORWTS   ;WRITE SUBSEQUENT DIRECTORY SECTOR
2052: A6 FF              LDX $FF      ;FETCH VALUE OF NEXT SECTOR TO DO
2054: E0 01              CPX #$01     ;CHECK TO SEE IF WE'RE DONE
2056: D0 EC              BNE L3
```

a DUAL DOS data disk. This was to alleviate the problem of having to copy the entire disk for my friends who wanted one, but mostly because I wanted to write this article. Have you ever tried to COPYA a disk out of a magazine article?

Unfortunately, I have only scratched the surface of how DOS and ProDOS disks work. Although the intent was not a DOS tutorial, I hope I have exposed enough of DOS and ProDOS to form the foundation upon which the "DUAL DOS DISK MAKER" is based. For the most concise and in-depth coverage of both DOS and ProDOS, I highly recommend both *Beneath Apple DOS* and *Beneath Apple ProDOS* by Don Worth and Pieter Lechner.

Doing It

1) Type in the DUAL DOS DISK MAKER hexdump and save it on a DOS 3.3 disk.

2) Format a blank disk, or as many as you need, under the ProDOS operating system. You can use the FILER program on the ProDOS Users Disk, the Appleworks disk formatter, or any other suitable utility. You may name the disk anything you want, as long as it complies with the ProDOS rules.

3) Boot up any DOS 3.3 disk and load the program.

BLOAD DUAL DOS DISK MAKER

4) Place the ProDOS disk(s), formatted from step 2, into the drive and run the program. Repeat as often as necessary.

CALL 8192

5) You now have a DUAL DOS disk which can be copied with COPYA or any other suitable copy program.

6) When you wish to use the DOS User's Conversion Kit (DUCK), simply set the DOS 3.3 drive to the slot and drive your controller card is in, and set the prefix via the pathname or slot and drive option. You're all set to convert. This disk will also work on a system with more than one drive, but I see no real need to use it there.

One last note: since this disk only leaves half as much storage space as usual, I would recommend it only be used as a transfer disk, unless your particular application does not require large amounts of disk space. If you're using an applications program or programming, save the data you need to convert to both your master storage disk and to the DUAL DOS disk. If this is not possible, copy the appropriate files onto the DUAL DOS disk before you convert.

```
* ------------------------------------------------------------------------- *
*              TELL PRODOS TO ALLOCATE HALF THE DISK FOR DOS                *
* ------------------------------------------------------------------------- *
2058: 20 76 20  PRODOS   JSR CLEARBUF
205B: A9 FF              LDA #$FF      ;ALLOCATE A TRACK AS FREE
205D: A2 10              LDX #$10
205F: 9D 00 21  L4       STA BUFFER,X  ;STORE PRODOS BITMAP DATA IN BUFFER
2062: CA                 DEX
2063: 10 FA              BPL L4
2065: A9 01              LDA #$01
2067: 8D 00 21           STA BUFFER    ;STORE STATUS OF TRACK ONE
206A: A9 03              LDA #$03
206C: 8D ED B7           STA $B7ED     ;STORE SECTOR NUMBER TO WRITE
206F: A9 00              LDA #$00      ;SET TRACK NUMBER TO WRITE
2071: 20 82 20           JSR GORWTS    ;WRITE PRODOS BITMAP
2074: 60                 RTS           ;ALL DONE
2075: 60                 RTS           ;JUST IN CASE STACK WAS PLAYED WITH
* ------------------------------------------------------------------------- *
*                             MAIN SUBROUTINES                              *
* ------------------------------------------------------------------------- *
2076: A9 00     CLEARBUF LDA #$00
2078: 85 42              STA $42
207A: A9 21              LDA #$21
207C: 85 43              STA $43
207E: 20 D6 B7           JSR $B7D6     ;CLEAR BUFFER AT $2100
2081: 60                 RTS
2082: 8D EC B7  GORWTS   STA $B7EC     ;STORE TRACK NUMBER TO WRITE
2085: A9 01              LDA #$01
2087: 8D E8 B7           STA $B7E8     ;STORE RWTS TABLE TYPE
208A: A9 02              LDA #$02
208C: 8D F4 B7           STA $B7F4     ;STORE DISK FUNCTION
208F: A9 00              LDA #$00
2091: 8D F0 B7           STA $B7F0     ;STORE BUFFER LOCATION
2094: A9 21              LDA #$21
2096: 8D F1 B7           STA $B7F1     ;STORE BUFFER LOCATION
2099: A9 00              LDA #$00      ;PREPARE EXTRA DOS LOCATIONS
209B: 8D EB B7           STA $B7EB     ;STORE VOLUME MATCHING NUMBER1
209E: 8D F3 B7           STA $B7F3     ;STORE BYTE COUNT
20A1: 20 E3 03           JSR SETRWTS   ;PREPARE FOR RWTS CALL
20A4: 4C D9 03           JMP CALLRWTS  ;MAKE RWTS CALL
20A7: 60                 RTS
20A8: 60                 RTS           ;JUST IN CASE STACK WAS PLAYED WITH
* ------------------------------------------------------------------------- *
*                               DATA TABLES                                 *
* ------------------------------------------------------------------------- *
20A9: 01 31 01
20AC: 37 03 03
20AF: 0F 02     DATATBL1 .HS 0131013703030F02
20B1: 10 35 11
20B4: 01 11 30
20B7: 23 34              .HS 1035110111302334
20B9: 7A 27 FE
20BC: 06 00 00           .HS 7A27FE060000
```

Dual DOS Maker Hexdump

```
2000: 20 76 20 A2 80 A9 FF 9D    $FBF9
2008: 00 21 9D 01 21 E8 E8 E8    $F767
2010: E8 D0 F4 A2 00 BD A9 20    $F893
2018: F0 0A BC AA 20 99 00 21    $A0B5
2020: E8 E8 D0 F1 A9 00 8D ED    $E40F
2028: B7 A9 11 20 82 20 20 76    $A7CE
2030: 20 A9 01 8D ED B7 A9 11    $211A
2038: 20 82 20 A9 11 8D 01 21    $7ACF
2040: A2 0F 86 FF 8E ED B7 CA    $D331
2048: 8E 02 21 86 FF A9 11 20    $D480

2050: 82 20 A6 FF E0 01 D0 EC    $BC89
2058: 20 76 20 A9 FF A2 10 9D    $07C8
2060: 00 21 CA 10 FA A9 01 8D    $F091
2068: 00 21 A9 03 8D ED B7 A9    $3FA9
2070: 00 20 82 20 60 60 A9 00    $C8C2
2078: 85 42 A9 21 85 43 20 D6    $40DE
2080: B7 60 8D EC B7 A9 01 8D    $E654
2088: E8 B7 A9 02 8D F4 B7 A9    $8D71
2090: 00 8D F0 B7 A9 21 8D F1    $9358
2098: B7 A9 00 8D EB B7 8D F3    $4BE5

20A0: B7 20 E3 03 4C D9 03 60    $F8A3
20A8: 60 01 31 01 37 03 03 0F    $FC82
20B0: 02 10 35 11 01 11 30 23    $66E5
20B8: 34 7A 27 FE 06 00 00       $E09C
```

Adding "If Then

By John R. Vokey

Requirements:
Apple][Plus, //e, //c
A need for higher programming

The decision structure is one of the four basic structures of computer programs (Grogono, 1980). Most high-level computer languages implement this structure as an "IF" command, as in the BASIC statement "IF X>1 THEN X=1" in which the statement following the "THEN" is executed only if the expression following the "IF" evaluates as TRUE. Many computer languages, including some versions of BASIC, extend the "IF... THEN..." command to include an "ELSE" statement with which the programmer may state explicitly what the program is to do if the expression following the "IF" evaluates as FALSE. A few languages extend the concept even further by including the "ELSE IF" statement to allow the explicit conditional sequencing of decisions. Although the functions performed by the "ELSE" and the "ELSE IF" extensions to the "IF" statement may be simulated by appropriate use of the simple "IF... THEN..." command structure and branching commands, "ELSE" and "ELSE IF" add considerable clarity to the structure and flow of the program. For this reason, they have been included in virtually all structured programming languages, such as PASCAL.

This article describes a short machine-language "patch" to Apple's floating-point BASIC, Applesoft BASIC, that extends the BASIC "IF... THEN..." command to allow both "ELSE" and "ELSE IF". The patch will work for the complete series of Apple][computers under the DOS 3.3 environment.

Command Structure and Syntax

The syntax of the Applesoft BASIC "IF... THEN..." command as described in the Applesoft BASIC Reference Manual (1978/1981) is:

IF expr THEN {:} [{:instruction}]

where "expr" refers to any legal Applesoft BASIC expression, "instruction" refers to any legal Applesoft BASIC statement, and "[{: instruction}]" refers to an optional extension of the program line that will be executed only if "expr" evaluates as TRUE (i.e., any non-zero value). Because conditional branching is a common occurrence, Applesoft BASIC allows conditional branch statements to be abbreviated:

IF expr THEN [GOTO] linenum

and

IF expr [THEN] GOTO linenum

where the bracketed command is implied. For each of these statements, if "expr" evaluates as FALSE (i.e., equal to zero), program execution branches to the next numbered line of the program, ignoring in the process any instructions following the "THEN". The machine-language patch is transparent to these statements; with the patch installed they still function as described.

ELSE and ELSE IF

With the patch installed, all versions of the "IF... THEN..." statement may be extended on the same program line with the statement ":ELSE:". Note that the colon is required both as a prefix and a suffix of the "ELSE" statement; otherwise, "ELSE" is assumed to be a variable name and will be ignored by the patch. The basic syntax is:

IF expr THEN {:} [{:instruction} {:ELSE:} {instruction}]

The instruction(s) following the "ELSE" will be executed only if "expr" evaluates as FALSE, whereas the instruction(s) preceding the "ELSE" but subsequent to the "THEN" will, as usual, be executed only if "expr" evaluates as TRUE.

Conditional sequencing of decisions (i.e., "ELSE IF") is effected by following the "ELSE" statement, either immediately or subsequent to other commands, with an "IF" statement, as in:

IF X THEN 1000:ELSE:IF Y THEN 2000

Multiple "ELSE" statements on the same program line are allowed. Each "ELSE" statement is associated with the most recent "IF" statement on the same line that is not already associated with a preceding "ELSE" statement. That is, "IF" and "ELSE" statements are associated in nested-pairs in a

Else" To Applesoft

manner similar to nested "FOR... NEXT" loops. More "ELSE" statements than preceding "IF" statements on the same program line, if executed, result in a "SYNTAX ERROR" in a manner similar to a "NEXT WITHOUT FOR" error.

GOTO and GOSUB

The Applesoft BASIC statements "GOTO" and "GOSUB" (and their ON... GOTO / GOSUB variants) complicate the meaning of the same line rule for "ELSE" statements cited earlier since both of these commands result in a branch from one line of BASIC code to another. For the purposes of the "ELSE" patch, the program line branched to is considered to be an extension of the program line branched from all subsequent lines, however, even with a subroutine, are considered to be new lines.

This complication is of no consequence for "GOTO" commands, but it can limit the effectiveness of "ELSE" statements in progrэm lines containing conditional "GOSUB" commands (e.g., IF X THEN GOSUB 1ØØ:ELSE: ...). The problem arises if the subroutine terminates with a "RETURN" statement on a program line subsequent to the one branched to by the conditional "GOSUB" statement, or if the "RETURN" statement is the last command on the program line even though the line is the one branched to. In both cases, the "ELSE" patch will detect an end-of-line (EOL) token, effectively forgetting the prior "IF" statement that initiated the branch. Upon encountering any "ELSE" associated with the now-forgotten "IF", the program will halt with a "SYNTAX ERROR". The cure for the problem is a simple rule: **Never precede an "ELSE" statement with a conditional "GOSUB" statement.** This rule is less severe than it sounds since the rule is satisfied if the "GOSUB" statement follows the "ELSE" statement rather than the "THEN" statement of an "IF - ELSE" pair. For example, the program line:

```
1Ø IF X THEN GOSUB 1ØØ:ELSE:
PRINT X
```

will crash with a "SYNTAX ERROR" on the "ELSE" statement upon RETURNing from a multiple-line subroutine at line 1ØØ. However, the logically equivalent program line:

```
1Ø IF NOT X THEN PRINT X:ELSE:
GOSUB 1ØØ
```

will execute correctly.

Installing the Patch

Start by using the instructions on page 2 of this magazine key in the hexdump and save it with:

```
BSAVE ELSE.OBJ,A$3ØØ,L$9C
```

It is assembled to reside in page three of memory, but may be re-assembled with a different origin to be located elsewhere (say, above HIMEM). BRUNning the object code from disk (or BLOADing it, then CALLing 768 from BASIC) will install the patch on the system. The patch will remain active until Applesoft BASIC is cold-started (i.e., either a re-boot of the system or the DOS 3.3 "FP" command). The patch is interfaced to Applesoft BASIC through the Applesoft CHRGET routine. Discussion of this approach to patching Applesoft BASIC may be found in Kaner and Vokey (1982) and Mossberg (1982).

It should be noted that CHRGET patches will not function with the ProDOS BASIC operating system.

Examples of the ELSE Statement

The simplest and probably the most common use of the "ELSE" command is exemplified by the following program line:

```
1ØØ IF X THEN PRINT "TRUE" :
ELSE : PRINT "FALSE'
```

Execution of this line will print "TRUE" if X <> Ø, and "FALSE" if X = Ø. Program lines of this sort may be used to replace the more cumbersome line combination normally required in Applesoft BASIC:

```
1ØØ IF X THEN PRINT "TRUE'
11Ø IF NOT X THEN PRINT
"FALSE'
```

An example of the conditional sequencing of decisions (i.e., ELSE IF) is given by:

```
1ØØ IF X THEN PRINT "TRUE" :
ELSE : IF Y THEN PRINT "TRUE"
: ELSE : PRINT "FALSE'
```

In this case, execution of the line will print "TRUE" if either X <> Ø or Y <> Ø, and

"FALSE" if both X = Ø and Y = Ø.

The real power of the "ELSE" command, however, arises with the ability to nest "IF - ELSE" combinations. For example:

```
100 IF X THEN PRINT "X-TRUE'; :
IF Y THEN PRINT " + Y-TRUE" :
ELSE : PRINT " + Y-FALSE" :
ELSE : PRINT "X-FALSE"
```

If X <> Ø, then execution of the statement will print either "X-TRUE + Y-TRUE" or "X-TRUE + Y-FALSE" depending upon whether or not Y <> Ø. On the other hand, if X = Ø, execution of the statement will print simply "X-FALSE'.

Further examples of using the "ELSE" command may be found in the Applesoft BASIC program "IF.THEN.ELSE.DEMO" shown in Listing 2.

Notes on the Patch

Applesoft BASIC is an interpreted language - meaning that execution of a statement involves the action of an interpreter that first determines what operation(s) are to be performed and then calls the appropriate routines(s). Full details of this process may be found in Kaner and Vokey (1982) and in many of Dr. Mossberg's Disassembly Lines columns in Nibble Magazine.

To effect this process, the interpreter "reads" the statement using a short routine known as CHRGET. Unlike the remainder of Applesoft BASIC, this routine resides in RAM and, hence, is modifiable. Redirecting calls to this routine to the ELSE patch allows the patch to interpret any statement before it is passed to the interpreter. In this way, the patch is able to determine whether an "IF" statement is to be executed, and is able to monitor for pending "ELSE" commands.

If an "IF" statement is encountered, the patch evaluates the expression following the "IF". If the result is FALSE (i.e., equal to Ø), the patch scans the remainder of the program line for an "ELSE" statement, noting any subsequent "IF" commands in the process. If a matching "ELSE" statement is found, TXTPTR (a pointer to the next character in the program line) is advanced to the statement following the "ELSE" and control is returned to the interpreter. Otherwise, TXTPTR is advanced to the next program line.

If the result of the expression evaluation is TRUE (i.e., not equal to Ø), control is passed to the routine that normally handles execution of TRUE "IF" statements, and a flag is set to indicate to the patch that any matching "ELSE" (and all code following it) subsequent to the "IF" on the same program line is to be ignored (i.e., not sent to the interpreter). Monitoring for a pending "ELSE" occurs until one is found or until an end-of-line token (EOL) is encountered. Thus, "ELSE" statements not preceded by a matching "IF" will be passed to the interpreter with a "SYNTAX ERROR" as the correct result, and "ELSE" statements and their associated code preceded by a matching TRUE "IF" will not be executed. Similarly, the code following the "ELSE"

ELSE.OBJ Source Code

```
                    .OR $300
                    .TF ELSE.OBJ

*------------------------------------------------------------------*
*                        IF...THEN...ELSE                          *
*------------------------------------------------------------------*

*                      Copyright (c) 1985 .                        *
*                      Softkey Publishing                          *

*------------------------------------------------------------------*
*                           EQUATES                                *
*------------------------------------------------------------------*

00B1- CHRGET    .EQ $B1       CHRGET routine
00B7- CHRGOT    .EQ $B7
00B8- TXTPTR    .EQ $B8       TeXT PoinTeR
DD7B- FRMEVL    .EQ $DD7B     FoRMula EVaLuation
D9E1- INIF      .EQ $D9E1     PART OF IF routine
00AD- IFF       .EQ $AD       IF token
00AB- GOTO      .EQ $AB       GOTO token
00C4- THEN      .EQ $C4       THEN token
D9DC- IFEXIT    .EQ $D9DC     FALSE-IF exit
00B4- ELSFLG    .EQ CHRGET+3  Else statement to skip over?
009D- FACEXP    .EQ $9D       FAC exponent
0005- LENGTH    .EQ $5        length of 'ELSE:
DEC9- ERROR     .EQ $DEC9     SYNTAX ERROR

*------------------------------------------------------------------*
*                          INITIALIZE                              *
*------------------------------------------------------------------*

0300: A9 4C     INIT    LDA #$4C        load a 'JMP'
0302: 85 B1             STA CHRGET      store at CHRGET
0304: A9 0D             LDA #START      low byte of interpreter
0306: 85 B2             STA CHRGET+1
0308: A9 03             LDA /START      high byte
030A: 85 B3             STA CHRGET+2
030C: 60                RTS             and return to BASIC

*------------------------------------------------------------------*
*                     SCAN FOR PENDING ELSE                        *
*------------------------------------------------------------------*

030D: A5 B4     START   LDA ELSFLG      pending ELSE?
030F: F0 1B             BEQ NOELSE      No, go

0311: 98        ELSCHK  TYA             Yes, save Y
0312: 48                PHA             on stack
0313: A0 05             LDY #LENGTH     use Y as index
0315: B1 B8     LOOPØ   LDA (TXTPTR),Y Get CHAR
0317: D9 96 03          CMP CMD,Y       Match?
031A: D0 0E             BNE FIXY        No, go
031C: 88                DEY             Yes, try next
031D: 10 F6             BPL LOOPØ
031F: A9 00             LDA #Ø          Clear FLAG
0321: 85 B4             STA ELSFLG
0323: 68                PLA             Adjust stack
0324: 20 DC D9          JSR IFEXIT      Move TXTPTR to EOL
0327: 4C B7 00  OUT     JMP CHRGOT      and exit.

032A: 68        FIXY    PLA             Recover Y
032B: A8                TAY

*------------------------------------------------------------------*
*                           IF CHECK                               *
*------------------------------------------------------------------*
```

```
032C: 20 8D 03  NOELSE    JSR ADVTXT    Next CHAR
032F: C9 00               CMP #0        EOL?
0331: D0 04               BNE IFTST     No, test 'IF'
0333: 85 B4     EOLOUT    STA ELSFLG    clear FLAG
0335: F0 F0               BEQ OUT       always taken

0337: C9 AD     IFTST     CMP #IFF      IF Token?
0339: D0 EC               BNE OUT       No, exit
033B: 20 8D 03  IFEVAL    JSR ADVTXT    Yes, next CHAR
033E: 20 7B DD            JSR FRMEVL    Evaluate expression
0341: A5 9D               LDA FACEXP    TRUE or FALSE (1 or 0)?
0343: F0 1D               BEQ TKELSE    FALSE, take ELSE
0345: 85 B4               STA ELSFLG    TRUE, flag pending ELSE
0347: 20 B7 00            JSR CHRGOT    Recover CHAR
034A: C9 AB               CMP #GOTO     GOTO token?
034C: F0 0E               BEQ DOIF      Yes, do it
034E: C9 C4               CMP #THEN     THEN token?
0350: F0 03               BEQ DOIF0     Yes, do it
0352: 4C C9 DE            JMP ERROR     No, SYNTAX ERROR

0355: 20 8D 03  DOIF0     JSR ADVTXT    Get CHAR after 'THEN'
0358: C9 AD               CMP #IFF      IF token?
035A: F0 DF               BEQ IFEVAL    Yes, evaluate it
035C: 20 E1 D9  DOIF      JSR INIF      Do THEN...
035F: 4C B7 00            JMP CHRGOT    and exit

* ----------------------------------------------------------------- *
*                            TAKE ELSE                              *
* ----------------------------------------------------------------- *

0362: E6 9D     TKELSE    INC FACEXP    'IF' count
0364: 20 8D 03  LOOP1     JSR ADVTXT    next CHAR
0367: 20 B7 00  LOOP2     JSR CHRGOT    Recover CHAR
036A: F0 06               BEQ TSTEOL    If Z then test for EOL
036C: C9 AD               CMP #IFF      IF token?
036E: D0 F4               BNE LOOP1     No, try next
0370: F0 F0               BEQ TKELSE    Yes, increment count

0372: C9 00     TSTEOL    CMP #0        eol?
0374: F0 BD               BEQ EOLOUT    Yes, exit

0376: A0 01     TSTELSE   LDY #1        index
0378: 20 8D 03  LOOP3     JSR ADVTXT    next CHAR
037B: D9 96 03            CMP CMD,Y     match?
037E: D0 E7               BNE LOOP2     No, try next
0380: C0 05               CPY #LENGTH   Yes, done?
0382: F0 03               BEQ TSTELSE2  Yes, one down...
0384: C8                  INY           No,
0385: D0 F1               BNE LOOP3     go again

0387: C6 9D     TSTELSE2  DEC FACEXP    any more to go?
0389: D0 EB               BNE TSTELSE   Yes, do it
038B: F0 9A               BEQ OUT       No, exit

* ----------------------------------------------------------------- *
*                          ADVANCE TXTPTR                           *
* ----------------------------------------------------------------- *

038D: E6 B8     ADVTXT    INC TXTPTR    Replace chrget
038F: D0 02               BNE EXIT      to allow processing
0391: E6 B9               INC TXTPTR+1  of BASIC commands.
0393: 4C B7 00  EXIT      JMP CHRGOT

0396: 3A 45 4C
0399: 53 45 3A  CMD       .AS ":ELSE:"
```

statement of a matching FALSE "IF" will be executed in place of the code immediately following the "IF'.

Suggestions on Extensions

One obvious extension to the patch is to remove the "on the same line" rule. This extension could be implemented by having the routine monitor both within and across program lines for, say, an "ENDIF" statement, rather than an EOL before ceasing its search for a matching or a pending "ELSE" statement. While such an approach has much commending it (it would, for example, emulate the Pascal language's implementation of "IF"), it also would require that all "IF" statements be followed at some point in the program by the "ENDIF" statement, eliminating BASIC's usual (and often useful) handling of "IF" commands. Moreover, it would require some method of keeping track of nested "IF" statements, subroutines called from within "IF - ENDIF" blocks, and numerous other details that may render the approach too unwieldly for the gains made.

Another extension is to modify the routine so that it will function from within the ProDOS BASIC system. Since patches to CHRGET do not work under the ProDOS BASIC system, an alternative approach is to flag all "IF" and "ELSE" statements with ampersands, and use the Applesoft BASIC ampersand vector to transfer control to the (appropriately modified) "ELSE" patch.

References

Grogono,P. **Programming in PASCAL (revised edition).** Don Mills, Ontario: Addison-Wesley Publishing Company, Inc., 1980.

Kaner, H. C. and Vokey, J. R. **Modifying Apple's floating-point BASIC: An & interpreter without the &.** Compute!, 1982, 5, 146-152.

Mossberg, S. **MAMA (Move aside Mr. ampersand).** Nibble, 1982, 3, 105.

ELSE.OBJ Hexdump

```
0300: A9 4C 85 B1 A9 0D 85 B2   $F260
0308: A9 03 85 B3 60 A5 B4 F0   $F762
0310: 1B 98 48 A0 05 B1 B8 D9   $8C3C
0318: 96 03 D0 0E 88 10 F6 A9   $CA1E
0320: 00 85 B4 68 20 DC D9 4C   $9B47
0328: B7 00 68 A8 20 8D 03 C9   $37B1
0330: 00 D0 04 85 B4 F0 F0 C9   $7773
0338: AD D0 EC 20 8D 03 20 7B   $D7E9
0340: DD A5 9D F0 1D 85 B4 20   $1DF0
0348: B7 00 C9 AB F0 0E C9 C4   $2DD5

0350: F0 03 4C C9 DE 20 8D 03   $73A2
0358: C9 AD F0 DF 20 E1 D9 4C   $E251
0360: B7 00 E6 9D 20 8D 03 20   $E107
0368: B7 00 F0 06 C9 AD D0 F4   $31AA
0370: F0 F0 C9 00 F0 BD A0 01   $63EB
0378: 20 8D 03 D9 96 03 D0 E7   $9940
0380: C0 05 F0 03 C8 D0 F1 C6   $1FD7
0388: 9D D0 EB F0 9A E6 B8 D0   $7B1E
0390: 02 E6 B9 4C B7 00 3A 45   $4BCB
0398: 4C 53 45 3A               $8E58
```

Bank Street Speller

By Jeff Lucia

Bank Street Speller
Broderbund Software, Inc.
17 Paul Drive
San Rafael, CA 94903
$70.00

Requirements:
A blank disk
Bank Street Speller
An NMI card or a way into the Monitor (optional)
At least 48K
COPYA

Note: This procedure only works if you have not used the copy feature on Bank Street Speller. If you already have, the copier has been destroyed.

It is well known that Broderbund probably has always had the best protections, from the release of Lode Runner, to the current dilemma, Bank Street Speller, henceforth referred to as BSS. Broderbund must spend a lot of money in creating their protections, but it is worth it for the reduction in pirated copies.

When I first purchased BSS I thought it would be easy to backup. Usually, no matter what program I buy, I can manage to bit-copy it. However, the scene changed dramatically when I tried to backup BSS. No such luck. The copied disk worked fine until it checked track $00 for a certain pattern of bytes. No matter how hard I tried to re-copy track $00 and match those bytes, the copy would always fail.

Lately though, Broderbund has added a feature that lets the user obtain ONE backup of the disk. This is fine and dandy, until both the backup and original die. Then it will cost you $5.00 for each new backup. The only way I was able to obtain more than one backup was to modify this feature so it will not write to the original disk and tell it that you have already backed-up the disk.

BSS has a unique copier because it will check if the disk has already been copied in several places. Our objective is to find out how BSS's copier prevents you from making more than one backup, and how to utilize it in such a way that it will let you make as many backups as you need. The answer is simple. To find out how BSS prevents you from making more than one backup, just put a write protector on your original disk. Sooner or later, the program will ask you to remove the write protect tab. This occurs when the copier is reading tracks 1-9. While it's reading those tracks, it will try to write, on track $01, that you have made your one backup. Finding a way to utilize BSS's copier to mass produce backups is a little harder to do.

In order to produce many backups of BSS, we must find the exact routine which will check to see if the disk is write protected. Next, find the routine which tries to write to track $01. Last but not least, change the routine so it doesn't check at all (just makes you backups). So, let us explore the following routine which does the checking:

```
42EC- LDY #09    ; START TRACK $09
42EE- LDA #01    ; END TRACK $01
42F0- JSR $40ED  ; PREPARE TO READ
42F3- JSR $B400  ; READ TRACKS INTO $5000
42F6- JSR $B40F  ; ATTEMPT TO WRITE
42FE- BCS $430E  ; SAY "WRITE PROTECTED"
  .       .
  .       .
430C- BCC $4314  ; IF OK, GO WRITE

4311- JSR $42EC  ; IF NOT OK, GO READ
```

What does this mean? Well, $42EC tells the copier which track to start on and $42EE tells the copier which track to end on. Now we hit the routine which gets the copier ready to read tracks $01 - $09 this routine is $42F0. Looking at $42F3 we will find out that it turns on the drive and reads in the tracks. Finally, we have found the routine which does all the checking. At $42F6 the copier will try to write to the original disk and tell it that you have made your ONE backup. The rest of the routines that I have listed ($430C & 4311) just ask you to remove the write protect tab then start reading from track $09 again.

There are two ways to defeat the checking routine, an easy way and a hard way. I gather that you all want to do it the easy way, but unfortunately those of you who don't have any means of entering the Monitor must use the hard way.

Note: Always keep your original BSS disk write-protected during this procedure. Do not take it off until you have finished making the number of backups desired.

The Easy Way

1) Write-protect your BSS disk.

2) Boot it up.

PR#6

3) Hit Escape to enter the copier.

4) When the menu appears, enter the Monitor by your favorite means.

5) Once in the Monitor, change $42F6-42F8 to JSR $4314.

42F6:20 14 43

6) Execute the copier.

4000G

7) Use COPYA on the dictionary side.

8) Congratulations! You now have no excuse to ever make a spelling error again!

The Hard Way

1) Write-protect your BSS disk.

2) Boot it up.

3) As soon as the Logo has been totally loaded, open the disk drive door and then hit reset. When you get an "R" in the upper left hand corner, hit reset again. Hit reset one more time. You should now have an Applesoft prompt. If you don't get a prompt, keep hitting reset until you do.

4) Enter the Monitor and close the drive door.

CALL-151

5) Make these necessary modifications to the program residing in $9000 so it will read the Utility Menu.

910D: 4C 59 FF
B400G

90F9G

(reads the Utility Menu into $1E00)

If, for some reason, the Utility Menu doesn't read in, repeat steps 1-5.

6) When the computer beeps you should be in the Monitor. If not, repeat steps 1-5. Modify the built in copy program so it doesn't write to the original.

42F6:20 14 43

7) Run the program

4000G

8) Copy the disk.

9) COPYA the dictionary disk.

To make other backups just use the procedure again, always using the original BSS. Hopefully none of you will ever have spelling errors again.

Where In The World Is Carmen Sandiego?

by Ronald Wilson

Broderbund Software
17 Paul Dr.
San Rafael, CA 94903

Requirements
Apple][, 64K minimum
Modified F8 ROM in Hardcore COMPUTIST No. 19
DOS 3.3 (or a fast DOS)
A blank disk or two

"Where In the World Is Carmen Sandiego?" is a new and interesting release from Broderbund Software. By playing the game, we can learn something about the cultures of other countries. However, the program is heavily protected.

Developing the Softkey

I started by installing the modified F8 ROM (Hardcore COMPUTIST No. 19) in my apple //e and booting the Carmen Sandiego disk. When the title page appeared, I hit the Reset key. The modified ROM should have stopped and waited for me to press a key, but the program cleared the screen and rebooted. After rebooting the disk, I saw that the red lights on my Microsoft RAM card were on when the title page appeared. This meant that the program was using the 16K RAM card for itself. Fortunately, the red lights went off after a man had appeared on the screen. Then I pressed Reset followed by M to go directly to the Monitor. Looking through memory, I found that HIRES pages 1 and 2 were not used. I used the search routine of the modified ROM to try to find the entry point of the program. I tried

```
200:FF 3 AD 50 C0 N 800.BFFFS
```

to find out where the graphic mode was turned on. A bit of searching determined that the entry point was $6000. Then I relocated parts of the program as follows:

Original Location	Relocated To:	Number of Pages
$0000-07FF	$2100-28FF	$8
$0800-1FFF	$0800-1FFF	$18
$6000-88FF	$6000-88FF	$29
$8900-BFFF	$2900-5FFF	$37

Then I wrote a restart routine at $2000 to relocate the program, turn on the RAM card and jump to the entry point. I saved all the code to disk as a file named "CAR". Then I wrote a short routine to save the RAM card's banks 1 and 2 to disk, and another routine to restart the RAM card, turn it off, and BRUN the file "CAR". We now have the boot side of Carmen Sandiego in the form of binary files, while the other side was COPYAable in the first place.

Step by Step

1) INIT a blank disk to create a slave disk with no HELLO program.

```
INIT HELLO
DELETE HELLO
```

2) Boot the Carmen Sandiego disk. After the man appears, press Reset, followed by "S" (to Save the lower 8 pages of memory).

3) Boot your slave disk so you can save the lower RAM to disk.

```
6⌃P
BSAVE P0-P7,A$2000,L$800
```

4) Boot the Carmen Sandiego disk again. After the man appears, press Reset followed by "M" to enter the Monitor.

5) Move some of the code out of the way to a safe place.

```
2800<800.8FFM
2900<8900.BFFFM
```

6) Boot the slave disk.

```
6⌃P
```

7) Move the code that lived at $800 back where it was.

```
CALL-151
800<2800.28FFM
```

8) Type in the restart program (Hexdump 1) after this softkey. This will set things up for the program to run.

9) Get the lower memory that you saved earlier and put it at $2100.

```
BLOAD P0-P7,A$2100
```

10) Modify the program so that nothing new is put into the RAM card.

```
6034:EA
6035<6034.6053M
```

11) Fix DOS so you can BSAVE more than $8000 bytes.

```
A964:FF
```

12) At $7FD, put a JuMP to the restarting routine at $2000. Then BSAVE the main part.

```
7FD:4C 00 20
BSAVE CAR,A$7FD,L$8103
```

13) Type in Hexdump 2. This is a short routine to move RAM card memory to $1000. Execute it with

```
E00G
```

14) Type in Hexdump 3. This one puts the memory at $1000 back into the RAM card and BRUNs the file "CAR". BSAVE it and the RAM card memory.

```
BSAVE CARMEN SANDIEGO,A$F00,L$4100
```

15) COPYA the other side to another blank disk, or use the back side of your newly deprotected Boot disk.

Hexdump 1 (program restart)

```
2000: AD 52 C0 AD 57 C0 AD 50   $CC16
2008: C0 A0 08 A2 00 BD 00 21   $7E1F
2010: 9D 00 00 E8 D0 F7 EE 0F   $69A6
2018: 20 EE 12 20 88 D0 EC A0   $8A37
2020: 37 A2 00 BD 00 29 9D 00   $F942
2028: 89 E8 D0 F7 EE 25 20 EE   $364F
2030: 28 20 88 D0 EC AD 83 C0   $2A70
2038: AD 83 C0 4C 00 60 00 00   $81FC
```

Hexdump 2 (RAM card to $1000)

```
0E00: A0 00 AD 80 C0 A2 10 B9   $BD0E
0E08: 00 D0 99 00 10 C8 D0 F7   $3414
0E10: EE 09 0E EE 0C 0E CA D0   $3C0E
0E18: EE AD 8B C0 AD 8B C0 B9   $CFDF
0E20: 00 D0 99 00 20 C8 D0 F7   $9E55
0E28: EE 24 0E EE 21 0E D0 EF   $477B
0E30: AD 81 C0 60               $10A8
```

Hexdump 3 ($1000 to RAM card)

```
0F00: 20 58 FC AD 81 C0 AD 81   $3281
0F08: C0 A0 10 A2 00 BD 00 10   $9DB8
0F10: 9D 00 D0 E8 D0 F7 EE 0F   $12E1
0F18: 0F EE 12 0F 88 D0 EE AD   $F5A6
0F20: 8B C0 AD 8B C0 BD 00 20   $AE33
0F28: 9D 00 D0 E8 D0 F7 EE 27   $798E
0F30: 0F EE 2A 0F D0 EF AD 81   $9895
0F38: C0 20 58 FC 20 E0 9E A2   $29D4
0F40: 38 A9 A0 9D 75 AA CA D0   $9E6A
0F48: F8 A9 C3 8D 75 AA A9 C1   $4CA0

0F50: 8D 76 AA A9 D2 8D 77 AA   $45CA
0F58: 4C 8E A3 00               $A002
```

BANK STREET

By Jordi Le Vant

Bank Street Writer //c
Broderbund Software, Inc.
17 Paul Drive
San Rafael, CA 94903-2101

Requirements
48K and up (The program requires 128K)
Super IOB v1.5
A sector editor
A slave disk with deleted HELLO
A blank disk
If you have a //c, you need the disk controller ROM from a][or //e saved as a binary file.

Although the program requires an Apple //c or a //e with 128K (extended 80 column card) to run the program, you don't need that to crack the program. All of the tracks except Ø have fairly normal sectors, but with modified address and data field headers and trailers. Track Ø, sector Ø is almost normal with the exception of a bad checksum, which the controller card ROM doesn't check while booting.

When the program is booted, this sector is loaded into page $Ø8 and is jumped to at location $Ø8Ø1 from the controller card ROM (boot Ø). This program at page $Ø8 (boot 1) then loads the main loading program at page $14 to page $1A. This loader is all on track zero and is one long stream of data which can't be read easily with Super IOB if at all. I resorted to boot tracing the first two parts of the boot process to get this loader into memory.

1) First get into the monitor.

CALL-151

2) Move the controller card ROM into RAM so we can change it to jump to the monitor instead of location $Ø8Ø1 after loading boot 1.

6600<C600.C700M
66F9:59 FF
6600G

The drive will load in the first stage normally and return you to the monitor.

3) Turn the drive off.

CØE8

4) Examining the program at page $Ø8, you will find a JMP $14ØØ at location $82Ø. This jumps to the program it loads off of track zero which is the main loader for the actual program. Since we want to stop after the main loader is in memory, boot 1 needs to be changed to jump to the monitor instead of the loader.

821:59 FF

5) At this point, if you try to restart by typing 66ØØG boot Ø will load boot 1 again over the changes made. Location $6659 tells what page to load boot 1 into, so you should change this to store it somewhere in ROM so as not to do any damage.

6659:FØ

6) We had also changed the controller ROM to jump to the monitor. This needs to be changed back to jump to boot 1 at its usual location.

66F9:Ø1 Ø8

7) Now reboot the modified controller with

66ØØG

Then turn the drive off again.

CØE8

Now you can examine the main loader from page $14 to page $1A. Since this program is fast at loading the main program I thought it would be great to modify it and still use it.

WRITER (128K)

After searching for a while, I was able to find the routines that searched for the headers and trailers.

8) By making the following changes in memory the loader will be able to read normal sectors.

```
1449:D5
1453:AA
145D:AD
14D4:DE
152E:D5
1538:AA
1543:96
156A:DE
```

9) Boot a slave diskette with a deleted HELLO program.

C600G

10) Save the loader program on disk for later.

BSAVE LOADER,A$1400,L$700

11) Now we need to make a modified RWTS to read the protected disk for Super IOB to use. Get into the monitor and copy the RWTS to lower memory.

1900<B800.BFFFM

12) Make the following changes to the RWTS you just put at $1900, and save it onto your Super IOB disk.

```
1A55:A5
1A5F:96
1A6A:BF
1A91:9A
1A95:18 60
19E7:D4
19F1:D5
19FC:D6
1A35:D7
1A39:18 60
BSAVE RWTS.BSW2C,A$1900,L$800
```

13) Now you can use the NewSwap controller for Super IOB v1.5. The only changes are the name of the RWTS that is loaded, and the starting track is now 1, since everything except track Ø needs to be copied.

Install Controller 1 with this softkey into Super IOB (using whatever method you use) and copy the original BSW //c disk.

14) We are going to use Super IOB in a different way now. Since the Bank Street Writer's loader is in a binary file at this point, we can BLOAD the loader file directly into the sector buffer and write it out a sector at a time onto track Ø. The result will be that the portion loaded into $1400 (during the boot, not now) will be put in sector 1, and stored sequentially from there until page $1A is stored in sector 7. This can be done by hand with fancy sector editors, but making a program do it for you is easier (with no mess).

Install Controller 2 with this article into Super IOB. RUN Super IOB (don't format) and use it to "copy" the binary file to track Ø, sectors 1-7 of the copy disk. Just treat the disk with the binary file as the source disk.

15) Start up your sector editor and load in track Ø, sector Ø from a normal DOS 3.3 disk.

16) This sector usually loads RWTS into page $B7 and up when booting, so a couple of changes need to be made to load the BSW loader into page $14 and jump to the correct location. Change the following bytes of the sector and write it out to track Ø, sector Ø of the **copy**.

Byte	From	To
$4A	$6C	$4C
$4B	$FE	$ØØ
$4C	$Ø8	$14
$FE	$B6	$13

17) Now the disk should boot to the editor without any problems, but if you press Escape during the boot and go to the utility menu, you will not be able to save the changes you make onto your deprotected system disk. The utility program is in BASIC and makes a call to modify all the address and data field headers and trailers in its RWTS to save the changes. Since all these markers are normal now, thi subroutine needs to be eliminated. You can do this by changing track 4, sector 7, byte $3 from $AØ to $6Ø.

You now have a COPYAable backup of Ban Street Writer //c.

controller 1

```
1000 REM BSW //C CONTROLLER
1010 TK = 1 : LT = 35 : ST = 15 : LS = 15 : CD = WR : FAS
     = 1
1020 GOSUB 360 : GOSUB 490 : GOSUB 610
1030 GOSUB 360 : GOSUB 490 : GOSUB 610 : IF PEE
     (TRK ) = LT THEN 1050
1040 TK = PEEK (TRK ) : ST = PEEK (SCT ) : GOTO 102
1050 HOME : PRINT "COPYDONE" : END
10010  PRINT  CHR$  (4  )  "BLOAD
     RWTS.BSW2C,A$1900"
```

controller 1 checksums

```
1000  - $356B     1040  - $055A
1010  - $2545     1050  - $8B03
1020  - $C808     10010 - $2E18
1030  - $FA9C
```

controller 2

```
1000 REM BSW //C LOADER CONTROLLER
1010 CD = WR : GOSUB 490
1020 PRINT CHR$ (13 ) CHR$ (4 ) "BLOAD
     LOADER,A$2700,S" SO ",D" DV
1030 TK = 0 : ST = 1 : CD = RD : GOSUB 490
1040 GOSUB 430 : GOSUB 100 : ST = ST + 1 : IF ST
     8 THEN 1040
1050 HOME : PRINT "DONE" : END
```

controller 2 checksums

```
1000  - $356B     1030  - $A5FA
1010  - $46B0     1040  - $0BD4
1020  - $D6FC     1050  - $D5C4
```

Word

By Larry Jasonowicz

Word Challenge
Hayden Software
600 Suffolk St.
Lowell, MA, 01854
$39.95

Requirements:
Word Challenge
A way into the Monitor
Super IOB v1.2
1 or 2 disk drives
A blank disk

Word Challenge is a word game similar to Boggle in which you play against the computer (LEX). It sets up a matrix of letters and the object is to make as many words from this matrix as possible. There is an option to change many aspects of the game (time limit, score, # of letters, etc). The rest of this article is somewhat technical, so if you would like to just copy Word Challenge, skip to the section on just the steps necessary.

The Challenge Begins

I always like to have a backup of my disks in case of an emergency. I first tried Super IOB with the swap controller on Word Challenge but was disappointed when it could not read the disk. I also tried a couple of bit copiers but could not get a reliable copy. At this point I decided to attempt to determine the copy protection scheme and remove it. The first step is to examine the raw track data. My favorite disk "snooper" is CIA from Golden Delicious Software. As I was skimming the different tracks and sectors of the disk I was confronted by a nightmare. Every sector (except on track 0) had a different set of address and data headers! Also, the sector numbers were strange; most were numbered above 0F. This meant I must look at Word Challenge's RWTS and compare it to normal DOS 3.3 RWTS. The method (for any disk) is as follows:

1) Boot the original disk, and after the drive stops get into the monitor your favorite way.

2) Move the protected RWTS down to $1805.

```
1800<B800.BFFFM
```

3) Boot a normal DOS 3.3 Disk with a short HELLO program.

```
C600G
```

The protected RWTS is now at $1800 and normal DOS RWTS is at $B800.

4) Then use the slightly unknown monitor command "V" (for Verify memory).

```
1800<B800.BFFFV
```

This will display the differences in the ranges of memory from $1800 To $1FFF and $B800 To $BFFF.

To my disappointment there were very few significant differences between the two RWTSs (usually, the range of memory from $BB00 To $BC55 will be different because it holds a couple of buffers).

The Challenge Goes On

For lack of anything else to do, I decided to boot code trace, even though the Word Challenge disk loads data twice. (If you would like some tips on boot code tracing, see the end of this article.) The Word Challenge booting process jumps back and forth between pages 8 and 9, and then from nearly the end of page 8 it jumps to $B400. From there it jumps to $B600 and then to $B700. After examining the code from $B600 to $B7FF I found this is where the secret of the whole disk was. This is where the disk does all the loading, coding and decoding of the sectors! The people at Hayden went through a lot of trouble coding the sectors and then hiding this change. They do some fancy EORing and ANDing the original sector number and then storing this value in the sector translate table ($BFB8 to $BFC7). They also pull different values from a table and store them in the data and address header read locations. At this point the sector is finally read into memory. Then the sneaky little rascals immediately restore these changes back to normal with some more fancy code so when examining $B800 to $BFFF you will never notice the changes.

I had to figure out a way to incorporate the coding/decoding scheme on the Word Challenge disk into Super IOB. Since Word Challenge uses the normal location for the IOB ($B7E8) and Super IOB uses $30A, I had to modify a small portion of the Word Challenge RWTS and save it as a file. This file is then called just before and after reading each sector. Super IOB had to be modified slightly to use this file.

Challenge

During the copying process, several bytes needed to be changed so that the copy can read a standard disk format. Here is a rundown of these changes:

```
------------------------------------------
 Track   Sector  Byte  From  To
------------------------------------------
 $00     $05     $71   $AC   $28
                 $72   $ED   $18
                 $73   $B7   $60
                 $9A   $59   $4C
                 $9B   $B8   $68
                 $9C   $BF   $B7
------------------------------------------
```

Copying the Disk

1) Type in the following code and save it

BSAVE WORD,A$1800,L$4F

```
1800:AD 0E 03   LDA $030E
1803:AC 0F 03   LDY $030F
1806:59 B8 BF   EOR $BFB8,Y
1809:48         PHA
180A:29 07      AND #$07
180C:AA         TAX
180D:0A         ASL
180E:0A         ASL
180F:0A         ASL
1810:0A         ASL
1811:19 B8 BF   ORA $BFB8,Y
1814:99 B8 BF   STA $BFB8,Y
1817:68         PLA
1818:18         CLC
1819:6D 0E 03   ADC $030E
181C:29 07      AND #$07
181E:A8         TAY
181F:BD 32 18   LDA $1832,X
1822:BE 32 18   LDX $1832,Y
1825:8D E7 B8   STA $B8E7
1828:8D 55 B9   STA $B955
182B:8E F1 B8   STX $B8F1
182E:8E 5F B9   STX $B95F
1831:60         RTS
1832:D5 D4      CMP $D4,X
1834:D2         ???
1835:CA         DEX
1836:AA         TAX
1837:A9 A5      LDA #$A5
1839:95 00      STA $00,X
183B:AC 0F 03   LDY $030F
183E:B9 B8 BF   LDA $BFB8,Y
1841:29 0F      AND #$0F
1843:99 B8 BF   STA $BFB8,Y
1846:A9 D5      LDA #$D5
1848:A2 AA      LDX #$AA
184A:20 25 18   JSR $1825
184D:18         CLC
184E:60         RTS
```

2) Install the controller at the end of this article in Super IOB and RUN.

3) Run the program on Word Challenge (make sure the file WORD is on the same disk as IOB.OBJO).

Track 0 has standard DOS format, with the exception of sector 1 which is not used. So when Super IOB copies track 0 it skips sector 1. Line 100 does the coding and decoding of the sectors.

Now, if only I could find all the words LEX does...

controller

```
100 BF = 0 : POKE TRK , TK : POKE SCT , ST : POKE CMD
     , CD : IF TK = 0 THEN 103
102 IF CD = RD THEN CALL 6144 : CALL IO : CALL 6203
     : GOTO 105
103 IF TK = 0 AND ST = 1 THEN 105
104 CALL IO
105 POKE BUF , PEEK (BUF ) + 1 : IF PEEK (BUF ) =
     > MB THEN BF = 1
1000 REM  WORD CHALLENGE CONTROLLER
1010 TK = 0 :ST = 0 :LT = 35 :CD = WR
1020 T1 = TK : GOSUB 490 : RESTORE
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 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 : PRINT "DONE^ WITH^ COPY" :
     END
5000 DATA 6^ CHANGES
5010 DATA 0 ,5 ,113 ,40 ,0 ,5 ,114 ,24
5020 DATA 0 ,5 ,115 ,96 ,0 ,5 ,154 ,76
5030 DATA 0 ,5 ,155 ,104 ,0 ,5 ,156 ,183
10055 PRINT : PRINT CHR$ (4 ) "BLOAD^ WORD"
```

controller checksums

```
100   - $896F     1060  - $4F6A
102   - $9CC9     1070  - $476F
103   - $051B     1080  - $C699
104   - $4C0D     1090  - $1E10
105   - $EFD5     1100  - $A275
1000  - $EC87     5000  - $770C
1010  - $9E37     5010  - $E124
1020  - $38B8     5020  - $682F
1030  - $2EB9     5030  - $F17B
1040  - $5E98     10055 - $A25B
1050  - $4683
```

Hardcore COMPUTIST is not related to B.M.E Enterprises in any way.

THE SECURITY DISK
PROTECTION / DEPROTECTION

Hard-Core Programmers team up to give you software designed to Protect Your Private Files, Plus give you the ability to Analyze & Copy your other ''Locked-Up'' Disks to standard DOS 3.3 format. Not Locked Up. Source codes included. A special ''Tips & Techniques'' program included. For APPLE II series.

PROTECTION SCHEMES

PASSWORD PROTECTOR CATALOG MOVER NO-LIST PROTECTOR
SINGLE/MULTIPLE PASSWORDS
PROGRAM ENCRYPTION DISK ENCRYPTION

DISUN: :THE DISK UNLOCKER

UNLOCKS DISK PROTECTION SCHEMES RECOPIES TO DOS 3.3
ANALYZE ANY APPLE II DISK
REPAIRS DAMAGED DISKS RECOVERS LOST DATA

NOT LOCKED UP **1 OR 2 DRIVES**

To order send $29.95 CHECK/MO To:
B.M.E. ENTERPRISES, P.O. BOX 191-H, KILA, MT 59920
Dealer Inquiries Invited Phone 406-755-6816

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

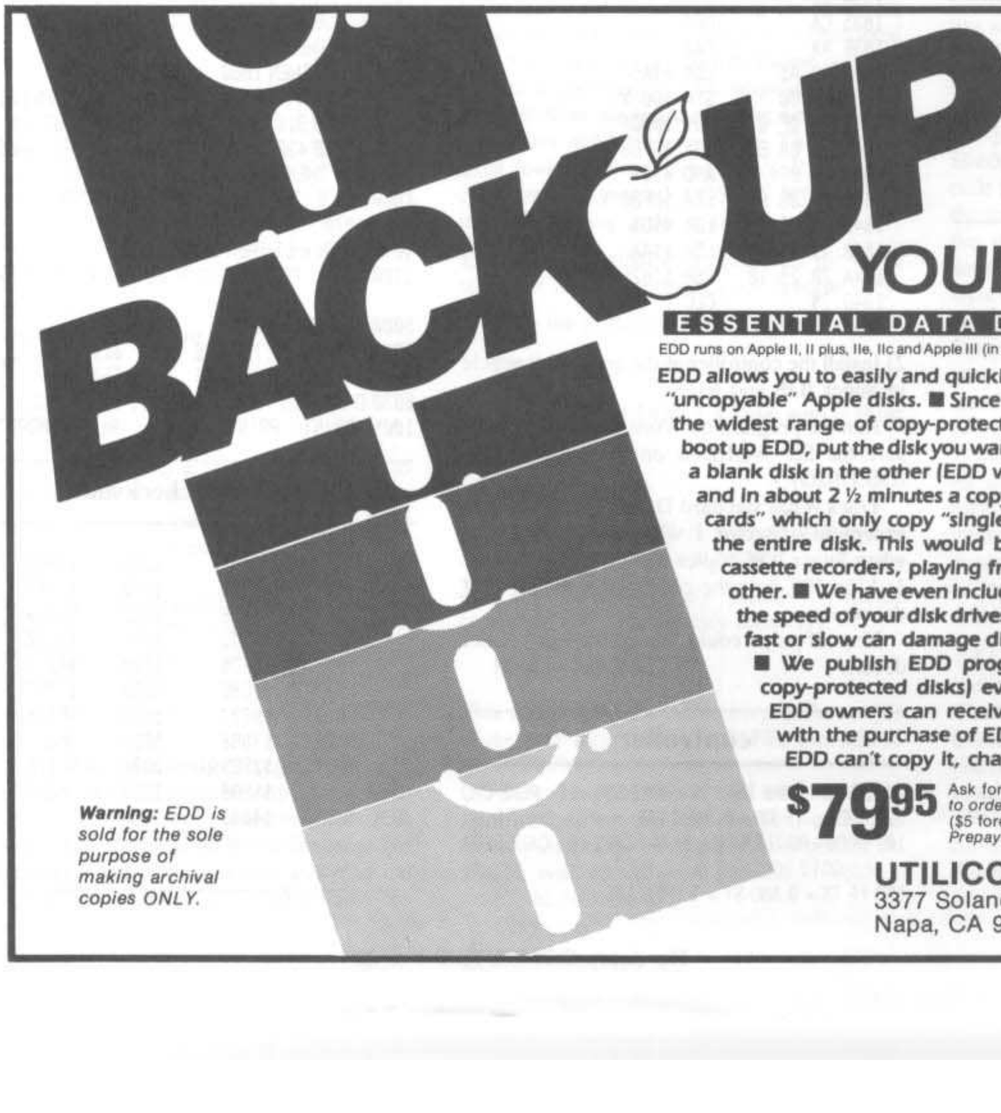
"a very enlightening piece of software/book . . . top of my list for good buys" - H.S., BLAINE, MINNESOTA

"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 AVAILABLE AT YOUR LOCAL COMPUTER STORE

YOUR DISKS

ESSENTIAL DATA DUPLICATOR III™

EDD runs on Apple II, II plus, IIe, IIc and Apple III (in emulation mode) using one or two disk drives.

EDD allows you to easily and quickly make back up copies of your "uncopyable" Apple disks. ■ Since EDD has been preset to copy the widest range of copy-protections possible, you just simply boot up EDD, put the disk you want to copy in one disk drive and a blank disk in the other (EDD will work using one drive also) and in about 2 ½ minutes a copy is made. ■ Unlike the "copy-cards" which only copy "single load" programs, EDD copies the entire disk. This would be similar to hooking up two cassette recorders, playing from one, and recording to the other. ■ We have even included an option so you can check the speed of your disk drives because drive speeds running fast or slow can damage disks and cause other problems. ■ We publish EDD program lists (information about copy-protected disks) every couple of months, which EDD owners can receive. The current list is included with the purchase of EDD. ■ The bottom line is this; if EDD can't copy it, chances are nothing will.

$79.95 Ask for EDD at your local computer store, or, *to order direct;* send **$79.95** plus $2 shipping ($5 foreign). Mastercard/Visa accepted. *Prepayment required.*

Warning: EDD is sold for the sole purpose of making archival copies ONLY.

UTILICO MICROWARE
3377 Solano Ave., Suite #352
Napa, CA 94558 (707) 257-2420

THE MOST POWERFUL BACK-UP UTILITY
YOU'VE EVER SEEN...

$59.95 *add $5 ship.*

* Back-ups 1/2, 1/4, 3/4 tracks.
* Automatic back-up options.
* No parms needed for most of the back-ups.
* Excellent DOS copy on flip side.

Ask for our other products RAM-LOCK (for L-Smith)
And also SHOUGI (Japanese chess-type game.)

ART GALLERY
Yoshinoya Bldg, 438 Sasu-machi, Chofu-shi
Tokyo 182, Japan
Send money or check. VISA/MASTER CARD accepted.

DEALERS

Overstocked on *computer magazines* that seem to have MORE ADS than articles?

Put Hardcore COMPUTIST on your shelves, instead!

Here's how:

Send in the coupon below and we'll put your dealer info in the mail. DON'T WAIT! Apple computer users everywhere salute you!

Yes! Please send me information on how I can obtain copies of Hardcore COMPUTIST at substantial savings.

Dealer Name ____________________
Address ____________________
City ________ State ________ Zip ________

Send to: Hardcore COMPUTIST, PO Box 110816, Tacoma, WA 98411 or call (206) 474-5750.

Unlock Software Mysteries!

The Senior PROM //c, //e: An affordable hardware & software device that combines many features into one. Included are:

- Ability to enter the Monitor ANY time.
- Capture all memory to a normal DOS disk.
- Restart a captured program from disk.
- Advanced sector, track, memory editor.
- ROM resident DOS with complete utils.
- Read and edit copy-protected software.
- Mini-Assembler, Step & Trace in ROM.
- Study disk boots with RAM test pattern.
- Copy volatile RAM to accessible RAM.
- Copy all of Main RAM to Aux, or reverse.
- Nothing else like it available for the //c!

The Senior PROM combines the functions of a "Copy Card", a nibble copier, a sector editor, an old F8 Monitor ROM and much more into a single device. **Everything is in ROM, instantly available when needed.** Does not use a peripheral slot and **does not** compromise compatibility!

$88.95. Call 317-743-4041 for VISA Mon-Fri, 10-5 EST. **$79.95** check or money order direct to: Cutting Edge Enterprises, Box 43234 Ren Cen Station, Detroit MI, 48243. Call modem 313-349-2954. Specify //c, or "Standard" or "Enhanced" //e ROMs.

echo plus copy program

FORGET ABOUT PARAMETERS.

Get right down to backing up your Apple II software with the advanced, easy-to-use Echo Plus™.

Now you can copy almost all, even the best protected disks without bothering with up to 500 parameters as with ordinary copy programs. What's more, Echo boots up in under 4 seconds, copies most disks in under 2 minutes.

For the toughest disks, you just switch to the exclusive automatic disk copier. It selects the right tracks, reads in and analyzes each, adjusts parameters, then copies each track, all automatically.

Echo Plus gives humans a break with an easy-to-read screen format, cursor-controlled menus and real command flexibility. Plus a 42 page comprehensive–and comprehensible–manual that even details several copy protection schemes.

You also get drive speed in RPMs, drive alignment, certify and erase features, and a sophisticated track editor.

If you can find a more powerful, easier to use program at a better price, buy it. If not, send us your check or money-order now. Only $59.95 plus $5.00 shipping and handling (In Massachusetts, add $3.00 sales tax.)

For the Apple II, II+, IIe, or IIc with at least 48K RAM and one disk drive (DOS 3.3).

AGRANAT SYSTEMS
10 Winthrop Circle, Weston, MA 02193
Dealer inquiries invited.

Order Form

Issue	Mag	Disk
	$4.75	$9.95
HC 24	☐	☐
HC 23	☐	☐
HC 22	☐	☐
HC 21	☐	☐
HC 20	☐	☐
HC 19	☐	☐
HC 18	☐	☐
HC 17	☐	☐
HC 16	☐	☐
HC 13	☐	☐
HC 11	☐	☐
HC 7	☐	☐
HC 4	☐	☐
HC 1	☐	see below
C 3	☐	☐
C 2	☐	☐
C 1	☐	see below
HC 1 /C 1	disk only	☐

Send me the back issues indicated above:

ID# ______________________ HC25

Name ______________________

Address ______________________

City ______________________

State ________ Zip ______________

Phone ______________________

VISA/MC _____ - _____ - _____ - _____

Exp ______________________

Signature ______________________

Send check or money order to:
Hardcore COMPUTIST, PO Box 110846-T, 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.

Offer good while supply lasts.

Back Issues of Hardcore COMPUTIST

and *CORE are PACKED with information that you won't want to miss!

Hardcore COMPUTIST 24: Softkeys for Electronic Arts software, Grolier software, Xyphus, F-15 Strike Eagle, Injured Engine; Readers' Softkeys to Mr. Robot And His Robot Factory, Applecillin II, Alphabet Zoo, Fathoms 40, Story Maker, Early Games Matchmaker, Robots Of Dawn; Features: Essential Data Duplicator copy parms; Core: Direct Sector Access From DOS

Hardcore COMPUTIST 23: Softkeys for Choplifter, Mufplot, Flashcalc, Karateka, Newsroom, E-Z Draw; Readers' Softkeys to Gato, Dino Eggs, Pinball Construction Set, TAC, The Print Shop: Graphics Library, Death In The Caribbean; Features: Using A.R.D. To Softkey Mars Cars, How To Be The Writemaster; Core: Wheel Of Money

Hardcore COMPUTIST 22: Softkeys for Miner 2049er, lode Runner, A2-PB1 Pinball: Readers' Softkeys to the Heist, Old Ironsides, Grandma's House, In Search of the Most Amazing Thing, Morloc's Tower, Marauder, Sargon III; Features: Customized Drive Speed Control, Super IOB version 1.5; Core: The Macro System

Hardcore COMPUTIST 21: Softkeys for DB Master version 4+, Dazzle Draw, Archon, Twerps; Readers' Softkeys to Advanced Blackjack, Megaworks, Summer Games, College Entrance Exam Prep, Applewriter revisited; Features: Demystifying The Quarter Track; Core: Proshadow: A ProDOS Disk Monitor

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 13: Softkeys for Laf Pak, Beyond Castle Wolfenstein, Transylvania, 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 ProDOS 1.0.1 BSAVE Bug / REVIEW: Enhancing Your Apple / Locksmith 5.0 and 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 7: Softkeys for Zaxxon, Mask of the Sun, Crush, Crumble & Chomp, Snake Byte, DB Master, & Mouskattack; Features: Making Liberated Backups That Retain Their Copy Protection, S-C Assembler: Review, Disk Directory Designer; Core: COREfiler: Part 1, Upper & Lower Case Output for Zork

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. But disks are still available for ALL sold-out issues of Hardcore COMPUTIST upon request.**

Don't *TYPE IN* programs that appear in Hardcore COMPUTIST.

Order the Library Disk, instead!

Each month a Library Disk with all the programs that appeared in the previous issue of Hardcore COMPUTIST is prepared for **SMART READERS** like you who have *better* things to do with their time than type in program listings. Please use the order form to the left to order disks.

WARNING!

You are entering DANGEROUS territory

The

Book of Softkeys

shows you how to softkey (remove copy-protection from) commercial software.

Volume I (157 pages) **contains softkeys for: Akalabeth, Ampermagic, Apple Galaxian, Aztec, Bag of Tricks, Bill Budge's Trilogy, Buzzard Bait, Cannonball Blitz, Casino, Data Reporter, Deadline, Disk Organizer II, Egbert II Communications Disk, Hard Hat Mack, Home Accountant, Homeword, Lancaster, Magic Window II, Multi-disk Catalog, Multiplan, Pest Patrol, Prisoner II, Sammy Lightfoot, Screen Writer II, Sneakers, Spy's Demise, Starcross, Suspended, Ultima II, Visifile, Visiplot, Visitrend, Witness, Wizardry, Zork I, Zork II, Zork III, plus how-to articles and program listings of need-to-have programs used to make unprotected backups.**

Volumes II and III are being compiled now!

YES, I want __ copies of The Book Of Softkeys Volume 1. I have enclosed $20 per book. Foreign orders add 20%. U.S. funds drawn on U.S. banks. Washington state orders add 7.8% sales tax. Send your orders to: **Softkey Publishing, PO Box 110816, Tacoma, WA 98411**

Name______________________________ HC25-V1

Address______________________________

City______________ State__________ Zip__________

Country______________ Phone______________

Visa,MC______________________ Exp.__________

Signature______________________________

If you want to make backups, then you want The Book Of Softkeys Volume I, the only resource that teaches you how to backup your expensive Apple][software.