For The Serious User Of Apple][Computers

Hardcore

COMPUTIST

Issue No. 22 $3.75

Softkeys For:

Miner 2049er

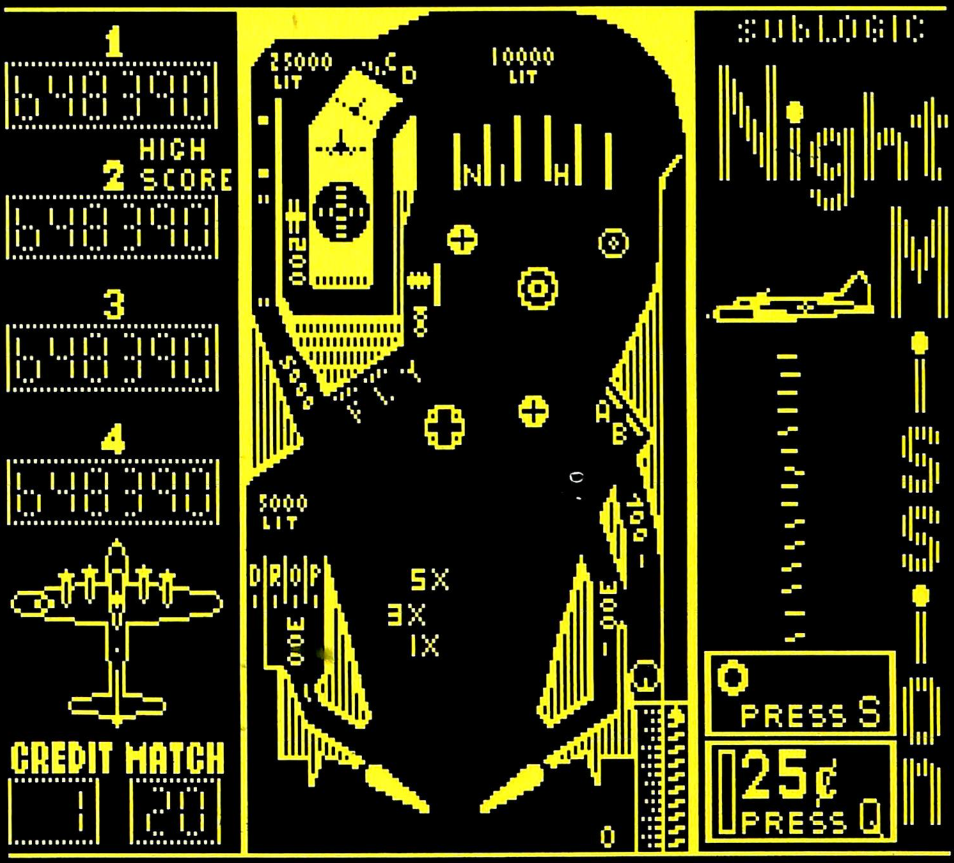
Lode Runner

A2-PB1 Pinball

Customized Drive Speed Control

Super IOB

Core's Macro System



2 ID#NJ 24 5 issue(s) left in SUB
RON LUKACS
10 BELL AVENUE
FORDS NJ 08863

PEEL HERE

BULK RATE
U.S. Postage
PAID
Tacoma, WA
Permit No. 269

Hardcore COMPUTIST
PO Box 110846-T
Tacoma, WA 98411

BACKUP PROTECTED SOFTWARE with COPY II PLUS™ ver. 5

From the team who first brought you **COPY II PLUS** in 1981 comes a completely updated disk backup utility for your Apple // computer. New features include:

- **Fully automatic bit copy*.** All parameters are stored on disk. Simply type in the name of the program you wish to backup, and **COPY II PLUS** does the rest!
- **New utilities** including Alphabetize Catalog, Fast 2-pass Disk Copy on a //c or //e, and an all-new Sector Editor.
- **Supplied on a standard DOS** diskette. Runs on the Apple //, Apple //+, Apple //e, Apple //c. Requires 64K and one or two disk drives.

Increase the power of your Apple //...
Use **COPY II PLUS™ 5.0**

Call M-F 8-5:30 (W. Coast time) with your VISA MasterCard : **503/244-5782.**
Or send a check (add $3 s/h, $8 overseas) to

$39.95

CENTRAL POINT Software, Inc.
9700 SW Capitol Hwy. #100
Portland, OR 97219

* We update Copy II Plus regularly to handle new protections; you as a registered owner may update at any time for 1/2 price! (To update, just send original disk and $20.)

This product is provided for the purpose of enabling you to make archival copies only.

DISC COMMANDER

COPYRIGHT 1985 WILLIAM STEPHENS

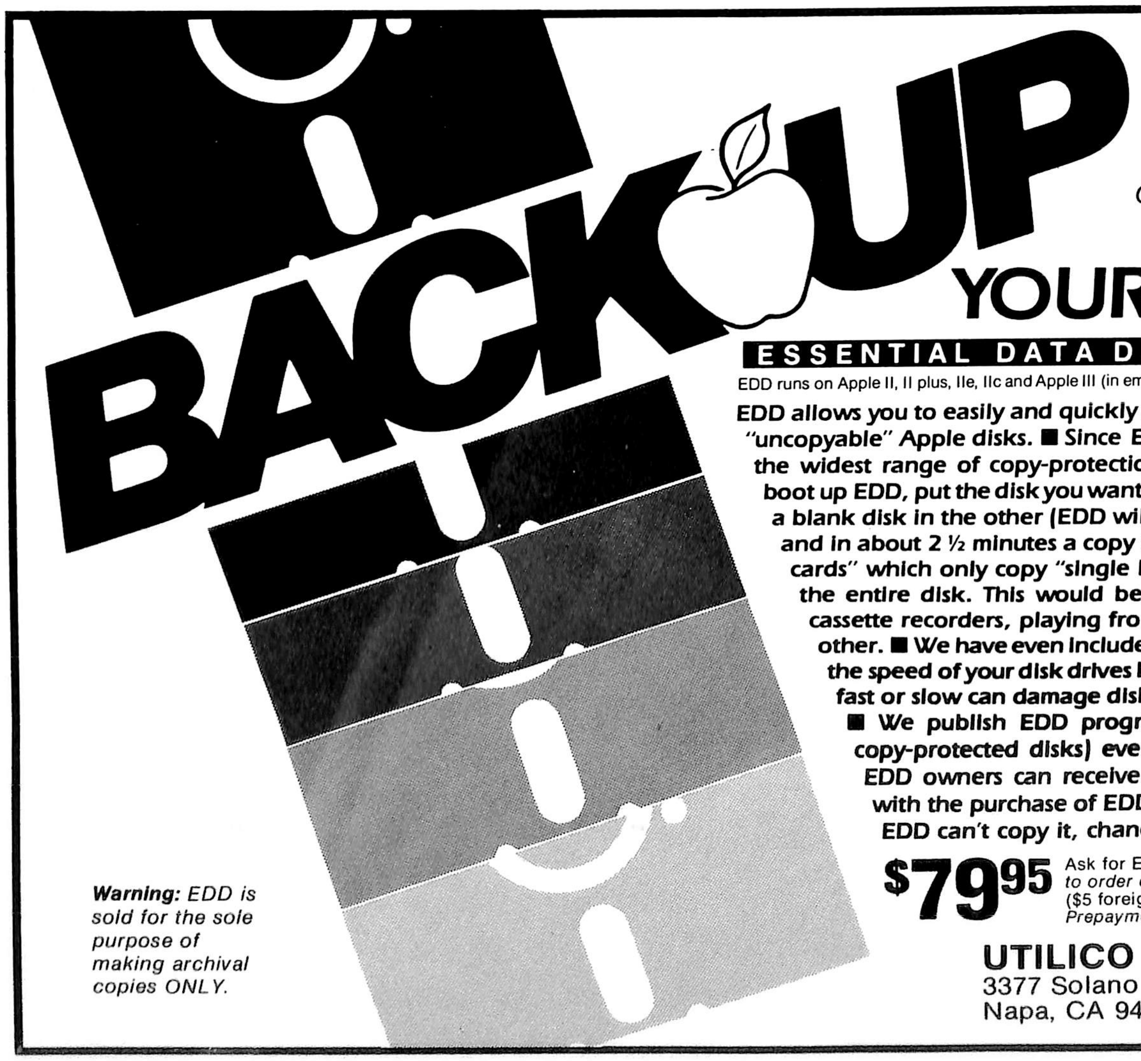
DISK EDITING SYSTEM
FOR THE APPLE II FAMILY OF COMPUTERS
USING DOS 3.3 OR DOS 3.3 WITH
PRONTO-DOS ENHANCEMENT

● 4 Editors ● 13 Sub-editors ● Completely menu driven ● 1 or 2 drive operation ● INSTRUCTORIALS and TECHNICAL INFORMATION on disk ● Sort catalog entries ● Hide catalog entries ● Create multiple heading catalogs ● Create multi-column catalogs ● Reassign lock/unlock and file type symbols, spaces, columns, catalog heading and program titles to any combination of NORMAL, FLASH, INVERSE, LOWER CASE or CONTROL characters ● Rename all DOS commands ● Rewrite all DOS error messages ● Make a (non-VTOC dependent) disk map showing the TRACK/SECTOR LISTS, VTOC and CATALOG TRACK/SECTORS ● Trace out individual files ● Change the HELLO program ● Change any byte anywhere on a disk ● Make instructions, disk maps, track/sector displays and traces using your printer

AND MUCH MUCH MORE

SO WHAT SOFTWARE
10221 Slater Ave. Suite 103 Fountain Valley, Ca. 92708

TO ORDER: Send a check or money order (U.S. funds) for $29.95 plus $2.00 shipping ($5.00 foreign) California residents add 6%.



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

80¢ per disk?

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

SS/DD 5 1/4" for use with:
Apple
Commodore
Atari
Epson
TRS-80

25 Verbatim disks For only $20.00

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

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

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

Name ____________________
Address ____________________
City __________ St ______ Zip ______
Country ____________________
VISA/MC____________Exp______
Signature ____________________

Moving Soon?

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

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

My new address is:

Address ____________________
City __________ St ______ Zip ______
Phone ____________________

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

Name ____________________
Address ____________________
City __________ St ______ Zip ______
Country ____________________
Phone ____________________

Public Domain Software
SPECIAL
From The
COMPUTER LEARNING CENTER!

GET Over 170 BUSINESS & FINANCE Programs FOR ONLY $28.00

Forecast your investment profits
Evaluate the worth of your stock portfolio
Experiment with Visicalc formulas
Design a business or personal letter writing format
Organize your household expenses
Balance your checkbook
Manage your loans and payroll

The Business & Finance SPECIAL contains the following programs and many more...

Keogh Savings Program	Household Exp. Profile
Mortgage Calculation	Income Tax 1040 For 77
Regular Deposits I	Check Book Balancer
Stocks	Address File
Calendar Personal	Names Search
Letter Writer	Gen Ledger Printer
Phone List	Constar
Visicalc Formulas	etc...

ORDER NOW!

A complete catalog of Public Domain Software available from the CLC will arrive with your order.

☐ Yes! Please send me the Business & Finance Special Offer (Volumes 18-25 from the CLC Public Domain Library). I understand that Public Domain Software is not commercial quality and is supplied as-is and that orders are filled on double sided disks.

Name ____________________ S-4
Address ____________________
City __________ St ______ Zip ______
Country __________ Phone __________
VISA/MC____________Exp______
Signature ____________________

Send check or money order (US funds drawn on US bank) to: Computer Learning Center, PO Box 110876-A, Tacoma, WA 98411. Washington residents add 7.8% sales tax. Foreign orders add 20% shipping and handling.

(The Computer Learning Center's PDS Library contains over 170 volumes, all which are formatted for Apple][Plus computers and Apple compatibles. Most will also run on Apple //e and //c.)

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

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

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

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

PR#6

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

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

6$\boxed{\text{CTRL P}}$

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 (Synergistics) sells the instructions necessary to make this modification. Making this modification to an Apple //c will void its warranty but the increased ability to remove copy protection may justify it.

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

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

An illustration- If you strike these keys:

10 HOME:REMCLEAR SCREEN

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

10 HOME : REM CLEAR SCREEN

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

10 DATA 67,45,54,52

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

10 DATA 67,45,54,52

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

All of this brings us to the Hardcore COMPUTIST LISTing format. In a BASIC LISTing, there are two types of spaces; spaces that don't matter whether they are keyed or not and spaces that must be keyed. Spaces that must be keyed in are printed as delta characters (▵). 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 $\boxed{\text{CTRL Y}}$.

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

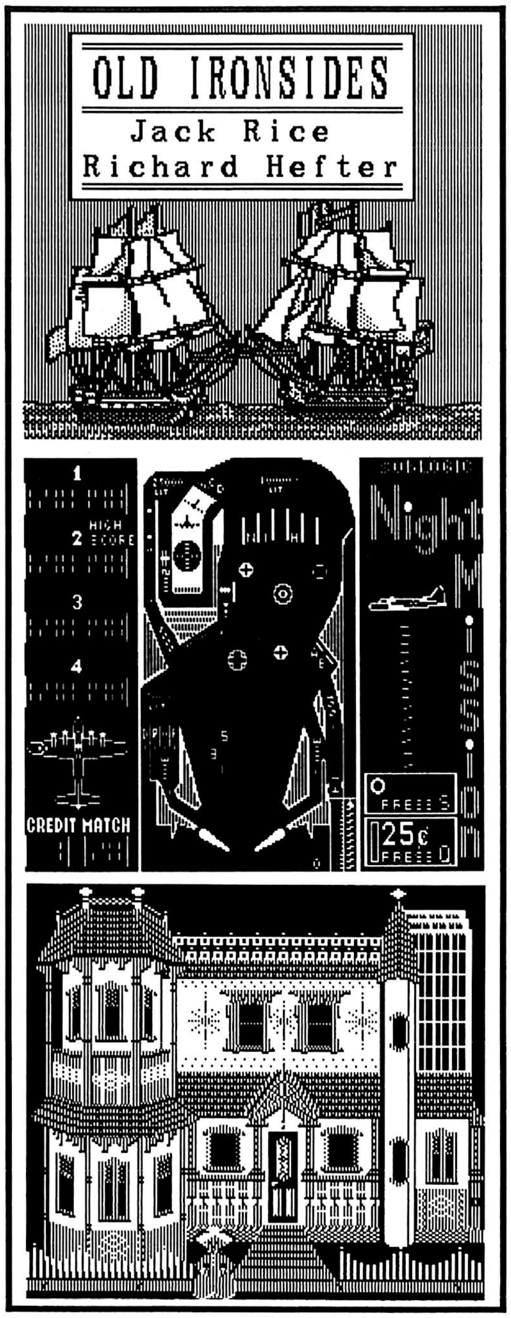
And correct the lines at which the checksums differ.

How-To's Of Hardcore

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

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

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



This month's cover: *Graphics from Sublogic's A2-PB1 Night Mission Pinball.*

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 $34, Mexico $39, Foreign $60. Direct inquiries to: **Hardcore COMPUTIST, Subscription Department, PO Box 110846-T, Tacoma, WA 98411.** Please include address label with correspondence.
DOMESTIC DEALER RATES: Call (206) 474-5750 for more information.
Change Of Address: Please allow 4 weeks for change of address to take effect. On postal form 3576 supply your new address and your most recent address label. Issues missed due to non-receipt of change of address may be acquired at the regular back issue rate.

Hardcore Issue 22

COMPUTIST

Publisher/Editor: Charles R. Haight **Technical Editor:** Ray Darrah
Production & Graphics: Lynn Campos-Johnson, Bev Haight **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

softkeys

10 Miner 2049er
Hardcore COMPUTIST wrestles this classic arcade and achieves a victory in the form of a deprotected diskette. *by Tom Phelps*

11 Lode Runner
Finally, you can have a COPYable version of this most-liked game. Follow this article and learn Lode Runner's secrets *by Tom Phelps*

13 A2-PB1 Pinball
The pinball machine has come a long way since its invention. Now it's your turn to help in pinball's evolution by deprotecting this excellent representation. *by Joseph W. Leathlean*

features

15 Customized Drive Speed Control
If you're tired of removing the disk drive cover everytime you want to adjust your drive speed, you can use this article to replace your speed control with a hand (and much more precise) knob. *By William Wingfield, Jr.*

17 Super I O B version 1.5
The official Hardcore COMPUTIST deprotection program has updated after seven long months of sameness. This version of Super IOB can deprotect disks 350‡ faster than previous versions of Super IOB. *by Ray Darrah*

core

25 The Macro System
Have you ever wanted to have the capability to type a long sequence of characters by only pressing a few keystrokes? The possibilities of this program which allows you to do just that are indeed great. *By Tim Lewis*

departments

4 Input
Backing up MECC software, Deprotecting Storymaker, PFS softkey correction, Modified PROMS on the //e, SSI Deprotection help, Space invaders on Skyfox, Bootlegger problems.

6 Readers' Softkey & Copy Exchange
Bill Bood softkeys Microlab's **The Heist.** Bryce Swimley softkeys Xerox Educational Software's **Old Ironsides.** Phil Pattengale softkeys Spinnaker's **Grandma's House,** and **In Search Of The Most Amazing Thing.** William Wingfield, Jr. softkeys EPYX's **Morlock Tower.** Denny Colt softkeys Sierra Online's **Marauder.** Ray Darrah softkeys **Sargon III**'s other protection scheme

9 ADVENTURE TIPS
24 A.P.T.s

input

Backing Up MECC Software

As a teacher, I refuse to go into a classroom without having three copies of the program I'm going to use: one to use, one in the drawer, and the original far, far away. Please try to publish more softkeys and backup information for educational material (software like the Print Shop). Most of the publishers of educational software don't seem to realize that students can unintentionally kill a disk in an infinite number of ways.

Now, let me climb down off my soapbox and offer some practical advice for other teachers. The Minnesota Educational Computing Corporation (MECC) offers a large variety of good educational material, most of which is written in BASIC making it easy to modify. Unfortunately, most of MECC's disks are protected. Fortunately, they are easy to unprotect using Super IOB with the standard Swap Controller. The only protection that MECC uses is an altered RWTS and I used the material in Ken Greenlaw's article, "Secret Weapon: RAMcard (Hardcore COMPUTIST No. 16) to capture it, although there have been several other methods published in Hardcore.

After you have removed the protection, two modifications I recommend are to put a faster DOS on the disks and rewrite the HELLO programs to eliminate the MECC hi-res "title page". Waiting for that map gets boring after awhile.

One question: Does anyone know who manufactures the Disk Muncher copy program? Everyone I know has a copy (in dozens of versions) but nobody bought it and I've never seen it advertised! It is the fastest general purpose copy program I have ever seen. I would like to use it in class to make backups of student program disks, but I don't want to step on someone's copyright.

P.S. Hardcore is the most useful computer magazine around. Keep up the good work!

Tim Stahmer

We at Hardcore COMPUTIST have also seen several versions of Disk Muncher (one is even marketed by ABACUS Software). We also have no idea where it originated.

Deprotecting Storymaker

After long hours of trying to locate the nibble count on Story Machine (Spinnaker), I decided to give the program to my friend and let him have a go at it. To my surprise, he gave me three different methods of removing the copy-protection. The requirements and the step-by-step procedure to remove the protection are listed below:

Requirements:
Apple or compatible with 48K
One disk drive
COPYA
A sector editor program
One blank disk

You must first copy the program using COPYA. Then you can proceed in one of three ways. Each method involves either removing a command that sets a byte to a certain value or putting in a command that sets a byte to a correct value.

Method 1:
On track Ø, sector 3 change bytes 64 and 65 from A5 1E to A9 FØ.

Method 2:
Change track Ø, sector 3 byte 71 from Ø2 to ØØ.

Method 3:
Change track Ø, sector 3 bytes 72 and 73 from 69 FF to A9 ØØ.

I have tried each of these methods separtely and they all work equally well.

The Programmer

PFS Softkey Correction

Re: Deprotecting PFS Software, Reader's Softkey Exchange (Issue No. 14). I have a small modification to make.

In the original article, the reader was instructed to look for:

DØ 04 88 98 FØ 27

Instead, the search should be for:

88 98 FØ 27

because the actual routine is as follows:

DØ xx 88 98 FØ 27

where "xx" can be almost any number from $04 to $1Ø, and the change should be to the address of:

A9 FF CD

which can be from $27 to $35 bytes away from the $DØ.

I have found the new Copy II Plus v5.2 very handy for this kind of work. The sector editor has a search feature in its disassembly list feature. This is very useful for software that uses a lot of relative jumps. For the absolute addressing problems, I am still stuck with using the Integer ROM card or an Interrupt card to help find things.

George Mochizuki
Madera, CA

Modified PROMs on the //e

I read with interest the article by Ernie Young (Issue No. 6) on modifying PROMS for the Apple computer but resisted making the change because of my reluctance to work inside my machine, an Apple //e. After some hand wringing I decided to dig in and give it a try. To my surprise, the //e does not have an F8-ROM. It instead has the monitor code in the EF-ROM which is the 2364 PROM, not a 2316. The 2364 PROM's are directly pin compatible with the 2764 EPROM's. Thus, all I had to do was burn the modification into a 2764 and pop it directly into the //e. Everything, including the changes suggested by Mr. Mindrup (Issue No. 9), worked without any wiring changes.

Similarly, I was intrigued by the articles by Ray Darrah and Earl Taylor on doubling the ROM space and an improved monitor routine (Issue No. 19). On checking, I found that the 2364 PROM and the 27128 EPROM are also pin compatible. However, the pin positions for the switch connections on the 27128 EPROM are different: ground is pin 14, pin 28 is +5v, and pin 26 is the most significant address bit which would be used to select the appropriate bank in the 27128 EPROM. Due to my limite experience with a soldering iron I decided t make the switch connection to the appropriat pins of a low profile DIP socket instead o directly to the EPROM. This avoids potential heat damage to the EPROM and allows for changing or reprogramming the EPROM without re-soldeing the switch.

In both of these cases, I have not done any modifications to the machine (no voltage inversion, etc.). It all seems fine to me.

Can anyone tell me if I've overlooked some point that could give me problems down the road? Also, the "How-To" page in your magazine mentions a modified CD-ROM available from Synergistics as a means to RESET to the monitor. Can anyone tell me why this is needed or what it does?

Even though Mr. Taylor warned that his modifications were specifically for the F8-ROM, I tried to implant them in the EF-ROM. Needless to say, I encountered several problems. The binary search routine overwrites code at $FBB4 which serves as a conduit to display and escape sequence routines on the 8Ø-column card. Without this code the machine dies. Since this code is referenced from a number of locations, it's not a good idea to change its position. Whether the lack of support for the arrow keys in Taylor's approach is of major or minor importance probably depends on whether the user has an Apple][,][+ or //e.

I was able to patch Mr. Taylor's code into the EF-ROM and retain all features except for the blinking cursor and the NMI code. The only way I could see to implant the blinking underline cursor was to alter the PROM on the 8Ø-column card. Since I wanted to use the second cursor to indicate which version of the monitor was active, making an "always active" change to the PROM did not seem beneficial. Because I don't know how to initiate an NMI and typically want my machine to boot a disk on powerup, I deleted the NMI code and changed the NMI and reset vectors ($FFFA-$FFFD) to mimic the code by Mr. Mindrup (i.e. 1. Boot a disk on powerup, 2. On RESET followed by a return, use the vector at $Ø3F2. I find a reset to the intended program vector to be useful in boot tracing some types of programs.)

Here are the steps for adding Mr. Taylor's modifications to the EF-ROM of the Apple //e:

1. Load a copy of the Apple //e monitor into RAM

```
1ØØØ<EØØØ.FFFFM
```

2. Alter RESET routine to go to NEWRESET thru the old NMI vector. Kill jump to the print title code

```
2AA3: 6C FA FF EA EA EA
```

3. Part 2 of the routine to set up search display

```
2B6Ø: A5 3C E9 ØF 85 3A A5 3D
2B68: E9 ØØ 85 3B 4C 84 FE
```

4. Jump to check for "esc H'

```
2B98: 4C ED FE
```

5. NEWRESET- same as Taylor's except uses vector at $Ø3F2

```
2CC9: AD ØØ CØ 1Ø FB 2C 1Ø
2CDØ: CØ 29 DF C9 D3 FØ Ø7 C9
2CD8: CD FØ 2D 6C F2 Ø3 BA 8E
2CEØ: Ø3 29 AØ ØØ B9 ØØ ØØ 99
2CE8: ØØ 2Ø B9 ØØ Ø1 99 ØØ 21
2CFØ: C8 DØ F1 84 3C 84 3E 84
2CF8: 42 A9 Ø2 85 3D A9 Ø9 85
2DØØ: 3F A9 22 85 43 2Ø 2C FE
2DØ8: 4C 59 FF EA
```

6. Taylor's MOD.NOTCR

```
2D58: EØ F8 9Ø Ø3 2Ø 3A FF E8
```

7. Taylor's STOP PATCH

```
2D7E: 4C 84 FD E8 A9 88
```

8. Taylor's LIST2 MOD

```
2E74: AA
```

9. Misc. Code, Disassembler Mod, Window set & vector to SEARCH, Tape error, Test for "esc H" and CRMON

```
2EC2: 6Ø EA EA 4C F8 Ø3
2EC8: 2Ø 5E FE A9 Ø1 2Ø 63 FE
2EDØ: AD ØØ CØ 1Ø FB 2C 1Ø CØ
2ED8: C9 95 FØ EC C9 AØ FØ EB
2EEØ: C9 9B 6Ø A9 Ø2 85 22 4C
2EE8: FD FE 4C 2D FF C9 C8 DØ
2EFØ: Ø2 A9 CØ 4C 2C FC 2Ø ØØ
2EF8: FE 68 68 DØ 6C
```

10. Taylor's SEARCH and Part 1 of display setup for SEARCH

```
2EFD: AØ ØØ B9
2FØØ: Ø2 Ø2 CD ØØ Ø2 FØ Ø4 D1
2FØ8: 3C DØ 13 C8 CC Ø1 Ø2 9Ø
2F1Ø: EE 2Ø 8Ø FE 2Ø 26 FF A2
2F18: ØØ 2Ø C8 FE FØ Ø5 2Ø BA
2F2Ø: FC 9Ø DA 4C 2F FB 2Ø 92
2F28: FD 4C 6Ø FB EA
```

11. Modification to Monitor's CHRTBL so ctrl Y vector will function

```
2FCF: ED
2FD2: EC
```

12. Modification to Monitor's SUBTBL to set vectors for ctrl Y, T, S, L, W and R commands

```
2FE4: C4
2FE6: E1
2FE9: E2
2FF2: C7 E9
2FF5: E9
```

13. Modification of NMI and RESET vectors

```
2FFA: C9 FC 62 FA
BSAVE MODROM,A$1ØØØ,L$2ØØØ
```

14. Burn file into a 2764 EPROM.

I really appreciate the series of articles on modified PROMS, both for what I learned directly from the articles and for all the additional things I've learned in trying to understand and accomplish the tasks set forth.

If someone is in the market for an EPROM burner I'd recommend the PROMgRAMMER ($149.5Ø) by the Southern California Research Group, especially if you need to program some of the larger chips. The burner handles 2716 through 27512 EPROMS, it has a fast burn routine and the software is great. It is, however, an internal card system.

Donald L. Berry
Midland, MI

SSI Deprotection

I have noted that the SSI Wargame Series has been on the Most Wanted List for some time. If you haven't received a crack by this time, I have just the thing for you. It's two machine language programs, RDOS BUSTER and RDOS DISK CREATOR, written by the staff at Everyware, that will convert all RDOS-based disks to a COPYAble format.

Richard C. Colby
President
Everyware Computer Prod.
Riverdale, MD

Space Invaders on Skyfox

This may interest you: I have found a trap door (or whatever you call it) in the game Skyfox.

When you get to the tactical display, right after choosing your level, press ctrl G and you will be put into a Space Invaders-type game.

Jon Camp

Bootlegger Problems

In a previous letter I wrote to explain a problem I am having with my subscription to Bootlegger. In reply to that letter, you wrote, after talking to the appropriate person at Bootlegger, that my money would soon be refunded. Seven weeks later, the Bootlegger *HAS NOT* refunded the $18.75 for the 3 issues I have not received. As of today, 5 issues of Bootlegger have been issued, But I have recieved only one of these, No. 3.

I don't know what their problem is, but they have evidently ripped me off and lied to you about refunding my money. I have subsequently lodged a complaint with the Postal Inspector's office. Eric Larson, Regional Chief Postal Inspector, San Bueno, CA wrote to Bootlegger on April 4 but, so far, Mike Betitec has not responded to either his letter or mine. At this point I feel that I have no other choice but to request that the Postal Inspector turn the matter over to the US Attorney for prosecution for mail fraud.

Munson Compton
Shreveport, LA

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.

Softkey & Copy Exchange

Bryce Swimley's Softkey For...

Old Ironsides

Xerox Educational Software
245 Long Hill Road
Middletown, CT Ø6457
(2Ø3) 347-7251
$39.95

Requirements:
Apple][Plus, //e or //c
Means of entering the monitor at will
COPYA or copy program which ignores errors
A blank disk
A disk with no HELLO program

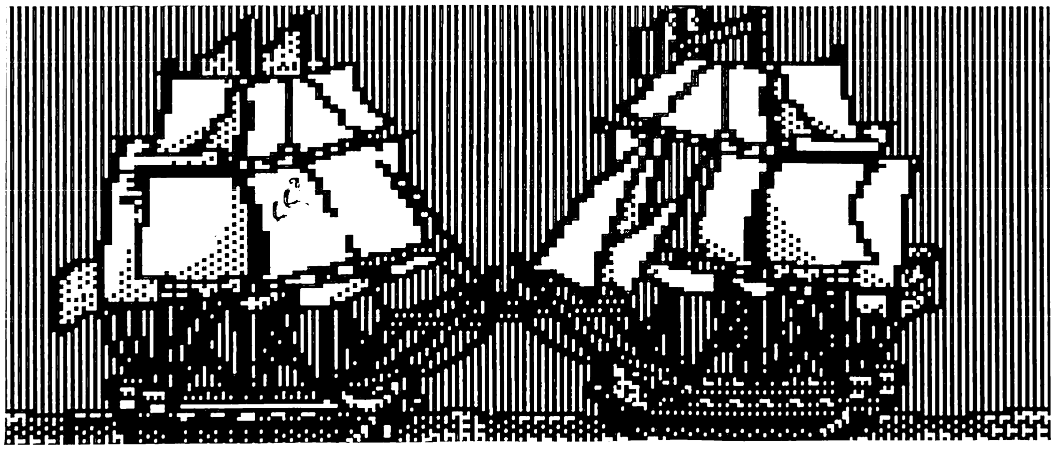
In my efforts to use Mr. Jerry Caldwell's "Stickybear Bop series" softkey (Hardcore COMPUTIST No. 15) to make a copy of Old Ironsides, a program also published by Xerox, I ran into a few problems. It soon became apparent that the code to read the protected sector (Track Ø1 Sector ØF on my copy of Old Ironsides) was located in many places on the disk. I found the code on Track Ø3, Sector Ø9; Track Ø4, Sector Ø6; Track Ø9, Sector Ø5; and also on Track ØB, Sectors ØF.

Following Mr. Caldwell's procedure through, I discovered that the protected sector was loaded into $BØØ. I recovered the data by using the "Extended RamCard" technique in Issue No. 16. This worked admirably. I then proceeded to change the load accumulator command (A2) to a RTS (6Ø) on all the locations where I found the code to read the protected sector. Then I wrote the data I had recovered to my copy and eagerly booted the unprotected copy only to discover that the program rebooted when it tried to load the now deprotected sector.

This made me more than a little curious as well as frustrated. I then unleashed my trusty sector editor and scanned the disk for a jump to the disk controller ROM (4C ØØ C6). To my pleasant surprise, I found these jumps at two locations on the disk- Track ØB, Sector ØF and Track 1A, Sector Ø4. Disassembling the code near those jumps, I found "Branch Not Equals" (BNE's) to the location which contained the JMP instruction. I changed these BNE's to NOP's (EA) and wrote the sectors back to my copy. Now when I boot the disk, as far as I can tell, it works perfectly! If you haven't been able to softkey a "Xerox" disk using the original method by Jerry Caldwell, you might want to try these additional procedures.

Step-By-Step

Follow this procedure to deprotect Old Ironsides:



1) Load COPYA and stop it while it is asking you for the slot number

LOAD COPYA
CTRL C

2) Enter the monitor and tell the machine language portion of COPYA to ignore errors

CALL -151
3A1:EA

3) Re-enter BASIC, tell COPYA not to reload its machine language portion and make a copy of your Old Ironsides disk

EØØ3G
7Ø
RUN

4) Boot your protected copy of Old Ironsides and reset into the monitor as soon as the title screen appears

5) Insert a disk with no HELLO program and boot it

C6ØØG

6) Save the protected sector

BSAVE PROTECTED SECTOR, A$AØ4, L$1FC

7) Get out your sector editor and make the following changes to the copied Old Ironsides disk:

Track	Sector	Byte	Change to
$Ø3	$Ø9	$29	$6Ø
$Ø4	$Ø6	$29	$6Ø
$Ø9	$Ø5	$25	$6Ø
$ØB	$ØF	$FD	$6Ø
$1A	$Ø4	$54	$ØØ
$ØB	$ØF	$C9	$ØØ

8) Now you must find the location (on your disk with no HELLO program) at which the protected sector is stored. To do this, use your sector editor to scan the CATALOG track ($11) for the PROTECTED SECTOR filename. When it is found, the three bytes directly before the filename indicate the track & sector, respectively, and filetype of the file.

9) Read in the track and sector indicated by the first two of the three bytes directly before the filename.

1Ø) Next, read in the track and sector indicated by bytes $ØE and $ØF of this sector.

11) Put in your copy of Old Ironsides and write this sector to track $Ø1, sector $ØF.

12) Enjoy your deprotected Old Ironsides!

Bill Bood's Softkey For...

The Heist

Microlab, Inc
2699 Skokie Valley Road
Highland Park, IL 6ØØ35
(312) 433-755Ø

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

The Heist is a low action game program in which you must try to steal as much loot from a building as possible. While performing the heist, you must avoid a wide array of obstacles ranging from burglar detection robots to walls that fall from the ceiling.

Fortunately, Microlab didn't include much protection on this disk- they only did a nibble count and changed the end of address marker from $DE AA to $9E E7 and the end of data marker from $DE AA to $D5 AA. This kind of protection is easily defeated with a little help from Super IOB.

To make a COPYAable copy of The Heist

install the controller, printed at the end of this article, into Super IOB and RUN it.

Sector Edits Performed By The Controller

When Super IOB is finished with the disk, the following sector edits have been performed so that the new Heist disk will expect normal sector markers and not bother with a nibble count.

Track	Sector	Byte	From	To
$00	$02	$9E	$D5	$DE
$00	$03	$35	$D5	$DE
$00	$03	$9B	$E7	$AA
$00	$03	$91	$9E	$DE
$00	$08	$38	$4C	$08
$00	$08	$39	$6A	$B0
$00	$08	$3A	$BA	$8E

A Few APT's

I have discovered several locations in memory that hold vital information about the game. To perform any of these APT's you must have the means to enter the monitor at will. Once in the monitor during a Heist game, perform the APT you want and then get the game going again with:

A05G *(restart game)*

To change the number of men you start with, change the byte at $0F92 to:

F92:xx *(xx=number of men)*

To give yourself an amount of keys to start out with, erase the zeroing subroutine and put the amount of keys you want in location $948 as follows:

F88:EA EA EA
948:xx *(xx=number of keys)*

To start at any level, NOP the zeroing routine, NOP the routine which inserts the real level and put your starting level at $945:

F8B:EA EA EA
F9D:EA EA EA
945:xx *(xx=level to start at)*

The high scores are stored on track $01, sector $04 in the following format:

II II II 04 SS SS SS FF

The II's are the high-bit clear ASCII equivalent of the initials, the $04 separates the initials from the Score, the SS's are the Binary coded decimal equivalent of the score and the $FF separates entries.

Heist Controller

```
1000 REM HEIST CONTROLLER
1010 TK = 0 :ST = 0 :LT = 35 :CD = WR
1020 T1 = TK : GOSUB 490 : RESTORE : GOSUB 170
1030 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1030
1040 IF BF THEN 1060
1050 ST = 0 :TK = TK + 1 : IF TK < LT THEN 1030
1060 GOSUB 310 : GOSUB 490 :TK = T1 :ST = 0 : GOSUB
     230
1070 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1070
1080 ST = 0 :TK = TK + 1 : IF BF = 0 AND TK < LT THEN
     1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "DONE^WITH^COPY' : END
5000 DATA 158 ,231 ,213 ,170 : REM EOA , EOD
5010 DATA 7^CHANGES
5020 DATA 0 ,2 ,158 ,222
5030 DATA 0 ,3 ,53 ,222
5040 DATA 0 ,3 ,145 ,222
5050 DATA 0 ,8 ,56 ,8
5060 DATA 0 ,8 ,57 ,176
5070 DATA 0 ,8 ,58 ,142
5080 DATA 0 ,3 ,155 ,170
```

Heist Controller Checksums

```
1000  - $356B     1100  - $360A
1010  - $3266     5000  - $9D88
1020  - $7F0A     5010  - $34C5
1030  - $690B     5020  - $F851
1040  - $51E4     5030  - $9026
1050  - $FDC1     5040  - $26B9
1060  - $0BDF     5050  - $A575
1070  - $03DA     5060  - $2C21
1080  - $73B7     5070  - $C85F
1090  - $E201     5080  - $9DEA
```

Phil Pattengale's Softkey For...

In Search of the Most Amazing Thing Grandma's House

Spinnaker Software
One Kendall Square
Cambridge, MA 02139
$39.95

Requirements:
Apple][Plus or //e
1 or 2 drives
In Search of the Most Amazing Thing
or Grandma's House
Super IOB v1.5

These two children's programs by Spinnaker are both delightful and intriguing for the younger child; however, they are copy-protected.

Fortunately, they both use a semi-standard DOS and it is just this scratch in the armor that allows us to deprotect them both.

The protection scheme on In Search of the Most Amazing Thing involves changing the sector address markers from $D5 AA 96 DE AA to $AA D5 AB DE AB and changing the sector data markers from $D5 AA AD DE AA to $AA D5 EB ED AA. The protection on Grandma's House is a change of sector data ending marks from $DE AA to $BF AA.

But not to fear: Hardcore COMPUTIST readers are equipped with Super IOB v1.5 which can easily deprotect this sort of scheming.

Step-By-Step

1) Format a blank disk (to be used as the copy disk) and put a fast (preferably) DOS on it with "HELLO" as the boot file

INIT HELLO

2) Install the proper controller into Super IOB v1.5
3) RUN Super IOB and answer the questions. That's it! Pretty easy.

In Search of... Controller

```
1000 REM IN SEARCH OF . . .
1010 TK = 3 :LT = 35 :ST = 15 :LS = 15 :CD = WR :FAST
     = 1 : POKE 48573 ,128
1020 RESTORE : GOSUB 190 : GOSUB 210 : GOSUB 170
     : GOSUB 490 : GOSUB 610
```

```
1Ø3Ø GOSUB 23Ø : GOSUB 49Ø : GOSUB 61Ø : IF PEEK
     (TRK ) = LT THEN 1Ø5Ø
1Ø4Ø TK = PEEK (TRK ) :ST = PEEK (SCT ) : GOTO 1Ø2Ø
1Ø5Ø HOME : PRINT "COPYDONE" : END
5ØØØ DATA 17Ø ,213 ,171
5Ø1Ø DATA 17Ø ,213 ,235
5Ø2Ø DATA 222 ,171 ,237 ,17Ø
```

In Search...Controller Checksums

1ØØØ	- $356B	1Ø5Ø	- $964E
1Ø1Ø	- $9E6E	5ØØØ	- $7Ø43
1Ø2Ø	- $1E94	5Ø1Ø	- $82AD
1Ø3Ø	- $2BØD	5Ø2Ø	- $BØFA
1Ø4Ø	- $48DD		

Grandma's House Controller

```
1ØØØ REM GRANDMAS HOUSE
1Ø1Ø TK = 3 : LT = 35 : ST = 15 : LS = 15 : CD = WR : FAST
     = 1
1Ø2Ø GOSUB 17Ø : RESTORE : GOSUB 49Ø : GOSUB 61Ø
1Ø3Ø GOSUB 23Ø : GOSUB 49Ø : GOSUB 61Ø : IF PEEK
     (TRK ) = LT THEN 1Ø5Ø
1Ø4Ø TK = PEEK (TRK ) :ST = PEEK (SCT ) : GOTO 1Ø2Ø
1Ø5Ø HOME : PRINT "COPYDONE" : END
5ØØØ DATA 222 ,17Ø ,191 ,17Ø
```

Grandma's Controller Checksums

1ØØØ	- $356B	1Ø4Ø	- $D1F1
1Ø1Ø	- $2445	1Ø5Ø	- $88D7
1Ø2Ø	- $5Ø99	5ØØØ	- $B7E6
1Ø3Ø	- $65ØØ		

William Wingfield Jr. Softkeys...

Morloc's Tower

EPYX
1043 Kiel Ct.
Sunnyvale, CA 94089
(4Ø8) 745-Ø7ØØ
$34.95

Requirements:
Apple][,][Plus or //e
1 or 2 disk drives
MUFFIN program from the system master
One blank disk or a disk with at least 77 free sectors

Morloc's Tower is one of the earlier adventures in the Dungeonquest series and consists of two Applesoft files and one binary file which use a total of 77 disk sectors. Since it is a good idea to make a backup of all programs, and this one is relatively short, we'll make a standardized copy that doesn't require a complete disk.

An inspection of the disk with a nibble editor shows that it is a standard DOS 3.2 disk except for the last bit of the data prologue. To protect it, the standard D5 AA AD data marker has been replaced the AD with other bytes on some tracks.

My philosophy is to do as little work as possible to unlock a protected program, and although boot code tracing and RWTS stuff have their uses, all we need to do is to make a small change to the Muffin program and let *it* do all the dirty work.

Step-By-Step

1) Boot the system master disk and load Muffin

BLOAD MUFFIN

2) If you have a regular DOS 3.3 disk with at least 77 free sectors, insert it now and skip to Step 4.
3) Insert your blank disk and format it

INIT HELLO

4) Get into the monitor

CALL -151

5) Modify DOS so that it ignores the last byte of the data trailer

A1C:29 ØØ

6) Start up MUFFIN

8Ø3G

7) Use "=" for the filename and transfer the files from your Morloc's Tower disk to your DOS 3.3 disk. The first file MUFFIN will come to is HELLO. You may wish to rename this file as M.T. HELLO just to distinguish it as part of Morloc's Tower.

You now have a fully functional and standard copy. Enjoy!

Denny Colt's Softkey for...

Marauder

Sierra On-Line
36575 Mudge Ranch Road
Coarsegold, CA 93614
(2Ø9) 683-6858
$29.95

Requirements:
Apple][with 48K
One blank disk
COPYA
Sector editor

Marauder is a two stage game by On-Line systems. In the first stage, your ship must destroy an enemy force field and then the planet's defenses. In the second stage, your disembark from your ship to locate a reactor that you try to stop.

This game, overall, is quite entertaining. But it's the second stage that gets all the applause. It is the best "Berzerk" type scenario I've ever played.

Unfortunately, the amount of disk access between stages makes a backup (as opposed to a file copy) necessary.

The copy protection used on this disk, and on many other On-Line disks, is a standard DOS with a nibble count. To deprotect it, all you must do is defeat the count.

Step-By-Step

1) Copy the disk with COPYA or another whole disk copier.
2) Use a sector editor to read in track $11, sector $Ø7.
3) Change byte $9Ø from $A8 to $6Ø and write it back to the disk (Note: If track $11, sector $Ø7 doesn't work, try track $Ø3, sector $Ø7 or track $11, sector $Ø3. The bytes are always $9Ø from $A8 to $6Ø).

That's it!

Ray Darrah's Softkey for another

Sargon III

Requirements:
A copy program that ignores errors
A Sector Editor

It has come to my attention that the method for deprotecting Sargon III (in Hardcore COMPUTIST No. 2Ø) does not work for some copies of that excellent chess game.

Rather than using altered sector markers (as the copy protection is on the version mentioned in Hardcore COMPUTIST No. 2Ø), the other version of Sargon III is completely COPYAable with the exception of track 3. It turns out that a nibble count is performed on track 3 upon bootup.

The code that does the infamous nibble count is loaded from track $Ø, sector $5 into memory starting at $BBØØ with an entry point of $BBØ3. This code is extremely tricky to trace because it is encoded and decodes itself as it goes.

After deciphering this code (by the way, the code at $BCØØ is encoded in a similar manner), I found that it could be disabled by placing the following routine at $BB6B (track $Ø, sector $5, byte $6B):

```
BB6B-   A9 1Ø       LDA   #$1Ø
BB6D-   8D ØF BC    STA   $BCØF
BB7Ø-   38          SEC
BB71-   E9 1Ø       SBC   #$1Ø
BB73-   8D 39 B7    STA   $B739
BB76-   BD 88 CØ    LDA   $CØ88,X
BB79-   4C 1Ø BC    JMP   $BC1Ø
```

This routine is very close to what the nibble counter does if everthing is okay. By placing it at $BB6B, we exit the routine before it is entered. Of course, since most of this sector is encoded, we will have to encode this patch also. Therefore, the code that translates into this routines is as follows:

```
BB6B- 12 BA 3Ø 26 Ø6
BB7Ø- 9E CE 8D ØØ F9 ØA 31 48
BB78- 48 9C FB 19
```

Step By Step

1) Copy your original Sargon III disk with any whole disk copier (not file copier) and ignore any errors on track $Ø3.
2) Get out your sector editor and starting at byte $6B of track $ØØ, sector $Ø5 of your copy of Sargon III, enter the code listed directly above.

Many thanks to Sheldon Atterbury

Adventure Tips

Colossal Caves

Adventure International

Listen to the bean plant.

Don't give the troll any treasures on the way back- give him a pet. But make sure you can "magically" transport your treasure you gave him (Fie, Fi, Fo...).

Oil does wonders for things that are stuck.

Map out the mazes.

Always carry a weapon.

Infidel

Infocom

Don't step between rocks!

Read and remember all hieroglyphics.

Need some more weight? Use someone's head.

Put boards in the holes in wall to step on.

Plaster might be susceptible to pick-axes.

Leave the crocs alone. Just get water.

Sorcerer

Infocom

Need another Zorknid? Look in your pocket...or maybe a gnome's.

Mines and snakes are hazardous to your health and mind. Don't be bothered by them.

Dorn's can't fly.

Bats have sonar.

Don't sleep in the woods.

Fly over the river bank or you'll fall in.

Pyramid of Doom

Adventure International

Look in all broken glass.

You may be an adventurer, but your hand still needs protection. Use a glove.

Throw the statue's jeweled "heart" into the acid.

Egyptian's aren't dumb! They hide things (in) tables.

Mummies hate incense.

Cranston Manor

Sierra On-Line

Monkeys can swing...so can you. Swing in a closet and the money will come to you.

You should "pry" out the eyes of a cat.

Knight's aren't afraid of lice, but they may be afraid of mice!

Ever tried priming a pump? It's very rewarding.

* *Contributed by Ron Searle.*

Cutthroats

Infocom

You must remember that McGinty can see everything that you're carrying, except money in your pocket.

When McGinty is around, choose your actions carefully. One wrong move may blow the whole game.

Zork II

Infocom

Robots do what you tell them.

The clay is actually plastic explosives.

You must pay the demon quite a bit before you can command him.

Once you have the wand you may use any of the magician's spells.

Escape From Rungistan

Sirius Software

Moving the bed under the window could give you a new perspective on the world.

The rope from the gallows may prove useful.

Run toward the chasm, then jump as you approach the edge.

Saying "Geronimo" would be a good way to begin a downhill slalom.

Space Vikings

Sub-Logic

If you find the prices for the upkeep of a Space Viking's heavy cruiser bothersome, try this:

Begin a new game, then go to the Earth base. Pick up as many troops/weapons as your initial bankroll will allow. Plan it so you have as few credits as possible.

Now repair/restock your ship. When they find out that you have no money, the local government will become angry. From here, all you have to do is return to ship control and fire one laser burst or missile and they will surrender. Now you can go on with your business.

You can only do this one time.

Zenith

If you have trouble figuring out how the map in Zenith works, this should help:

Your ship only travels from bottom to top of the screen. Although it looks as if you are turning, you are actually only moving side to side, and you wrap around all edges.

Contributed by David A. Martin.

ATTENTION ADVENTURERS!

Adventure Data Base

Hardcore COMPUTIST is looking for more adventure hints to any of the popular adventure/fantasy games sold for the Apple II, II Plus or //e. These will be used in our regular column, ADVENTURE TIPS.

Your Clues, Please

We prefer that these hints not be dead giveaway solutions to dilemmas presented by the particular game, but instead contain just enough information to nudge the stumped adventurer towards the solution to his/her problem.

How & Where

So, if you know how to open the jewel-encrusted egg, how to plug the hole in the rowboat, where to find the key to the treasure chest, or any other tidbits of information that may be helpful to your fellow traveler, please send this information on a 3 × 5 postcard to:

Hardcore COMPUTIST
Attention: Adventure Tips
P.O. Box 44549
Tacoma, WA 98444

P.S. Please don't forget to include the name of the adventure game to which your hint pertains and the name of the manufacturer.

And don't forget to send us your helpful

Advanced Playing Techniques, too!

Softkey for Miner 2049'er

By Tom Phelps

Miner 2Ø49'er
Microlab
2699 Skokie Valley Road
Highland Park, IL 6ØØ35
(312) 433-755Ø
$4Ø.ØØ

Requirements:
Apple with 48K
One disk drive
One blank disk
Sector editor such as DiskEdit
Super IOB v1.5

Miner 2Ø49'er is one of Microlab's all-time bestsellers and can consistently be found at the top of the sales charts, even today. This is more a thinking man's game than one of manual dexterity; if you plan your jumps right, you need not be a master of joystick control to complete each level.

Although the game itself is very entertaining, trying to copy the disk isn't: it takes a considerable amount of time to copy the nibble count on track zero with most major nibble copiers.

Fortunately, we have Super IOB and, with a little investigation, we find that the Miner RWTS is normal enough to allow most of the disk to be copied with the Swap Controller. Though the Swap Controller is needed on tracks $1-$22, track Ø can be copied normally with any copier. Some sector edits must be performed, in addition, to defeat the nibble count on track Ø. To eliminate unnecessary steps, we will combine the copying of the normal track and the defeating of the nibble count with the use of the Swap Controller to produce a unique controller that will copy the entire Miner 2Ø49'er disk.

Here are the steps needed to crack Miner 2Ø49'er:

Step-By-Step

1) First we need to capture Miner's RWTS. Fortunately, for those who don't have or have access to a ROM that enables you to enter the monitor when RESET is pressed, Miner allows us to do this without one. The first step to obtaining the RWTS is to insert the Miner 2Ø49'er disk in drive 1 and boot it.

PR#6

2) Now, when the title page comes up, press any key. Press RESET when asked which joystick you wish to use. Now enter the monitor (indicated by the "*" prompt).

CALL -151

3) Move the RWTS into a safe location for a DOS 3.3 reboot.

19ØØ<B8ØØ.BFFFM

4) Boot a standard 48K DOS 3.3 disk.

C6ØØG

5) Save the RWTS on the Super IOB disk.

BSAVE MINER 2Ø49'ER.RWTS, A$19ØØ, L$8ØØ

6) Install the Miner 2Ø49'er controller listed at the end of this article into Super IOB and run the resulting program. When asked whether to format the disk or not, answer "Y" for yes.

The Sector Edits

When Super IOB is finished deprotecting your disk it will have performed the following sector edits:

Track	Sector	Byte	From	To
$ØØ	$Ø3	$35	$96	$DE
$ØØ	$Ø3	$3E	$A9	$C9
$ØØ	$Ø3	$3F	$ØØ	$AA
$ØØ	$Ø3	$55	$D3	$D5
$ØØ	$Ø3	$5F	$96	$AA
$ØØ	$Ø3	$6A	$F2	$96
$ØØ	$Ø3	$91	$D3	$DE
$ØØ	$Ø3	$9B	$B2	$AA
$ØØ	$Ø2	$53	$96	$D5
$ØØ	$Ø2	$58	$D3	$AA
$ØØ	$Ø2	$5D	$E5	$AD
$ØØ	$Ø2	$9E	$96	$DE
$ØØ	$Ø2	$A3	$FF	$AA
$ØØ	$Ø2	$E7	$96	$D5
$ØØ	$Ø2	$F1	$D3	$AA
$ØØ	$Ø2	$FC	$E5	$AD
$Ø1	$Ø1	$81	$DØ	$EA
$Ø1	$Ø1	$82	$ØC	$EA
$Ø1	$Ø1	$89	$DØ	$EA
$Ø1	$Ø1	$8A	$Ø4	$EA
$Ø1	$Ø1	$92	$BE	$C5
$Ø1	$Ø1	$94	$BE	$C5

The first block of sector edits tells the Miner RWTS to expect normal address markers, the next tells the Miner RWTS to expect normal data markers, and the last disables the nibble count routine.

Advanced Playing Techniques

For you game buffs, here is a bit of information you might consider using. The high scores in Miner 2Ø49'er are recorded on Track $Ø1, Sector $Ø7, starting with byte $ØØ. You might want to sector edit these, temporarily of course, and impress friends, relatives and neighbors.

Or try the following APT where a sector modification will give you unlimited Bounty Bob's (I suggest that you make a copy of the softkeyed Miner before you make this change so you will have a "normal" copy plus the copy that allows the unlimited number of men). On Track $Ø2, Sector $Ø1, starting with byte $72, overwrite the existing code with A9 Ø3 8D 16 Ø8 8D 17 Ø8 4C 81 Ø9.

You now have a COPYAble Miner 2Ø49'er disk. And, if you did the APT, live long and prosper!

Miner 2Ø49'er Controller

```
1ØØØ REM MINER 2Ø49ER
1Ø1Ø TK = Ø :LT = 1 :ST = 15 :LS = 15 :CD = WR :FAST
      = 1
1Ø2Ø GOSUB 49Ø :T1 = TK : IF TK = Ø THEN GOSUB 61Ø
      :LT = 35 :TK = 1
1Ø25 RESTORE : GOSUB 36Ø : GOSUB 61Ø
1Ø3Ø TK = PEEK (TRK ) : GOSUB 31Ø : GOSUB 36Ø :
      GOSUB 49Ø :TK = T1 : GOSUB 61Ø : IF PEEK (TRK
      ) = LT THEN 1Ø5Ø
1Ø4Ø TK = PEEK (TRK ) :ST = PEEK (SCT ) : GOTO 1Ø2Ø
1Ø5Ø HOME : PRINT "COPYDONE' : END
5ØØØ DATA 22^CHANGES
5Ø1Ø DATA Ø, 3, 53, 222, Ø, 3, 62, 2Ø1, Ø,
      3, 63, 17Ø
5Ø2Ø DATA Ø, 3, 85, 213, Ø, 3, 95, 17Ø, Ø,
      3, 1Ø6, 15Ø
5Ø3Ø DATA Ø, 3, 145, 222, Ø, 3, 155, 17Ø, Ø,
      2, 83, 213
5Ø4Ø DATA Ø, 2, 88, 17Ø, Ø, 2, 93, 173, Ø,
      2, 158, 222
5Ø5Ø DATA Ø, 2, 163, 17Ø, Ø, 2, 231, 213, Ø,
      2, 241, 17Ø
5Ø6Ø DATA Ø, 2, 252, 173, 1, 1, 129, 234,
      1, 1, 13Ø, 234
5Ø7Ø DATA 1, 1, 137, 234, 1, 1, 138, 234, 1,
      1, 146, 197
5Ø8Ø DATA 1, 1, 148, 197
1ØØ1Ø IF PEEK (64ØØ ) < > 162 THEN PRINT CHR$ (4
      ) "BLOAD^RWTS.MINER,A$19ØØ'
```

Controller Checksums

```
1ØØØ  - $356B      5Ø2Ø  - $2C1D
1Ø1Ø  - $EA41      5Ø3Ø  - $BCAC
1Ø2Ø  - $83CD      5Ø4Ø  - $9E7B
1Ø25  - $AØ6B      5Ø5Ø  - $E27B
1Ø3Ø  - $1E91      5Ø6Ø  - $3682
1Ø4Ø  - $CCD4      5Ø7Ø  - $CBC8
1Ø5Ø  - $97CA      5Ø8Ø  - $6Ø3A
5ØØØ  - $5EF4      1ØØ1Ø - $BBAC
5Ø1Ø  - $FFØ4
```

Softkey For Lode Runner

By Tom Phelps

Lode Runner
Broderbund Software, Inc.
1938 Fourth Street
San Rafael, CA 94901
(415) 456-6424
$34.95

Requirements:
Apple with 48K
One disk drive with DOS 3.3
One blank disk
Modified F8 ROM capable of saving pages $0-$7
Super IOB v1.2
Lode Runner

Lode Runner, another best-selling game from the fun-loving bunch at Broderbund, has recently succeeded Choplifter as everyone's favorite game, not just because everyone was tired of playing Choplifter, but because Lode Runner has to be the ultimate in non-repetitive play. It boasts 150 unique screens plus the ability to create an unlimited number of your own. Unfortunately, Lode Runner also surpasses Choplifter in the complexity of copy-protection.

Whereas you could buy a "copy-card" and "crack" the single-load Choplifter with the press of a button, Lode Runner poses a more difficult challenge because it requires disk access during game play.

Following the unusual boot-up, Lode Runner doesn't check the quarter tracks (tracks protected with spiral protection) and, amazingly enough, uses a fairly normal RWTS to read and write the game boards. We just need to capture the main program from memory and fit the game board tracks (along with the main Lode Runner program) and a DOS all on one disk.

The two parts of the softkey are completely different so to prevent confusion, the procedure below is divided into two sections. If one section doesn't turn out well, the whole softkey need not be redone. You should find the softkey easy to follow and, if you possess the required hardware, you should encounter no difficulty in performing it.

Part One: Grab 150 Game Boards

If you played all 150 levels of Lode Runner and monitored the disk drive read/write head along the way, you would find that only tracks $03-$0C are used to load boards. Fortunately, these tracks are nearly DOS 3.3 formatted and Super IOB need not work overtime. To be certain that our broken main Lode Runner B-file finds the tracks in the same location as they were on the original disk, we will place them on the disk first, not giving the main file a chance to overwrite those tracks. Part One of the softkey is as follows:

Lode Runner, another best-selling game from the fun-loving bunch at Broderbund, has recently succeeded Choplifter as everyone's favorite game, not just because everyone was tired of playing Choplifter, but because Lode Runner has to be the ultimate in non-repetitive play. It boasts 150 unique screens PLUS the ability to create an unlimited number of your own. Unfortunately, Lode Runner also surpasses Choplifter in the complexity of copy-protection.

1) Type in the Super IOB controller listed at the end of this article and save it to disk.
2) Boot up any disk with a DOS that allows a disk to be initialized, preferably one with a "fast" DOS such as Diversi-DOS, Pronto DOS, or normal DOS 3.3. (later, HyperDOS could be added to a normal 3.3 DOS)

PR#6

3) Initialize the BASIC pointers and clear any program in memory

FP

4) Type in a BASIC program that will run our Lode Runner main file.

```
10 TEXT : HOME : PRINT CHR$(4)"BRUN LODE RUN
   NER.B'
```

5) Insert a blank disk and format it

INIT LODE RUNNER

6) Transfer the game boards from the original disk to ours by running Super IOB with the Lode Runner controller installed
7) At this point, if you don't have a sector editor (such as The Inspector or Disk Edit), you can go ahead to Part Two. We really don't need to mark these tracks as "used" in the VTOC since the main Lode Runner B-file will be saved on tracks beyond track $11 but, just as a precaution, we mill mark them anyway. Marking them also enables us to save other programs on this disk but, if you want to copy Lode Runner to another disk, you must use COPYA. Delete the unwanted files afterwards.

Now get out your Sector Editor and modify the VTOC.

```
Read Track $11, Sector $00
Change bytes $44-$6B to $00
Write the sector back to the disk.
```

Part Two: Capture the Main Program

The modified ROM is used to shove pages $0-$7 into high memory so they can be saved out onto a normal disk. Most copy-cards will do the job just as well, but I found the modified ROM (from Hardcore COMPUTIST No. 6) the easiest to use as it leaves us in the monitor where we can perform the other necessary moves.

Mapping the memory of a running Lode Runner shows us what we need to save:

```
0000.08FF  program code
0900.0EFF  00's 0F00.1FFF;  program code
2000.5FFF  hi-res pages; no need to save
6000.8FFF  program code
9000.97FF  unused
9800.9AFF  00's
9B00.BFFF  program code
```

We will not save the regions with only 00's. Cleverly, we will clear the hi-res screen (thus filling it with 00's), then move the appropriate number of pages of 00's into the correct location.

8) Before capturing the main file, type in and save the following hexdump which will move everything back into place and start the game.

```
2900: A0 5A A9 9B A2 25 20 4B    $C20D
2908: 29 A0 2A A9 60 A2 30 20    $2EEF
2910: 4B 29 A9 40 85 E6 20 F2    $0416
2918: F3 A0 40 A9 09 A2 06 20    $8B5D
2920: 4B 29 A0 40 A9 90 A2 0B    $9D80
2928: 20 4B 29 A0 22 A9 02 A2    $5D67
2930: 07 20 4B 29 A0 00 B9 00    $AA5C
2938: 20 99 00 00 C8 D0 F7 B9    $1371
2940: 00 21 99 00 01 C8 D0 F7    $8BC4
2948: 4C 00 60 84 01 85 03 A9    $7A14

2950: 00 85 00 85 02 A0 00 B1    $FEB6
2958: 00 91 02 C8 D0 F9 E6 01    $C502
2960: E6 03 CA D0 F0 60          $BC3B
```

9) Check your typing with CHECKBIN and now save it to another disk (one other than our Lode Runner disk) with a

BSAVE LODE RUNNER.MOVES, A$2900,L$66

10) Enter the monitor

CALL -151

11) Clear memory

220:0 N 221<220.BFFEM

12) With the Modified ROM installed, insert LODE RUNNER and boot it up

C600G

13) When the title screen appears, Press RESET and then a ":" (colon) to move $0-$8FF to $2000-$28FF.

14) From the monitor, tuck the code in safely and compactly to prepare for booting a DOS 3.3 disk (which wipes out memory except for $900-$95FF)

2A00<6000.8FFFM
5A00<9B00.BFFFM

With the code relocated, the new memory map looks like this:

```
0F00.1FFF   same as before (0F00.1FFF)
2000.28FF   destination of 0000.08FF
2900.29FF   move routines go here
2A00.59FF   destination of 6000.8FFF
5A00.7EFF   destination of 9B00.BFFF
```

15) Preface the compacted code with a jump to our move routines

EFD:4C 00 29

16) Insert a DOS 3.3 disk and boot it

C600G

17) Install the move routines

BLOAD LODE RUNNER.MOVES

18) Insert the disk with the copied game boards on it and save the main Lode Runner file to disk

BSAVE LODE RUNNER.B, AEFD,L7003

19) You should make a backup copy immediately by COPYAing the disk to a blank or no longer needed disk.

You're done! Have fun!

Lode Runner Controller

```
1000 REM LODE RUNNER CONTROLLER
1010 TK = 3 :LT = 13 :CD = WR :MB = 151
1020 T1 = TK : GOSUB 490 : RESTORE : GOSUB 170 :
     GOSUB 270
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 222 ,235 ,222 ,170
```

Controller Checksums

```
1000  - $356B     1060  - $A2E5
1010  - $9BCC     1070  - $2133
1020  - $4A02     1080  - $9A47
1030  - $CC05     1090  - $00CC
1040  - $EC60     1100  - $B6BC
1050  - $ECDF     5000  - $5471
```

Softkey For A2-PB1 Pinball

By Joseph W. Leathlean

"Just like all the other programs that make it to the pages of Hardcore COMPUTIST, the only REAL problem with A2-PB1 is that it is protected. To make things worse (as if being copy protected isn't bad enough), the protection scheme causes a very slow booting process."

Use the procedure below to change all that!

A2-PB1 Night Mission Pinball
subLOGIC Communications Corp.
713 Edgebrook Drive
Champaign, IL 61820
$34.95

Requirements:
A2-PB1 Night Mission Pinball
1 blank disk
64K Apple][, //+, //e

A2-PB1 is, in my opinion, the best pinball game that you can get for the Apple. The game itself is well designed and, with the ability to change certain parameters to control the action of the game, you can make various versions of the game which can be either challenging, or easy. The graphics are superb and the unmodified ball motion is smooth as well. To play the game you may use the keyboard or two paddle buttons, either on (?) two paddles, a joystick or the open and closed apple keys on the //e.

The Problem

Just like all the other programs that make it to the pages of Hardcore COMPUTIST, the only REAL problem with the game is that it is protected. To make things worse (as if being copy protected isn't bad enough), the protection scheme causes a very slow booting process. It takes approximately 30 seconds to boot up the game.

After examining the disk using TRAX from Bag of Tricks, I determined that I would have to either use a WILDCARD device on the game, or boot code trace it. While any WILDCARD device will work, the boot process is still somewhat slow since 64K of memory must be loaded in. As you will see later, the game itself is only 36K in length. My first attempt at boot code tracing, it took me two hours to unprotect the program. This was almost a year and a half ago but, before I was able to write the article, I misplaced my notes.

When I finally decided to redo the procedure, it only took 15 minutes to break it down. Using these notes (the successful boot tracing venture under my belt) I was able to complete the procedure in less than five minutes. If you are experienced in boot code tracing it should take no more than five minutes to unprotect your A2-PB1.

I am presenting my procedure in two parts. The boot code trace with explanations of each step so newcomers might learn from it, and the procedure in cookbook form.

The Procedure Explained

Initialize a diskette using DOS 3.3 or a high-speed DOS such as PRONTO-DOS. Please use the method below so that there is a null HELLO program.

```
NEW
INIT HELLO
```

Clear the computer's memory

```
CALL -151
C081
C081
F800<F800.FFFFM
C083
C083
300:0
301<300.BFFFM
D000<1000.37FFM
```

This clears out all memory and copies the monitor over to the language card so it can be used. Because of the length of the program and the boot process, the boot code trace cannot be done on a 48L machine.

Move the disk controller ROM into the language card so it can be modified. Modify it to exit to $F701 instead of $801 and insert a routine to shut off the disk drive at $F701.

```
D600<C600.C6FFM
D6FA:F7
F701:AD E8 C0 4C 59 FF
```

Executing this modified boot stage will load track 0, sector 0 into $800.

```
D600G
```

After a moment the computer will sound off its somewhat annoying beep and the disk drive will stop. At this point memory location $800 contains the total number of sectors to be loaded by the $C600 code (normally and, in this case, a one (1)) and $801 is the next address to be executed.

Move the first stage code into the language card so that it can be modified.

```
D800<800.8FFM
```

Now examine the code at $D800. You should notice that it jumps to a routine at $D8DA which moves page eight down to page three and executes the moved code. We must modify this routine so that it moves the code from $D800 to $300 and locate where the page three code exits and force it to exit instead to $F700. The location I refer to is at $D8B4 which normally jumps to $800. Make the following modifications:

```
D803:D8
D8DE:D8
D8B6:F7
```

Now modify the boot code ROM to jump to this modified routine and load the next stage of the boot code by re-executing the controller ROM in RAM.

```
D6FA:D8
D600G
```

After a moment the drive will again stop and the next stage of the boot code will be loaded. This stage occupies memory from $800 to $1000. Even so, the only location that we need to modify is the first routine, listed below, which loads the rest of the program.

```
0800: 4C 03 08  JMP  $0803
0803: A9 90     LDA  #$90
0805: C5 7B     CMP  $7B
0807: F0 04     BNE  $080D
0809: 20 0E 03  JSR  $030E
080C: 00  BRK
080D: AD E8 C0  LDA  $C0E8
```

You should be able to figure out that the routine at $30E loads a page of memory from the disk and jumps to $800. This routine uses the same exit point as the last modification, so we can modify it to exit to the new location where we will put a modified version of the above routine.

```
E800<800.80CM
E802:E8
E80D:4C 00 F7
D8B6:E8
```

Now execute the controller routine to re-execute the boot code.

```
D600G
```

The program will be loaded in about 20 to 25 seconds, and occupies memory from $810-$90FF, but you just can't save that portion and later re-execute it. This is because the page three code contains something needed by the program. We have to save that code and have it reload when first executed.

If you remember, the boot code has a routine which moves memory from $800 to $300 and executes it. This routine is now in page three and can be modified. First we have to move page three to $9100 and modify the routine to move it back and jump to $810 or, better yet, make it into a subroutine which can be called from $80D.

```
9100<300.3FFM
91DE:91
91E6:60
80D:20 DA 91
```

Now we can save the program as one whole chunk and recall it by simply BRUNing it, but we must also get DOS into memory so we can BSAVE it. Because booting a DOS diskette destroys page eight, we must first protect it. If a HELLO program is present, it can destroy other areas. That is why you should have typed NEW before initializing the diskette.

```
9200<800.8FFM
```

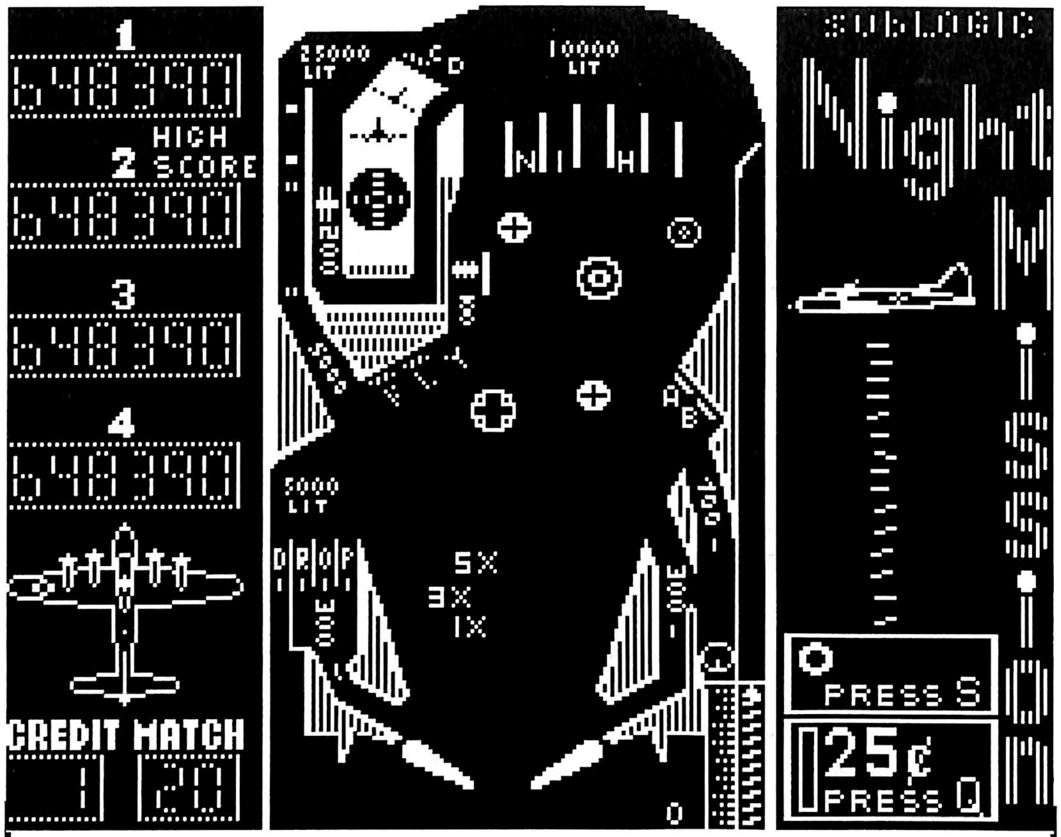
Insert new diskette now and boot it

```
C600G
```

When the diskette has completed booting, restore page eight, modify DOS so it can save programs longer than $8000 bytes and save A2-PB1

```
CALL -151
800<9200.92FFM
A964:FF
BSAVE NIGHT MISSION,A$80D,
  L$89F3
```

You now have an unprotected A2-PB1 program. To use it, either modify your HELLO program to automatically BRUN it or, after booting, type BRUN NIGHT MISSION.



1) Boot DOS 3.3 or high-speed DOS and format it with a null greeting program

```
NEW
INIT HELLO
CALL -151
```

2) Enable the language card RAM and copy the monitor into it

```
C081
C081
F800<F800.FFFFM
C083
C083
```

3) Wipe out most of memory

```
300:0
301<300.BFFFM
D000<1000.37FFM
```

4) Move the controller ROM to RAM and modify the boot slightly

```
D600<C600.C6FFM
D6FA:F7
F701:AD E8 C0 4C 59 FF
```

5) Insert your A2-PB1 disk and execute the partial boot

```
D600G
```

6) Copy $800 into $D800, modify the copy so it will work at the new location and then jump into the monitor and execute this partial boot

```
D800<800.8FFM
D803:D8
D8DE:D8
D8B6:F7
D6FA:D8
D600G
```

7) Modify the boot some more and execute it

```
E800<800.80CM
E802:E8
E80D:4C 00 F7
D8B6:E8
D600G
```

8) Compact the code

```
9100<300.3FFM
91DE:91
91E6:60
80D:20 DA 91
9200<800.8FFM
```

9) Insert the disk you initialized in Step 1 and boot it

```
C600G
```

10) Enter the monitor and move the $800 code back to its original location

```
CALL -151
800<9200.92FFM
```

11) Modify DOS so that it can save more than 32K and save Night Mission

```
A964:FF
BSAVE NIGHT MISSION,A$80D,
  L$89F3
```

Enjoy your deprotected version.

Customized Drive Speed Control

By William Wingfield, Jr.

Requirements:
Two disk drives recommended
Low-wattage soldering iron
Small hook-up wire
1K 10-turn pot and dial
Two 2.2K 1/4W or more resistors
A few screwdrivers
An Exacto blade
A small amount of heat shrink or spaghetti tubing
(a Multitester is recommended)

Note: The procedure described below requres dismantling of the disk drive which voids its warranty. Hardcore COMPUTIST will not be held responsible for any damages incurred while following this procedure.

Like myself, many of you would probably like to be able to adjust the drive speed on your Apple's disk drive without removing its cover. In fact, you may have already drilled a hole in the case to reach the speed control with a small screwdriver. Although this is much more convenient than removing the drive cover to adjust the speed, it still doesn't compare with simply turning a knob and being done with it.

The following article details a procedure in which a 1K (instead of the usual 5K) variable resistor is mounted on the front of the disk drive to provide easy access to the disk drive and a much finer control over its speed (you'll appreciate this modification to your disk drive the very first time you have to match a nibble count while making a backup).

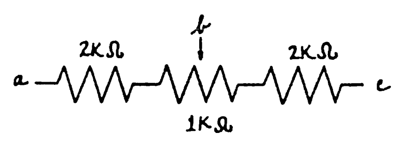
A Bit of Background

The speed control pot for the Apple Drive is on the back circuit board in the drive. You must use a very small screwdriver to access it, and a slight turn is usually all that is needed to adjust the drive speed.

This speed control, a 5,000 ohm (5K), 10-turn mini-potentiometer, has three contacts (we'll call them A, B and C) which are arranged as shown in Diagram A.

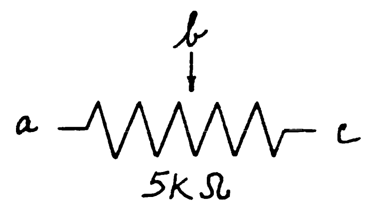
The resistance between points A and C is always 5K ohms. Point B slides along the resistive (??????) surface as you turn the knob. This has the effect of varying the resistance between point A and B and between point B and C. The resistance levels (from A to B and from B to C) vary inversely so that the resistance between A and B plus the resistance between B and C always equals 5K ohms.

Diagram A



We are going to replace this control with a standard-sized 1K pot (padded to 5K) that will be mounted on the front of the drive where it is easily accessible. See resulting schematic below in Diagram B.

Diagram B



Notice how the characteristics of this schematic match those of the original 5K pot schematic. This similarity is a necessity.

Digging Into The Drive

1) Turn off the power to your Apple, touch the power supply to remove excess static and, just to be safe, disconnect the power cord.
2) Disconnect the drive cable from the controller card by firmly and evenly pulling upward on it.
3) Remove the four screws that secure the cover of the disk drive (located on the bottom surface of the Apple drive) and slide the cover in a backwards direction so that it slips away from the unit. Place the drive cover and accompanying screws in an out-of-the-way place.
4) Open the drive door, remove the four screws (located on the left and right sides of the face plate) from the face-plate and gently pull it forward to allow the door to slide out of the guide tracks.
5) Use a small screwdriver or knife to pop the

plastic retaining ring away from the LED mount as illustrated in Diagram C.

6) Gently pull the LED out of the mount and the face-plate will come free of the drive.

7) About 1" of the plastic bracing behind the face-plate must be removed to allow the pot to be mounted. Use a scratch awl to score the plastic and snap the section of brace out (refer to Diagram D for the mounting hole location).

8) Drill the mounting hole for the pot and place your face plate and its four screws aside for the moment.

9) Disconnect the other end of the ribbon cable from the large circuit board. Be sure to note which way it is oriented.

1Ø) Take out the four screws (on the bottom of the drive) that secure the bottom part of the cover and remove it. Set the bottom cover, ribbon cable and four screws aside.

11) Remove the two screws from the small circuit board in the rear (the board with the speed adjustment pot on it). Note which terminal is the CCW and which is the CW as marked on the 5K mini-pot.

12) Desolder the mini-pot.

13) Cut three pieces of wire about 1Ø inches long and solder one to each position where the pot terminals were (see Diagram E).

14) Remount the small circuit board using the small spacers.

15) If you have a VOM, measure the resistance between the wiper terminal and the other two. They should both be between 2K and 3K showing that the wiper is somewhere near the center of its rotation. If it isn,t you'll have to compensate the padding resistors accordingly.

16) Clip the resistor leads to 1/2" and solder the two 2K padding resistors to the 1K 1Ø-turn pot, one to the CW terminal and one to the CCW.

17) Solder the wires from the circuit board to the corresponding lead on the pot assembly: CW lead to the resistor on the CW terminal, CCW lead to the resistor on the CCW terminal, and wiper lead to the wiper terminal. Cover the connections with heat-shrink or spaghetti tubing.

18) Put the LED back in the face plate, put the face-plate back on the drive and mount the 1Ø-turn pot per the instructions that come with it. Be sure that you can open and shut the disk drive door freely, then put the face plate screws back in.

19) Look over your work carefully and make sure that the connections are in good shape.

2Ø) Replace the bottom cover (making sure that the screw holes are lined up with the counter sunk holes in the cover) and hook up the ribbon cable to the large circuit board.

21) Connect the drive back to your Apple and set the control to the middle of its rotation. If you have two drives, connect this drive up as D2 and boot a speed test program in D1. Set the speed on D2 with the new pot that you just installed. If you only have one drive, put a disk in the drive and see if it boots. If it does you're almost done. Use a speed test program and fine tune it. Otherwise, turn the dial and keep trying to boot until it does.

22) If the speed won't adjust properly, turn off the Apple and re-inspect all of your work. Check to make sure that you haven't damaged the drive while handling it. Measure the resistance on the original pot again and make sure that the value is in the range of the new one with the resistors used. As a last resort, replace the original pot and try a boot. If there is still a problem you know that it's something other than the pot.

23) When you have the drive working, replace the top cover.

Diagram D

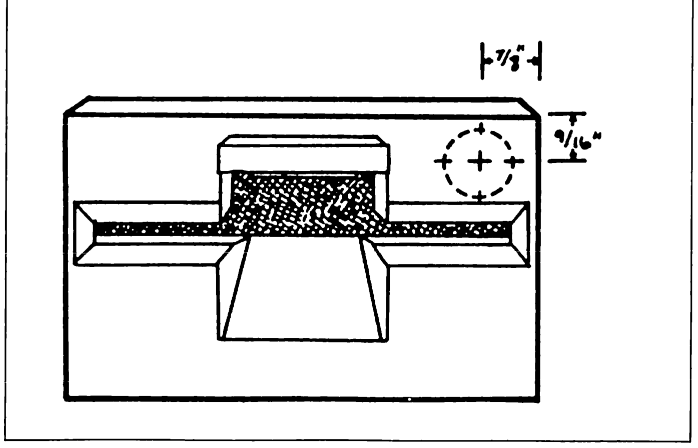
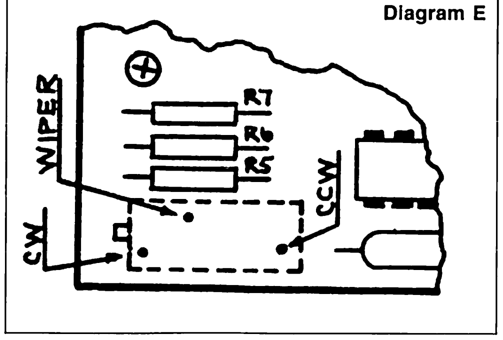


Diagram E



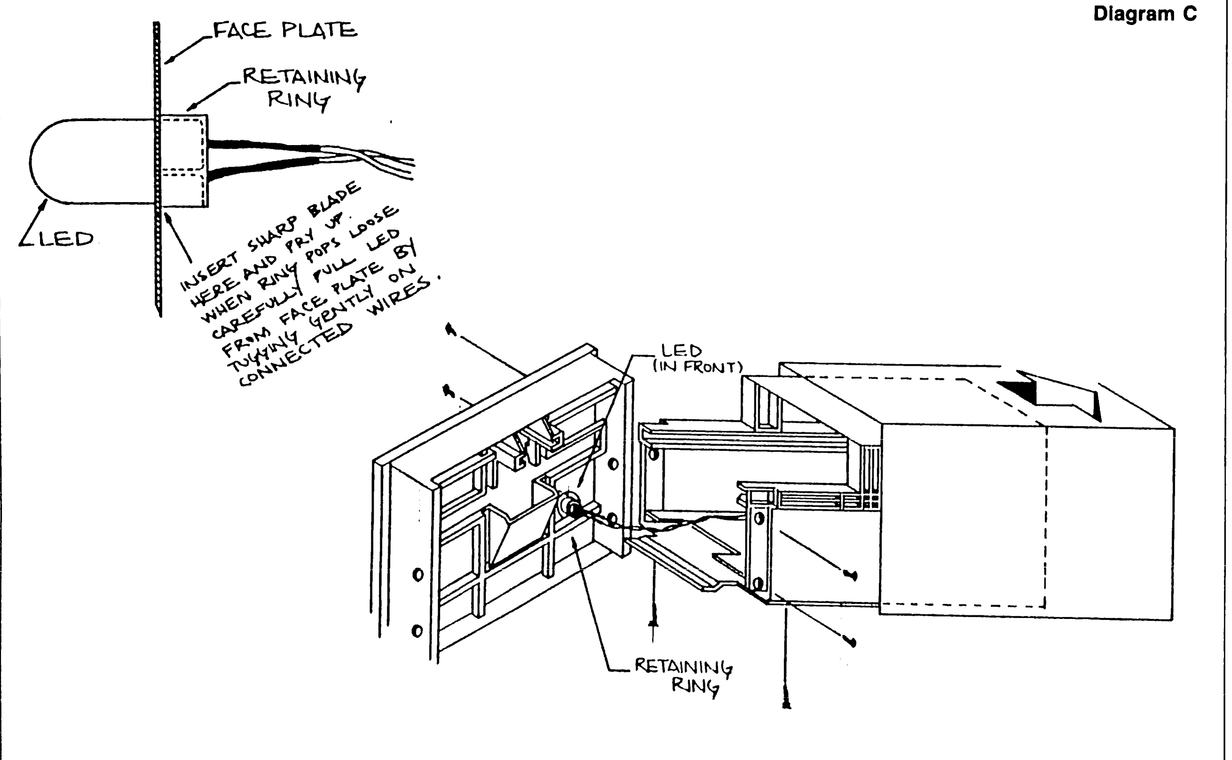
Now you can adjust the speed on your disk drive quickly and easily whenever you want. Doing nibble counts are a breeze and the dial lets you return to the correct speed afterwards.

As stated earlier a 5K pot can be used (without padding resistors) but a 1K allows for more accurate control plus it provides protection of the drive in the event that the pot is turned fully to one extreme or the other.

Here are a couple of recommendations which you might find helpful: If you can, get a locking dial so people won't be able to play with the setting when you aren't looking. And if you have a choice of color when purchasing a dial, get a black one. It looks better than silver.

If you aren't experienced with this type of work *DON'T* try it yourself- you could seriously damage your drive with improper handling. However, if you know what you're doing and decide to make the modification, I think you'll find it an enjoyable and satisfying project.

Diagram C


FACE PLATE
RETAINING RING
LED
INSERT SHARP BLADE HERE AND PRY UP. WHEN RING POPS LOOSE CAREFULLY PULL LED FROM FACE PLATE BY TUGGING GENTLY ON CONNECTED WIRES.
LED (IN FRONT)
RETAINING RING

Super I.O.B. v 1.5

With the addition to the machine language portion of Super I.O.B. plus the inclusion of a new subroutine, this newest version of Super I.O.B. will copy a disk more than 3 times faster than its predecessors.

by Ray Darrah

Requirements:
Apple][Plus, //e or //c
One DOS 3.3 disk drive

Somewhere near the beginning of time, as Hardcore knows it, an Applesoft program called IOB was written by "Bobby." This program was designed to deprotect disks by interfacing directly with a machine language program in DOS called the Read/Write Track Sector program (RWTS). IOB was so named because it modified a parameter list in DOS called the Input/Output control Block (IOB). At first, quite a bit of confusion was generated due to the fact that the Applesoft program had the same name as the parameter list used by the RWTS.

Not quite as long ago (a few short eons in the evolution of Hardcore) a program with the same concept as IOB was introduced. This new program was called Super IOB and, in all modesty, the new name was justly given because of the several advantages it offered over the original IOB. Perhaps the most notable of these is the controller concept in which a short subprogram is keyed into the main portion of the Super IOB program thus changing it adequately to allow it to copy many disks with varied protection schemes. Although IOB incorporated this same concept, due to the structure of the program, it was often necessary to change other lines of the program in addition to keying in a controller.

When you get right down to it, Super IOB's power is derived primarily from a small collection of Applesoft subroutines. These subroutines are called by the inserted subprogram (the Super IOB controller) in a particular sequence to allow deprotection of individual programs. The concept of Super IOB is simple, but only by understanding the function of the subroutines in Super IOB can an Applesoft programmer learn to create a controller for a specific disk.

Since the introduction of Super IOB, Hardcore COMPUTIST has used this flexible program to deprotect (or partially deprotect) dozens of commercial programs. In addition, Hardcore COMPUTIST has printed several utilities designed to aid the programmer in his use of Super IOB. These include "CSaver" (Hardcore COMPUTIST No. 13), a machine language program which helps in the merging of the controller and Super IOB, and "The Controller Writer" (Hardcore COMPUTIST No. 16) an Applesoft program which asks the user questions and then formulates the components of a Super IOB controller (depending upon the answers) which are needed for that particular application.

Presented here is the second update to Super IOB. Because of an addition to the machine language portion of Super IOB and the inclusion of a new subroutine this new version of Super IOB now can copy a disk up to 350% faster than its predecessors.

Keying in Super IOB for the First Time

If you don't already have Super IOB, you will have to use the procedures outlined on page 2 of this magazine to key in the Applesoft and Machine Language portions of the program. Save the Applesoft portion with:

SAVE SUPER IOB

And save the new machine language portion with:

BSAVE IOB.OBJ0,A$300,L$A8

Updating Super IOB v1.2

If you have a copy of Super IOB v1.2, to update it, you need only key in the BASIC lines whose corresponding checksums are shaded and key in the new hexdump. Use the commands listed directly above to save the two portions of your new Super IOB. You may then skip to the section entitled "The New Routine."

The Function Of Super IOB

Super IOB de-protects disks by pushing the RWTS subroutine in DOS to its upper-most limits. Because of this, it only copies disks with sectors that somewhat resemble normal sectors.

Before a disk can be softkeyed, the sector alterations must be determined. The easiest way to do this is to use a nibble editor program [like Bag of Tricks, DiskView or The Linguist (from CIA)] which shows these sector differences.

Once the protection has been discovered, a controller program (lines 1000 through 9999) designed to deprotect such a scheme must be inserted into Super IOB. Here is a list of the protection schemes Super IOB was designed to softkey:

1) **Altered address, data, prologue, or epilogue marks**
2) **Strangely numbered sectors or tracks**
3) **Modified RWTS (with same entry conditions)**
4) **Half tracks for any of the above**
5) **13 or 16-sector base format for any of the above**

A Little Briefing

The following is a short discussion of the protection scheme and how each relates to Super IOB. Keep in mind that often more than one scheme is used at a time. This has the effect of complicating the Super IOB controller.

Altered Marks

DOS looks for specific marks when trying to read a sector. Changing these is a common practice, especially on older releases. As previously explained in the Whiz Kid column, DOS puts certain reserved bytes on the disk (during INITialization) so it can determine where a sector begins and ends.

For example, a normal 16-sector disk has the bytes D5 AA AD designating the start of thedata field which contains the 256 bytes of a sector in encoded form. When a standard RWTS tries to find a sector, it looks for these marks. If they are not found, (either because they don't exist or they have been changed to something else) DOS returns with the dreaded I/O ERROR.

The sequences of the four reserved-byte marks (start of address, end of address, start of data, end of data) are handled by subroutines in Super IOB. These subroutines change the marks that the current RWTS looks for when reading.

Strangely Numbered Sectors

Within an address field, there are 8 bytes which alert the RWTS to which sector is about to pass under the read/write head. On some disks, these are not standard. These disks are easily softkeyed with Super IOB. The controller instructs Super IOB to read the sectors using the strange sector numbers and then write them using the correct numbers. This works because the RWTS merely compares the sector number found on the disk with the one the controller is looking for (even if it is higher than 15 and; therefore, illegal).

Modified RWTS

The disk-protectors will often rearrange and/or modify the standard RWTS subroutine. If this is the case, you must first save the strange RWTS onto a normal DOS disk and then use a controller which reads the protected disk using the strange RWTS and then writes via the normal 3.3 RWTS.

Such a controller is included in this article. It is called the SWAP controller. This is because of its use of the "Swap RWTS at $19ØØ with the one at $B8ØØ" routine in Super IOB.

Since the RWTS of a protected disk will be modified to read any altered DOS marks, this is an easy method to use if you are unable to determine the protection scheme.

Half-Tracks

There is a motor inside your disk drive that is responsible for moving the disk head to the various tracks. Known as the stepper motor, it has four electromagnets (numbered Ø to 3) that can be turned on or off by referencing memory locations. When one of these magnets is turned on, the permanent magnets in the motor are attracted to it and the motor shaft turns until the permanent magnets are aligned with the electromagnets.

To obtain continuous motion, a program would:

1) Turn a magnet (called phase) on
2) Wait for the motor to get aligned (it doesn't take much time)
3) Turn off the magnet
4) Turn on the next adjacent magnet (the next magnet differs depending on whether you want to go to a higher or lower track)
5) Go to step 2

Because of the resolution of the disk head combined with the accuracy of the stepper motor, normal DOS tracks are placed only on the even phases. This means that for every track DOS moves, it references two magnets. As a result, the disk head never stops at any of the odd phases (i.e. aligned with magnets 1 or 3). Therefore, the odd phases are commonly called half-tracks.

The disk-protectors will often put information on these phases that are inaccessible to normal DOS. A routine called "Move S Phases" (in Super IOB) handles the job of getting to these so called half-tracks and can also be used (by a controller) to get to tracks that have been

Super IOB v1.5 Source Code

```
*----------------------------------------------------------------------------*
*                         Super IOB machine routines                         *
*
*                               BY RAY DARRAH                                *
*----------------------------------------------------------------------------*

Ø3D9- RWTS.B8ØØ .EQ $Ø3D9   ENTRY POINT TO RWTS @$B8ØØ
D412- INVOKERROR .EQ $D412  ROUTINE THAT CASES BASIC TO DO THE ERROR CONTAINED IN X
1EØØ- RWTS.19ØØ .EQ $1EØØ   ENTRY POINT TO THE RWTS AT $19ØØ
B9AØ- SEEKABS  .EQ $B9AØ    ENTRY POINT TO THE SEEKABS ROUTINE AT $B8ØØ
ØØDE- BAS.ERR  .EQ 222      ;BASIC ON ERR ERROR CODE
ØØFC- SWFRM    .EQ $FC      ;EXCHANGE FROM PARAMTER
ØØFE- SWTO     .EQ $FE      ;EXCHANGE RWTS 'TO' PARAMETER
ØØEØ- PAGES    .EQ $EØ      ;NUMBER OF PAGES OF MEMORY TO EXCHANGE
ØØ24- CH       .EQ $24      CURSOR X POSITION
FDDA- PRNTBYTE .EQ $FDDA    PRINTS HEXADECIMAL BYTE
               .OR $Ø3ØØ    STARTS AT PAGE THREE
*              .TF IOB.OBJØ                                                  *

*----------------------------------------------------------------------------*
*                                 CALL RWTS                                  *
*----------------------------------------------------------------------------*

Ø3ØØ: A9 Ø3     IO        LDA /TABLETYP ENTRY POINT FOR CALLING THE RWTS THROUGH BASIC
Ø3Ø2: AØ ØA               LDY #TABLETYP A,Y POINT TO THE IOB TABLE
Ø3Ø4: 2Ø D9 Ø3            JSR RWTS.B8ØØ GO TO THE RWTS AT $B8ØØ
Ø3Ø7: BØ 16               BCS DOS.ERR   IF THE CARRY SET THEN CAUSE BASIC ERROR
Ø3Ø9: 6Ø                  RTS           OTHERWISE, ALL IS WELL SO RETURN
Ø3ØA: Ø1        TABLETYP .HS Ø1         TYPE OF TABLE (1=IOB)
Ø3ØB: 6Ø        SLT      .HS 6Ø         SLOT TO BE ACCESSED NEXT (VIA POKESLT,SO)
Ø3ØC: Ø1        DRV      .HS Ø1         DRIVE TO BE ACCESSED NEXT (1 OR 2)
Ø3ØD: ØØ        VOL      .HS ØØ         VLUME TO BE ACESSED (Ø=ANYTHING WILL DO)
Ø3ØE: ØØ        TRK      .HS ØØ         TRACK TO ACCESS
Ø3ØF: ØØ        SCT      .HS ØØ         SECTOR TO ACCESS
Ø31Ø: 1B Ø3     DCTPTR   .DA DCT        POINTER TO THE DEVICE CHARACTERISTICS TABLE
Ø312: ØØ        BUFFERLO .HS ØØ         ALWAYS MAKE LSB OF BUFFER POINTER ZERO!
Ø313: 27        BUF      .HS 27         SECTOR BUFFER PAGE POINTER
Ø314: ØØ        NOTHING  .HS ØØ         NOT USED
Ø315: ØØ        BYTCOUNT .HS ØØ         BYTE COUNT FOR PARTIAL SECTOR (Ø=256 BYTES)
Ø316: ØØ        CMD      .HS ØØ         COMMAND CODE (Ø=SEEK)
Ø317: ØØ        RWTS.ERR .HS ØØ         ERROR CODE THAT THE RWTS.B8ØØ RETURNS WITH
Ø318: ØØ        OVL      .HS ØØ         VOLUME NUMBER OF LAST ACCESSED DISK
Ø319: 6Ø        OLDSLT   .HS 6Ø         SLOT PREVIOUSLY ACCESSED
Ø31A: Ø1        OLDDRV   .HS Ø1         DRIVE PREVIOUSLY ACCESSED
Ø31B: ØØ        DCT      .HS ØØ         DEVICE TYPE OF DEVICE CHARACTERISTICS TABLE
Ø31C: Ø1        PHASES   .HS Ø1         PHASES-1 PER TRACK, (Ø OR 1)
Ø31D: EF D8     MOTORCNT .HS EFD8       MOTOR-ON TIME COUNT
Ø31F: AD 17 Ø3  DOS.ERR   LDA RWTS.ERR DOS HAS HAD AN ERROR, GET THE ERROR CODE
Ø322: 4A                  LSR           DIVIDE IT BY 16
Ø323: 4A                  LSR
Ø324: 4A                  LSR
Ø325: 4A                  LSR
Ø326: AA                  TAX           TRANSFER IT TO X SO BASIC WLL INDUCE THE
                                        FALSE ERROR CODE
Ø327: 4C 12 D4            JMP INVOKERROR CAUSE A BASIC ERROR

*----------------------------------------------------------------------------*
*                             MOVE THE DISK ARM                              *
*----------------------------------------------------------------------------*

Ø32A: A9 ØØ     MOVPHASES LDA #$ØØ      ROUTINE TO SET UP REGISTERS BEFORE CALLING
                                        SEEKABS
Ø32C: A2 ØØ               LDX #$ØØ      X AND A HAVE DUMMY NUMBERS THAT WILL BE POKED
                                        INTO BY "MOVE S PHASES"
Ø32E: 4C AØ B9            JMP SEEKABS

*----------------------------------------------------------------------------*
*                          CAUSE ERROR IN CONTROLLER                         *
*----------------------------------------------------------------------------*

Ø331: A6 DE     BASICERR LDX BAS.ERR    BASIC HAS MADE AN ERROR SO CAUSE THE ERROR
                                        NUMBER AT 222
```

continued

```
0333: 4C 12 D4          JMP INVOKERROR
*----------------------------------------------------------------------------*
*                              POP OFF RETURN                                *
*----------------------------------------------------------------------------*
0336: 68        POP     PLA          ROUTINE TO POP OFF ONE RETURN (BASIC) ADDRESS
0337: A8                TAY
0338: 68                PLA
0339: A6 DF             LDX BAS.ERR+1 GET WHAT STACK WOULD BE IF GOSUB WASN'T THERE
033B: 9A                TXS          PUT THAT AS THE STACK POINTER
033C: 48                PHA
033D: 98                TYA          RESTORE THE LAST RETURN ADDRESS
033E: 48                PHA
033F: 60                RTS
*----------------------------------------------------------------------------*
*                              EXCHANGE RWTS's                               *
*----------------------------------------------------------------------------*
0340: A0 00             LDY #0       ;ZERO THE LSB's
0342: 84 FC             STY SWFRM    ;AND HAVE Y AT ZERO FOR START
0344: 84 FE             STY SWTO
0346: B1 FC     MOVE.PAGE LDA (SWFRM),Y ;GET A BYTE
0348: 48                PHA          ;AND SAVE IT
0349: B1 FE             LDA (SWTO),Y ;GET THE BYTE WHERE THE SAVED ONE GOES
034B: 91 FC             STA (SWFRM),Y ;AND STORE IT WHERE THE SAVED ONE WAS
034D: 68                PLA          ;GET THE SAVED BYTE
034E: 91 FE             STA (SWTO),Y ;AND STORE IT WHERE IT GOES
0350: C8                INY          ;DONE WITH A PAGE
0351: D0 F3             BNE MOVE.PAGE ;NO KEEP WORKING ON IT
0353: E6 FD             INC SWFRM+1  ;GET NEXT MSB's
0355: E6 FF             INC SWTO+1
0357: C6 E0             DEC PAGES    ;DECREMENT THE NUMBER OF PAGES TO MOVE
0359: D0 EB             BNE MOVE.PAGE ;IF NOT DONE, MOVE ANOTHER PAGE
035B: 60        RTS1    RTS          ;FINISHED, RTS
*----------------------------------------------------------------------------*
*                         READ OR THE ENTIRE BUFFER                          *
*----------------------------------------------------------------------------*
035C: 97        MB      .HS 97       HIGHEST BUFFER PAGE+1
035D: 35        LT      .HS 35       LAST TRACK TO GET+1
035E: 0F        LS      .HS 0F       (LAST SECTOR TO GET-1)MOD16

035F: A9 00             LDA #0       DO A VTAB3
0361: 85 28             STA $28
0363: A9 05             LDA #5
0365: 85 29             STA $29
0367: A9 10     NEWTRK  LDA #16      HTAB
0369: 85 24             STA CH
036B: AD 0E 03          LDA TRK
036E: 20 DA FD          JSR PRNTBYTE PRINT IT
0371: A9 1C             LDA #28      HTAB
0373: 85 24             STA CH
0375: AD 0F 03          LDA SCT
0378: 20 DA FD          JSR PRNTBYTE
037B: 20 00 03          JSR IO       GET A SECTOR
037E: CE 0F 03          DEC SCT      NEXT SECTOR
0381: 10 08             BPL NXT.PG
0383: A9 0F             LDA #15      RESTORE TO SECTOR 15
0385: 8D 0F 03          STA SCT
0388: EE 0E 03          INC TRK      NEXT TRK
038B: EE 13 03  NXT.PG  INC BUF      NEXT PAGE OF MEMORY
038E: AD 13 03          LDA BUF
0391: CD 5C 03          CMP MB
0394: B0 C5             BCS RTS1     BUFFER FULL, RETURN
0396: AD 0F 03          LDA SCT      LAST SECTOR?
0399: CD 5E 03          CMP LS
039C: D0 C9             BNE NEWTRK
039E: AD 0E 03          LDA TRK
03A1: CD 5D 03          CMP LT
03A4: 90 C1             BCC NEWTRK
03A6: B0 B3             BCS RTS1     END OF DISK
```

end of Super IOB source code . . . .

marked as other tracks. A complete discussion of how to use the routine appears later in this article.

Anatomy Of A Controller

Before we attempt to write a controller, let's look at the subroutines at the controller's disposal. During this explanation, it would be wise to refer to the listing of Super IOB to see how each is accomplished.

The New Routine

There has only been one routine added to Super IOB to arrive at version 1.5. The explanation of its workings follows.

Name: R/W A RANGE
Line Number(s): **610** - 620
Entry Conditions: TK,ST= first sector to read or write, LT,LS= last sector (minus one MOD 16) to read or write, MB= maximum buffer page, CD= command code

This routine quickly reads or writes a range of sectors by calling a machine language program. To be faster, this machine language program stores the sectors in memory in a decreasing manner. That is: sector $0F is followed by sector $0E and sector $0D and so on.

Storing the sectors in memory in a different sequence than ascending is O.K. as long as, 1) the sectors are written in the same order they are read, and 2) "The Sector Editor" routine knows that the sectors were stored in descending order by setting the variable FAST equal to 1.

The Older Routines

The following is an explanation of the remainder of the routines found in Super IOB Version 1.5. These routines were included in the older Super IOB programs.

Name: START UP
Line Number(s): **10** - 60
Entry Conditions: Not Applicable

The first few lines merely identify the program; however, line sixty sets HIMEM and LOMEM so that they fit the memory usage requirements (see Memory Map following). It then goes to "CONFIGURATION TIME".

Name: INITIAL IOB SETUP
Line Number(s): **80**
Entry Conditions: DV= drive to be accessed, VL= volume of disk to be accessed, SO= slot to be accessed

This subroutine is normally GOSUBed via "TOGGLE READ/WRITE". Its purpose is to reset the buffer page and set the drive, slot and volume number to the disk to be accessed next.

Name: **R/W SECTOR**
Line Number(s): **100** - 110
Entry Conditions: TK= Track to be accessed, ST= Sector to be accessed, CD= Command code for the RWTS

This subroutine is GOSUBed directly from the controller. It reads or writes (depending upon CD) at the specified track and sector.

Name: **MOVE S PHASES**
Line Number(s): **130** - 140
Entry Conditions: SO= Slot of drive to move, DV= Drive number of drive to move, PH= Phase number that the disk head is currently over, S= Number of phases to move

This routine moves the disk read head the number of phases specified by S (one phase equals one half-track) and is capable of moving in either direction up to 128 phases (or 64 tracks). Care should be taken that PH + S isn't greater than 255 or less than 0 or an error will occur.

Name: **ALTERED ENDING MARKS**
Line Number(s): **170**
Entry Conditions: Proper DATA pointers

This routine changes the address field and data field epilogue markers in the normal RWTS. The values to change these to should be contained in a DATA statement. Because normal DOS only checks the first two bytes of these markers, only four values are required. The address field is changed first and should appear first in the data statement.

Name: **ALTERED ADDRESS MARKS**
Line Number(s): **190**
Entry Conditions: Proper DATA pointers

This routine modifies the RWTS (via POKE) so that it looks for a different sequence of address field prologue marks. The decimal values of the marks to look for should be stored as the next DATA elements.

Name: **ALTERED DATA MARKS**
Line Number(s): **210**
Entry Conditions: Proper DATA pointers

Same as previous subroutine except for DATA field prologue marks.

Name: **NORMALIZER**
Line Number(s): **230** - 250
Entry Conditions: None

This routine restores the values in the RWTS subroutine that are changed by any routine in Super IOB. This routine should be called just before writing in order to fix the RWTS so that it can access normal DOS disks.

Name: **IGNORE ADDRESS CHECKSUM**
Line Number(s): **270**
Entry Conditions: none

This routine modifies the RWTS subroutine so that it doesn't examine the checksum byte of the address field. This routine has been incorporated in many controllers.

Name: **ALTERED DATA CHECKSUM**
Line Number(s): **290**
Entry Conditions: Proper DATA pointers

This routine alters the starting checksum byte that the RWTS subroutine will use when reading a DATA field. The normal value for the RWTS is 0. The value to change the checksum to should be the next DATA element.

Name: **THE SECTOR EDITOR**
Line Number(s): **310** - 340
Entry Conditions: Proper DATA pointers and Elements, T1 = lowest track in buffer, TK = highest track in buffer

This routine automatically performs sector edits as the copy process goes on. It must be called (via GOSUB) just after reading a range of tracks. To indicate how many sector edits are to be performed, you must have a DATA element that has the number of sector edits followed by the word "CHANGES." For example:

```
1100 DATA 7 CHANGES,1,1,3,4
```

would tell the sector editor that the next 28 DATA elements are sector edits. This is because each sector edit is defined in four DATA elements. The location of the "x CHANGES" element in the DATA string does not matter because the sector editor will search it out and use the elements immediately following it.

The format for the four bytes that define a sector edit is: TRACK, SECTOR, BYTE, CHANGE TO. Each element is decimal and should be within the correct ranges since no error checking is done.

If you use the "R/W A Range Quickly" routine and you wish to perform some sector edits, you must set FAST equal to 1 so that this routine will be able to locate the specified sector in memory.

Name: **EXCHANGE RWTS's**
Line Number(s): **360**
Entry Conditions: A RWTS at $1900

This is the standard swap RWTS's routine. It uses a routine in IOB.OBJ0 to exchange the RWTS at $1900 with that which is located at $B800, the normal location for an RWTS. To tell the machine language swap routine (invoked by a CALL 832) what to exchange, a few POKE's must be executed. They are:

POKE 253,	start of first location
POKE 255,	start of second location
POKE 224,	number of pages to exchange

(a standard RWTS is eight pages long)

Name: **FORMAT DISK**
Line Number(s): **380** - 410
Entry Conditions: S2= slot of disk to format, D2= drive number of disk to format

This routine formats the target disk. It was meant to be used before the controller takes hold of Super IOB (and is GOSUBed by "Configuration Time") but can be called by the controller should the need arise.

Name: **PRINT TRACK & SECTOR #**
Line Number(s): **430**
Entry Conditions: TK= The track number to display, ST= The sector number to display

This is the subroutine that puts the current track and sector number at the top of the screen in hexadecimal during the softkey operation. It should be invoked just before reading or writing each sector.

Name: **CENTER MESSAGE**
Line Number(s): **450**
Entry Conditions: A$= The message

This routine prints a message in the center of the screen at the current VTAB position. Care should be taken that the message to print is not longer than 40 characters. If so, an error will occur.

Name: **PRINT MESSAGE AND WAIT**
Line Number(s): **470**
Entry Conditions: A$= The message

This routine uses "Center Message" to print the intended message at a VTAB of 11 and then it prints "PRESS ANY KEY TO CONTINUE." It waits for a keypress before RETURNing.

Name: **TOGGLE READ/WRITE**
Line Number(s): **490** - 530
Entry Conditions: CD= current command code

This routine toggles the state of CD (from ReaD to WRite and vice versa) and prints the current mode in flashing letters at the very top of the screen. In addition, if the user has only one drive, it asks him to swap disks. It then exits via "INITIAL IOB SETUP," thus making the sector buffer ready for the next operation.

Name: **IGNORE UNREADABLE SECTORS**
Line Number(s): **550** - 590
Entry Conditions: Not Applicable

If the controller should pay no attention to unreadable sectors, then somewhere in the beginning of it should be an "ONERR GOTO 550." This is used usually with RWTS.13 (since DOS 3.2 sectors are unreadable until they have been written to) but can be used with any disk that has unreadable sectors which should be ignored.

Note that this routine will not function correctly if you are using "R/W A Range Quickly". To ignore errors when using this routine, insert a POKE 775,96 into the beginning of your controller.

The Remainder Of The Program

Lines 1000 through 9999 are meant for the controller and all DATA statements it contains. All lines greater than 9999 are used by the error trapper or the configurer which consists of all the prompts when the program is run. The error trapper will print a disk error and stop the program. If the error wasn't a disk error, the error trapper will let it occur.

Now that you have an idea of the subroutines, take a look at the Variable Usage Table (Table 1) and note how the variables relate to them.

Memory Usage

Before actually looking at some controllers, let's say a few words about memory usage.

Following is a memory allocation table for the various parts of Super IOB. It is extremely important to stay within the boundaries when writing a controller. If not, horrible things may happen (the least of which would be the production of an incorrect copy).

Memory Allocation Table

$0800.$18FF (2048-6399)
intended for the Applesoft part of Super IOB

$1900.$20FF (6400-8447)
space allocated for a moved RWTS.

$2100.$26FF (8448-9983)
Super IOB Applesoft variable space

$2700.$96FF (9984-38655)
enough space for 7 tracks, this is the sector buffer

First, notice the amount of space available for the BASIC program. The Super IOB program as listed (with all REM's) ends about 290 bytes short of the $20FF (or 8447) limit. This leaves ample room for any controller that doesn't use an alternate RWTS. However, if a controller does use an alternate RWTS, then there are only about 900 bytes free for it. In view of the space requirement, the end of program should be checked by typing

PRINT PEEK(175) + PEEK(176) * 256

Before a controller with an alternate RWTS (Swap Controller, etc.) is used. If it has exceeded the 6399 limit, I suggest DELeting all subroutines not referenced by the controller and all REM lines until it fits within the allocated space.

Second, observe the 1534 bytes for variables. This should be enough space for the simple softkey procedure. It is impossible to allocate more memory for variables and use an alternate RWTS file. If you find that you need more memory and the program does not use RWTS.13 or some other moved RWTS, the **LOMEM: 8448** command may be removed from line 60. This will allocate what isn't used (by the BASIC program) of the 2K area reserved for the relocated RWTS as variable space.

Never omit the "HIMEM:" statement!

This could cause variables to overflow into the sector buffer, thus making a faulty copy.

With all this new knowledge, we are finally ready to scrutinize some sample controller programs. Keep in mind that protection schemes can be used with one another. Therefore, a more sophisticated controller for Super IOB will probably be required for most softkeys. Even so, developing new controllers isn't difficult.

The New Standard

If you were to install any of the previous controllers printed by Hardcore COMPUTIST into the new Super IOB, they would work fine, but would be considerably slower than a controller designed for v1.5. To take advantage of the new subroutines in Super IOB v1.5, a new standard controller (the building block controller that only copies normal disks) is necessary. This new controller has been named The Fast Controller. The Fast Controller, as well as a description of its operation, follows.

Fast Controller

```
1000 REM FAST CONTROLLER
1010 TK = 0 : LT = 35 : ST = 15 : LS = 15 : CD = WR : FAST
     = 1
1020 GOSUB 490 : GOSUB 610
1030 GOSUB 490 : GOSUB 610 : IF PEEK (TRK ) = LT
     THEN 1050
1040 TK = PEEK (TRK ) : ST = PEEK (SCT ) : GOTO 1020
1050 HOME : PRINT "COPYDONE" : END
```

Line Explanation

1000 - *identifies the controller*

1010 - *initializes variables:* TK= 0, ST= 15 - sets the starting sector to be copied at track 0, sector 0. LT= 35, LS=15 sets the last sector to be copied at track 34, sector 0. CD= WR sets the command code to write. FAST= 1 tells the sector editor subroutine that the sectors are stored in memory in a decreasing order.

1020 - *the read routine.* Begin by toggling read/write to read, read a chunk of sectors

1030 - *The write routine.* Begin by toggling read/write to write, write a chunk of sectors. If the last track was written then go to exit routine.

1040 - *update track and sector* number for next read and go to read routine

1050 - *The exit routine.* Clear the screen, print ending message and exit to Applesoft

How About a NewSwap?

By adding a couple of GOSUB's to The Fast Controller you can derive The NewSwap controller which is a controller that reads with one RWTS and writes with the normal RWTS. This controller is very handy and is used frequently by articles which appear in Hardcore COMPUTIST.

NewSwap Controller

```
1000 REM NEW SWAP CONTROLLER
1010 TK = 0 : LT = 35 : ST = 15 : LS = 15 : CD = WR : FAST
     = 1
1020 GOSUB 360 : GOSUB 490 : GOSUB 610
1030 GOSUB 360 : GOSUB 490 : GOSUB 610 : IF PEEK
     (TRK ) = LT THEN 1050
1040 TK = PEEK (TRK ) : ST = PEEK (SCT ) : GOTO 1020
1050 HOME : PRINT "COPYDONE" : END
10010 PRINT CHR$ (4 ) "BLOAD^RWTS.XXX,A$1900"
```

Note that the filename of the BLOAD command in line 10010 will need to be changed to the name you've given the RWTS of the protected disk.

Saving the Controller

It is recommended that you have easily accessible copies of both of these controllers because controllers printed for other disks will be very similar to these controllers. The most efficient controller/Super IOB relationship is one which makes use of 'CSaver' from Hardcore COMPUTIST No. 13.

Closing Notes

Every controller printed by Hardcore COMPUTIST, so far, will work with Super IOB v1.5. However, controllers printed in the future may not work with older versions of Super IOB.

Now, go out there and break some disks!

Table 1, Variable usage...........page22
Super IOB basic program..........page23

Super IOB v1.5 Hexdump

```
0300: A9 03 A0 0A 20 D9 03 B0   $BD35
0308: 16 60 01 60 01 00 00 00   $9CF5
0310: 1B 03 00 27 00 00 00 00   $4320
0318: 00 60 01 00 01 EF D8 AD   $55A7
0320: 17 03 4A 4A 4A 4A AA 4C   $B42B
0328: 12 D4 A9 00 A2 00 4C A0   $8038
0330: B9 A6 DE 4C 12 D4 68 A8   $6E1C
0338: 68 A6 DF 9A 48 98 48 60   $FDD9
0340: A0 00 84 FC 84 FE B1 FC   $3777
0348: 48 B1 FE 91 FC 68 91 FE   $AAB9

0350: C8 D0 F3 E6 FD E6 FF C6   $921F
0358: E0 D0 EB 60 97 35 0F A9   $5F1E
0360: 00 85 28 A9 05 85 29 A9   $8FFA
0368: 10 85 24 AD 0E 03 20 DA   $41EA
0370: FD A9 1C 85 24 AD 0F 03   $B401
0378: 20 DA FD 20 00 03 CE 0F   $011B
0380: 03 10 08 A9 0F 8D 0F 03   $3519
0388: EE 0E 03 EE 13 03 AD 13   $387C
0390: 03 CD 5C 03 B0 C5 AD 0F   $EFCA
0398: 03 CD 5E 03 D0 C9 AD 0E   $18A0

03A0: 03 CD 5D 03 90 C1 B0 B3   $250A
```

Table 1

IOB Variable Usage Table

A - general temporary usage, scrambled by "Move S Phases" and "The Sector Editor."

A$ - holds message to pass to the user via "Center Message" and "Print Message And Wait" and is scrambled by "Toggle Read/Write."

A1,A2,A3,A4 - scrambled by "Altered Address Marks", "Altered Data Marks", "The Sector Editor", "Altered Ending Marks" and "Altered Data Checksum" these are READ from DATA statements and POKEd into the appropriate RWTS to change it.

B$ - altered only by "Configuration Time."

BF - **B**uffer **F**ull holds the status of the sector buffer and is set to 1 if the buffer is either full or empty and 0 if neither. Is changed only by "R/W SECTOR."

BUF - **BUF**fer constant holds the address where the RWTS is expecting to find the page number of the sector and is used by "Initial IOB Setup" and "R/W Sector". A PEEK(BUF) will return the current sector buffer page number.

CD - **C**omman**D** code is used by the controller, "Toggle Read/Write" and "R/W Sector" and holds the current RWTS command code (see RD, WR, and INIT).

CMD - **C**o**M**man**D** code constant holds the address where the RWTS is expecting to find the previously stated command code and is used by "R/W Sector". A POKE CMD,CD will change the IOB command.

D1 - **D**rive **1**, set during configuration to the drive number of the source drive, is used by "Toggle Read/Write."

D2 - **D**rive **2**, same as above except for target drive.

DOS - Disk Operating System specifies the number of sectors to read or write and is initialized to 16.

DRV - **DR**i**V**e constant holds the address where the RWTS is expecting to find the drive number of the drive to be accessed and is used by "Initial IOB Setup". A PEEK(DRV) will return the drive last accessed.

DV - current DriVe, used by "Initial IOB Setup", "Toggle Read/Write", and "Move S Phases", holds the drive number of the drive to be accessed next.

ERR - **ERR**or code is used by "Disk Error" to determine the error that has just occurred.

FAST - used by "The Sector Editor" to calculate the correct addresses of specified sectors if set to 0. The "The Sector Editor" then assumes that all sectors are stored in consecutive memory pages in an ascending order. If set to a 1, "The Sector Editor" routine assumes that the sectors are stored in a decreasing order (see the section describing the operation of "R/W A Range of Sectors").

INIT - **INIT**ialize command code. A CD=INIT will set the command code to format the diskette.

IO - **I**nput/**O**utput constant holds a 768 (set during configuration) and is CALLed by "R/W Sector" to induce the RWTS subroutine.

LS - **L**ast **S**ector number is used to tell "R/W A Range Quickly" what sector is the last sector to be read.

LT - **L**ast **T**rack number is used to tell "R/W A Range Quickly" the last track to be read or written.

MB - **M**aximum **B**uffer page holds the last page of memory for the sector buffer, is used by "R/W Sector", is initialized (during configuration) to 151 and should be changed to 130 only when a 13-sector disk is read or written.

OVL - **O**ld **V**o**L**ume constant. A PEEK(OVL) will return the volume number of the previously accessed diskette.

PH - current **PH**ase. If "MOVE S PHASES" is referenced (by the controller), this variable must contain the disk arms current phase number (PH=2*TK).

RD - **R**ea**D** command code. A CD=RD will set the command to read the disk.

S - Step is used to tell "Move S Phases" how many phases to step through (-120 to 120).

S1 - **S**lot **1** means to set to the slot number of the source drive during configuration and is used by "Toggle Read/Write".

S2 - Slot **2**. Same as above except for target drive.

SCT - **S**e**CT**or number constant holds the address where the RWTS is expecting to find the sector to be accessed and is used by "R/W Sector" to tell the RWTS which sector is to be read or written. A PEEK(SCT) will return the last accessed sector number.

SLT - **SL**o**T** number constant holds the address where the RWTS is expecting to find the slot number of the disk to be accessed next and is used by "Initial IOB Setup". A PEEK(SLT) will return the last accessed disk slot number.

SO - **S**l**O**t number is used by "Toggle Read/Write" and "Initial IOB Setup" and holds the slot number of the disk to be accessed next.

ST - **S**ec**T**or number is used by the controller to tell "R/W Sector" which sector number is to be read or written next and is also used to tell "R/W A Range Quickly" the starting sector to be read or written.

TK - **T**rac**K** number is used by the controller to tell "R/W Sector" which track is to be accessed next and is also used to tell "R/W A Range Quickly" the starting track to read or write.

TRK - **TR**ac**K** number constant holds the memory location where the RWTS is expecting to find the track to be accessed. A PEEK(TRK) will return the last accessed track number.

VL - **V**o**L**ume number is used by the controller to tell "Toggle Read/Write" (which passes it to "Initial IOB Setup") the volume number of the disk to be accessed next.

VL$ - altered only by "Format Disk".

VOL - **VOL**ume number constant holds the memory location where the RWTS is expecting to find the volume to be accessed. A PEEK(VOL) will return the volume number last used by the controller.

WR - **WR**ite command code. A CD=WR will set the command to write.

Super IOB v1.5 BASIC program

```
10 REM ************************
20 REM **    SUPER IOB 1.5     **
30 REM **    BY RAY DARRAH     **
40 REM ************************
50 REM SET HIMEM BELOW BUFFER AND SET LOMEM
     ABOVE THE BLOADED RWTS
60 LOMEM: 8448 : HIMEM: 9983 : GOTO 10010
70 REM INITIAL IOB SETUP
80 POKE BUF ,39 : POKE DRV ,DV : POKE VOL ,VL :
     POKE SLT ,SO * 16 : RETURN
90 REM R/W SECTOR
100 BF = 0 : POKE TRK ,TK : POKE SCT ,ST : POKE CMD
     ,CD : CALL IO : POKE BUF , PEEK (BUF ) + 1 :
     IF PEEK (BUF ) = > MB THEN BF = 1
110 RETURN
120 REM MOVE S PHASES
130 POKE 49289 + SO * 16 + DV ,0 : POKE 49289 + SO
     * 16 ,0 :A = PH - INT (PH / 4 ) * 4 : POKE 1144
     ,128 + A : POKE 811 ,128 + S + A : POKE 813 ,SO
     * 16 : CALL 810 : POKE 49288 + SO * 16 ,0 :PH
     = PH + S : IF PH < 0 THEN PH = 0
140 RETURN
150 REM 16 SECTOR RWTS ALTERATIONS
160 REM ALTERED ENDING MARKS
170 READ A1 ,A2 ,A3 ,A4 : POKE 47505 ,A1 : POKE
     47515 ,A2 : POKE 47413 ,A3 : POKE 47423 ,A4
     : RETURN
180 REM ALTERED ADDRESS MARKS
190 READ A1 ,A2 ,A3 : POKE 47445 ,A1 : POKE 47455
     ,A2 : POKE 47466 ,A3 : RETURN
200 REM ALTERED DATA MARKS
210 READ A1 ,A2 ,A3 : POKE 47335 ,A1 : POKE 47345
     ,A2 : POKE 47356 ,A3 : RETURN
220 REM NORMALIZER
230 POKE 47505 ,222 : POKE 47515 ,170 : POKE
     47413 ,222 : POKE 47423 ,170
240 POKE 47445 ,213 : POKE 47455 ,170 : POKE
     47466 ,150 : POKE 47335 ,213
250 POKE 47345 ,170 : POKE 47356 ,173 : POKE
     47360 ,0 : POKE 47498 ,183 : RETURN
260 REM IGNORE ADDRESS CHECKSUM
270 POKE 47498 ,0 : RETURN
280 REM ALTERED DATA CHECKSUM
290 READ A1 : POKE 47360 ,A1 : RETURN
300 REM THE SECTOR EDITOR
310 READ A$ : IF RIGHT$ (A$ ,7 ) < > "CHANGES" THEN
     310
320 FOR A = 1 TO VAL (A$ ) : READ A1 ,A2 ,A3 ,A4
330 IF A1 < T1 OR A1 > TK THEN NEXT : RETURN
340 POKE 9984 + (A1 - T1 ) * 4096 + ABS (FAST * 15
     - A2 ) * 256 + A3 ,A4 : NEXT : RETURN
350 REM SWAP RWTS AT $1900 WITH THE ONE AT
     $B800
360 POKE 253 ,25 : POKE 255 ,184 : POKE 224 ,8 :
     CALL 832 : RETURN
370 REM FORMAT DISK
380 A$ = "VOLUME^NUMBER^FOR^COPY^=>254" : HOME
     : GOSUB 450 : HTAB 32 : INPUT "" ;VL$ :VL =
     VAL (VL$ ) : IF VL$ = "" THEN VL = 254
390 IF VL > 255 OR VL < 0 THEN 380
400 POKE CMD , INIT :SO = S2 :DV = D2 :A$ =
     "INSERT^BLANK^DISK^IN^SLOT^" + STR$ (S2
     ) + ",^DRIVE^" + STR$ (D2 ) : GOSUB 470
410 GOSUB 80 : HOME :A$ = "FORMATTING" : FLASH
     : GOSUB 450 : NORMAL : CALL IO :VL = 0 :
     RETURN
420 REM PRINT TRACK & SECTOR#
430 VTAB 3 : HTAB 10 : PRINT "TRACK^$" MID$ (HX$
     ,TK * 2 + 1 ,2 ) "^^SECTOR^$" MID$ (HX$ ,ST
     * 2 + 1 ,2 ) "^^" : RETURN
440 REM CENTER MESSAGE
450 HTAB 21 - LEN (A$ ) / 2 : PRINT A$; : RETURN
460 REM PRINT MESSAGE AND WAIT
470 HOME : VTAB 11 : GOSUB 450 : VTAB 13 :A$ =
     "PRESS^ANY^KEY^TO^CONTINUE" : GOSUB 450
     : WAIT - 16384 ,128 : GET A$ : RETURN
480 REM TOGGLE READ/WRITE
490 CD = (CD = 1 ) + 1 : IF CD = RD THEN A$ =
     "INSERT^SOURCE^DISK." :SO = S1 :DV = D1 :
     GOTO 510
500 A$ = "INSERT^TARGET^DISK." :SO = S2 :DV = D2
510 IF D1 = D2 AND S1 = S2 THEN GOSUB 470 : HOME
520 VTAB 1 : HTAB 1 : PRINT SPC( 39 ); : FLASH :A$
     = "READING" : IF CD = WR THEN A$ = "WRITING"
530 GOSUB 450 : NORMAL : GOTO 80
540 REM ONERR IGNORE UNREADABLE SECTORS
550 CALL 822 :ERR = PEEK (222 )
560 IF ERR = 255 OR ERR = 254 OR CD < > RD THEN 10230
570 IF ERR > 15 THEN POKE 216 ,0 : RESUME
580 PRINT CHR$ (7 ); : POKE BUF , PEEK (BUF ) + 1
     : IF PEEK (BUF ) = > MB THEN BF = 1
590 RETURN
600 REM R/W A RANGE QUICKLY
610 PR# 0 : IN# 0 : POKE 860 ,MB : POKE 861 ,LT :
     POKE 862 ,LS
620 POKE CMD ,CD : POKE TRK ,TK : POKE SCT ,ST :
     GOSUB 430 : CALL 863 : CALL 1002 : RETURN
10000 REM CONFIGURATION TIME
10010 REM BLOAD RWTS HERE
10020 IF PEEK (768 ) * PEEK (769 ) = 507 THEN 10060
10030 HOME :A$ = "*^SUPER^IOB^*" : GOSUB 450 :
     PRINT : PRINT :A$ = "CREATED^BY^RAY^DAR
     RAH" : GOSUB 450
10040 VTAB 10 :A$ = "INSERT^SUPER^IOB^DISK" :
     GOSUB 450 : PRINT : PRINT : PRINT :A$ =
     "PRESS^ANY^KEY^TO^CONTINUE" : GOSUB 450
     : WAIT - 16384 ,128 : GET A$
10050 PRINT : PRINT CHR$ (4 ) "BLOAD^IOB.OBJ
     0,A$300"
10060 TK = ST = VL = CD = DV = SO :RD = 1 :WR = 2 :INIT
     = 4 : ONERR GOTO 10220
10070 IO = 768 :SLT = 779 :DRV = 780 :VOL = 781
     :TRK = 782 :SCT = 783 :BUF = 787 :CMD = 790
     :OVL = 792
10080 HOME :DOS = 16 :MB = 151 :HX$ =
     "000102030405060708090A0B0C0D0E0F101
     112131415161718191A1B1C1D1E1F202122"
10090 VTAB 8 : PRINT :A$ = "ORIGINAL" :S2 = 6 :D2
     = 1 : GOSUB 10140 :S1 = S2 :D1 = D2
10100 PRINT : PRINT : PRINT :D2 = (D2 = 1 ) + 1 :A$
     = "DUPLICATE" : GOSUB 10140
10110 A$ = "FORMAT^BACK^UP^FIRST?^N" + CHR$ (8
     ) : HOME : VTAB 12 : GOSUB 450 : GET A$ : IF
     A$ = "Y" THEN GOSUB 380
10120 HOME :A$ = "INSERT^DISKS^IN^PROPER^DRIV
     ES." : GOSUB 470 : HOME : GOTO 1000
10130 REM GET SLOT AND DRIVE#
10140 GOSUB 450 : PRINT : PRINT : PRINT TAB( 10
     ) "SLOT=>" S2 SPC( 8 ) "DRIVE=>" D2;
10150 HTAB 16 :B$ = "7" : GOSUB 10180 :S2 = VAL
     (A$ )
10160 HTAB 32 :B$ = "2" : GOSUB 10180 :D2 = VAL
     (A$ ) : RETURN
10170 REM GET A KEY
10180 GET A$ : IF (A$ < "1" OR A$ > B$ ) AND A$ <
     > CHR$ (13 ) THEN 10180
10190 IF A$ = CHR$ (13 ) THEN A$ = CHR$ ( PEEK (
     PEEK (40 ) + PEEK (41 ) * 256 + PEEK (36 ) )
     - 128 )
10200 PRINT A$; : RETURN
10210 REM DISK ERROR
10220 ERR = PEEK (222 ) : IF ERR > 15 AND ERR < 254
     THEN POKE 216 ,0 : CALL 822 : RESUME
10230 IF ERR = 254 THEN PRINT "TYPE^AGAIN^PLEA
     SE:" : PRINT : RESUME
10240 IF ERR = 255 THEN STOP
10250 IF ERR = 0 THEN A$ = "INITIALIZATION^ER
     ROR"
10260 IF ERR = 1 THEN A$ = "WRITE^PROTECTED"
10270 IF ERR = 2 THEN A$ = "VOLUME^MISMATCH^ER
     ROR"
10280 IF ERR = 4 THEN A$ = "DRIVE^ERROR"
10290 IF ERR = 8 THEN A$ = "READ^ERROR"
10300 VTAB 12 : GOSUB 450 : PRINT CHR$ (7 ) : END
```

Super IOB v1.5 Checksums

10	- $BADD	480	- $88F2
20	- $9B13	490	- $58F6
30	- $4D3B	500	- $306E
40	- $AD92	510	- $799A
50	- $C899	520	- $C998
60	- $1FBA	530	- $F2E1
70	- $0061	540	- $3215
80	- $835F	550	- $B642
90	- $E171	560	- $F18A
100	- $AD0E	570	- $65E6
110	- $57B6	580	- $BC9F
120	- $8472	590	- $8F93
130	- $617E	600	- $70D2
140	- $0F1F	610	- $E0C1
150	- $F1B3	620	- $DAE1
160	- $C59A	10000	- $1AA6
170	- $24E7	10010	- $69F9
180	- $3991	10020	- $5B71
190	- $8D19	10030	- $B9FC
200	- $87A6	10040	- $5019
210	- $F20A	10050	- $3573
220	- $7021	10060	- $4306
230	- $B0F4	10070	- $47AE
240	- $B3A7	10080	- $FDD7
250	- $7FFD	10090	- $EBD1
260	- $51AB	10100	- $9723
270	- $E236	10110	- $93ED
280	- $2EE2	10120	- $6332
290	- $CBCA	10130	- $4518
300	- $BAE5	10140	- $DAC7
310	- $16EC	10150	- $D05F
320	- $4E65	10160	- $7204
330	- $7F0F	10170	- $8365
340	- $098C	10180	- $466A
350	- $3C7E	10190	- $5C6A
360	- $9882	10200	- $B6A0
370	- $32F9	10210	- $F765
380	- $D61F	10220	- $D6F1
390	- $D95E	10230	- $BA69
400	- $C556	10240	- $764A
410	- $D95A	10250	- $B0D6
420	- $DACB	10260	- $2A7A
430	- $3F10	10270	- $9243
440	- $F817	10280	- $F40E
450	- $7C8F	10290	- $87C8
460	- $D3B4	10300	- $19BB
470	- $8288		

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.

Archon II: The Adept	Electronic Arts
Apple Business Graphics	Apple Computer
Flight Simulator II	Sub Logic
Apple LOGO II	Apple Computer
Jane	Arktronics
Bookends	Sensible Software
Visiblend	Microlab
Sundog	FTL Games
Lifesaver	Microlab
Catalyst	Quark, Inc.
Gutenburg	
Jr. & Sr.	Micromation LTD
Prime Plotter	Primesoft Corp.
Robot Odyssey	The Learning Company
Zardax	Computer Solutions
The Handlers	Silicon Valley Systems
Milliken Math Series (NEW)	Milliken Publishing
The Apple's Core: Parts 1-3	The Professor
Where in the World is Carmen San Diego?	Broderbund
King's Quest	Sierra On-line
Hayes Terminal Program	Hayes
Fun Bunch	Unicorn
Murder on the Zinder neuf	Electronic Arts

A.P.T.'s

Serpentine Broderbund Software, Inc.

To get a free man, press and hold in ESCAPE. Then just press ! (shift 1). It appears that you can acquire as many free men as you like.

Snoggle

To get 3 free men in Snoggle, you must first commit suicide. While you are dying, press SHIFT (hold it in), CONTROL, and M. Only do this while you are dying.

Night Mission (A2-PB1) subLOGIC

To take a break without losing you game, press ! (shift 1). That's all!

Miner 2049'er MicroLab

To start on any of the levels, 1-8: boot up the disk. When you are asked "1 or 2 players?", press SHIFT and then the number of the desired level.

Labyrinth Broderbund Software, Inc.

To get extra men in this game press and hold in ESCAPE. Then press the K,A,Y and 9 (ESCAPE K-A-Y-9).

To choose the level you want (1-8) press ESCAPE (hold it in), press K, A, Y and then the number of the level (ESCAPE K-A-Y-#).

Cannonball Blitz Sierra On-Line

To make the second screen less difficult, when you have completed the first screen, *jump*.

Minit Man *Penguin Software*

If you have a broken copy of Minit Man (an unprotected binary file) this APT will allow you to have as many men as you want.

Type in this small Applesoft program, SAVE it, then RUN.

```
1Ø PRINT CHR$(4); "BLOAD MINITMAN, A5Ø43"
2Ø INPUT "HOW MANY MEN? "A
3Ø POKE 16272,A
4Ø CALL 5Ø43
5Ø END
```

Note: I have found that the more lives you start with, the harder the game becomes.

Contributed by David A. Martin.

Cytron Masters

One of the nice features about Cytron Masters is the user's ability to create his own set-up. However, I have found (through hours of playing and experimentation) that the best initial set-up is the one you start with. If you get exotic in your set-ups, you won't have much power to start with. This is most detrimental when playing on the Grand Master level.

For your first few times out, play on the Beginner Level using basic set-up. Then locate your beam point at the top of the screen. Make a vertical wall of three bunkers and, behind them, a vertical wall of shooters with a Commander between two shooters. Order the Commander to advance.

Next, locate your beam point at the bottom of the screen. Make another group of bunkers, shooters and Commander, then order him to advance.

When they are over power stations, order them North or South so as to take over as many power stations as possible. (Note: This will not work on higher levels because the computer fires missiles more often.) Once you have all or most of the power stations, *it's in the bag*.

Contributed by David A. Martin

...bugs

Hardcore Computist No. 19:

The last paragraph of the third column of page 10 should read:

Next, solder the wire connected to pin 21 to the center contact of the SPDT switch and solder the other two wires to the edge contacts of the switch. When you have done this, the schematic of it should look something like this:

Hardcore Computist No. 21:

PROSHADOW

The hexdump on page 16 was incorrectly printed. To correct it, add these lines:

```
400A: EF 40
4010: F0 40
4082: AF D0
```

thanks to Randy Sickle Bower

MACRO SYSTEM

Are you tired of having to type in the same subroutine for each new program? Don't you wish you could zip through those boring menus in a flash? With The Macro System and two key-presses, you can do all of this and more.

by Tim Lewis

In order to get the Macro System up and running, just follow these steps:

1) Key in the hexdump at the end of this article using the procedure detailed on page 2 of this magazine.
2) When you are finished, save the program to disk

BSAVE MACRO,A$92ØØ,L$1Ø7

3) Key in the Applesoft portion of The Macro System using the procedure detailed on page 2 of this magazine and save it with:

SAVE MACRO EDITOR

Note that The Macro System was written for the Apple //e. Because of this, you may wish to perform the "Apple][Modification" outlined under the section of the same name.

Using MACRO

The Macro System consists of two parts: a machine language program which resides in high memory and an Applesoft program which allows the configuration of the machine language portion.

The machine language portion will be referred to hence forth as MACRO. To get MACRO up and running, type the following:

BRUN MACRO
HIMEM: 37375

The HIMEM statement is necessary to protect MACRO from the ravages of Applesoft strings.

A macro is a string of user defined characters. By pressing the TAB key (or CTRL I) followed by a character in the ASCII range 33 through 9Ø (! - Z), the computer will act as if you have sequentially keyed all the characters in the macro associated with that character.

Be sure NOT to hold down the TAB key (or CTRL I) while pressing the other key.

If MACRO should get disconnected (i.e. pressing RESET or doing an IN#Ø), you may reconnect it with:

CALL37376

Using MACRO EDITOR

The Applesoft portion of The Macro System is called MACRO EDITOR and is used to define or edit the macro strings contained in MACRO. As MACRO stands right now, there are no macros defined therefore you must install MACRO and use MACRO EDITOR to define some macros.

When MACRO EDITOR is first RUN, it retrieves whatever macros are in memory and then gives you a menu of 5 options:

D)isplay the list of macros
E)dit some macros
S)tore macros in memory
P)ut macros in memory on disk
Q)uit this program

For those of you curious about why the BYTES USED listing at the bottom of the screen says 58 when you haven't entered any yet, it is because each blank macro takes up one byte (it is defined as one space).

Display List of Macros

This option is activated by pressing "D" and prints an alphabetical list of all the macros in memory. If you answer "Y" to the "Send list to printer?" prompt, then MACRO EDITOR will try to print the list to a printer in slot 1. You may pause the screen listing by pressing CTRL S.

Edit Some Macros

Pressing "E" will invoke this option which allows you to add or edit one or many macros. When keying in a macro, any control key may be entered by preceding it with a CTRL O.

Store Macros in Memory

This option is activated by pressing "S" and stores all the macros in memory where MACRO can get to them. If you don't do this option, then only MACRO EDITOR knows about the macros you've defined or edited.

Put Macros in Memory on Disk

Pressing "P" will invoke this option which BSAVES the macros stored in memory (via option "S") as another MACRO file with the filename you wish. When this option is called, a special disk access routine is invoked. To get a CATALOG of the current disk, press

RETURN when asked for a filename. To save the macro file on another disk drive simply include the parameters at the end of the filename.

For example, if you typed

MACRO.GRAPHIC GRAB,D2

as a filename, then a file called "MACRO.GRAPHIC GRAB" would be saved on drive 2.

Quit This Program

This option is activated by pressing "Q" and exits to Applesoft. If you want to try out your macros, then press "S" to store them in memory and "Q" to exit to Applesoft. Typing "RUN" will start up the program again and you can continue editing where you left off. If you try this option and MACRO doesn't seem to be functioning, try a

CALL37376

Programming Notes

Basically this program interrupts the normal input routines by changing $38-$39 to point to a routine inside of MACRO. If there is no macro in progress, it checks for the TAB key. If the TAB key is pressed in this circumstance, it gets another key and sets the Macro flag.

If there is a macro being used, the program gets the next byte in the macro and returns it as if it were the input.

Apple][Modification

For those of you who have the Apple][,][Plus, you will type the following after saving the hexdump:

```
BLOAD MACRO
CALL -151
921A: 9B
BSAVE MACRO,A$9200,L$107
```

This modifies MACRO so that it uses the ESCape key instead of the non existant TAB key on the Apple][series. You may substitute any key you want for the "9B" in the commands above. Just be sure that it is the key value plus 128.

This modifies MACRO so that it uses the ESCape key instead of the non existant TAB key on the Apple][series. You may substitute any key you want for the "9B" in the commands above. Just be sure that it is the key value plus 128.

Conclusion

The Macro System has been very useful to me, especially in typing in routines. I certainly hope you find useful as well. In all, have fun!

Macro Editor BASIC program..........page 28

MACRO Source Code

```
                       **********************
                       *       MACRO        *
                       *   BY TIM LEWIS     *
                       *     06/20/85       *
                       **********************
               .OR $9200   Put it just below standard himem
               .TF MACRO
FD1B- KEYIN    .EQ $FD1B   Get a key from the keyboard.
0024- CH       .EQ $24     Horizontal cursor position
0018- PTR      .EQ $18     Pointer to current macro
0028- BASL     .EQ $28     Address of first char posn on text screen
0038- INHOOK   .EQ $38     Input hook
03EA- DOS.HOOK .EQ $3EA    Hook this up to DOS
*----------------------------------------------------------------------*
*                   Init hooks MACRO to the input hooks                 *
*----------------------------------------------------------------------*
9200: A9 0B    INIT     LDA #INPUT
9202: A0 92             LDY /INPUT
9204: 85 38             STA INHOOK
9206: 84 39             STY INHOOK+1
9208: 4C EA 03          JMP DOS.HOOK
*----------------------------------------------------------------------*
*           Input either gets a key or continues with the macro         *
*----------------------------------------------------------------------*
920B: 2C 56 92 INPUT    BIT MACROFLG Check to see if there is a macro
920E: 10 35             BPL MACRO    if so then get character from it
9210: 8D 58 92          STA CURCHAR  Save character under cursor
9213: AD 58 92 MACKEY   LDA CURCHAR
9216: 20 1B FD          JSR KEYIN    otherwise, get a key
9219: C9 89             CMP #$89     is it TAB? (CTRL-I)
921B: D0 38             BNE RTS.1    no, we're finished
921D: AD 58 92          LDA CURCHAR  Restore character under cursor
9220: 20 1B FD          JSR KEYIN    get the macro key
9223: C9 A1             CMP #'!+$80  if less than a '!'
9225: 90 EC             BCC MACKEY
9227: C9 DB             CMP #'[+$80  or past a 'Z'
9229: B0 E8             BCS MACKEY   then forget it.
922B: 38                SEC          take $A1 off of key
922C: E9 A1             SBC #'!+$80
922E: 0A                ASL          and double it for pointer
922F: A8                TAY
9230: B9 59 92          LDA TABLE,Y
9233: 85 18             STA PTR
9235: C8                INY
9236: B9 59 92          LDA TABLE,Y
9239: 85 19             STA PTR+1
923B: A0 00             LDY #0
923D: 8C 56 92          STY MACROFLG Tell that there is a macro
9240: 8C 57 92          STY POINTER  and that we're at the first letter
9243: F0 C6             BEQ INPUT    go get the first key
9245: AC 57 92 MACRO    LDY POINTER  get the pointer
9248: B1 18             LDA (PTR),Y  get the character
924A: 30 05             BMI CHANGE   if hi-bit, we're not at end.
924C: 09 80             ORA #$80     otherwise set hi-bit
924E: 8D 56 92          STA MACROFLG and clear macro status
9251: C8       CHANGE   INY          increment pointer
9252: 8C 57 92          STY POINTER  NOTE: this is useless if finished
9255: 60       RTS.1    RTS          with macro.
9256: 80       MACROFLG .HS 80       whether there's a macro
9257: 00       POINTER  .HS 00       how many char's into macro
9258: 00       CURCHAR  .HS 00       Character under cursor

9259: CD 92    TABLE    .DA M.0      beginning of data pointers
925B: CE 92             .DA M.1
925D: CF 92             .DA M.2
925F: D0 92             .DA M.3
9261: D1 92             .DA M.4
9263: D2 92             .DA M.5
9265: D3 92             .DA M.6
9267: D4 92             .DA M.7
```

```
9269: D5 92              .DA M.8
926B: D6 92              .DA M.9
926D: D7 92              .DA M.1Ø
926F: D8 92              .DA M.11
9271: D9 92              .DA M.12
9273: DA 92              .DA M.13
9275: DB 92              .DA M.14
9277: DC 92              .DA M.15
9279: DD 92              .DA M.16
927B: DE 92              .DA M.17
927D: DF 92              .DA M.18
927F: EØ 92              .DA M.19
9281: E1 92              .DA M.2Ø
9283: E2 92              .DA M.21
9285: E3 92              .DA M.22
9287: E4 92              .DA M.23
9289: E5 92              .DA M.24
928B: E6 92              .DA M.25
928D: E7 92              .DA M.26
928F: E8 92              .DA M.27
9291: E9 92              .DA M.28
9293: EA 92              .DA M.29
9295: EB 92              .DA M.3Ø
9297: EC 92              .DA M.31
9299: ED 92              .DA M.A
929B: EE 92              .DA M.B
929D: EF 92              .DA M.C
929F: FØ 92              .DA M.D
92A1: F1 92              .DA M.E
92A3: F2 92              .DA M.F
92A5: F3 92              .DA M.G
92A7: F4 92              .DA M.H
92A9: F5 92              .DA M.I
92AB: F6 92              .DA M.J
92AD: F7 92              .DA M.K
92AF: F8 92              .DA M.L
92B1: F9 92              .DA M.M
92B3: FA 92              .DA M.N
92B5: FB 92              .DA M.O
92B7: FC 92              .DA M.P
92B9: FD 92              .DA M.Q
92BB: FE 92              .DA M.R
92BD: FF 92              .DA M.S
92BF: ØØ 93              .DA M.T
92C1: Ø1 93              .DA M.U
92C5: Ø3 93              .DA M.W
92C7: Ø4 93              .DA M.X
92C9: Ø5 93              .DA M.Y
92CB: Ø6 93              .DA M.Z

92CD: 2Ø        M.Ø      .HS 2Ø      actual data tables.
92CE: 2Ø        M.1      .HS 2Ø      notice that a blank macro
92CF: 2Ø        M.2      .HS 2Ø      is actually a one space.
92DØ: 2Ø        M.3      .HS 2Ø
92D1: 2Ø        M.4      .HS 2Ø
92D2: 2Ø        M.5      .HS 2Ø
92D3: 2Ø        M.6      .HS 2Ø
92D4: 2Ø        M.7      .HS 2Ø
92D5: 2Ø        M.8      .HS 2Ø
92D6: 2Ø        M.9      .HS 2Ø
92D7: 2Ø        M.1Ø     .HS 2Ø
92D8: 2Ø        M.11     .HS 2Ø
92D9: 2Ø        M.12     .HS 2Ø
92DA: 2Ø        M.13     .HS 2Ø
92DB: 2Ø        M.14     .HS 2Ø
92DC: 2Ø        M.15     .HS 2Ø
92DD: 2Ø        M.16     .HS 2Ø
92DE: 2Ø        M.17     .HS 2Ø
92DF: 2Ø        M.18     .HS 2Ø
92EØ: 2Ø        M.19     .HS 2Ø
92E1: 2Ø        M.2Ø     .HS 2Ø
92E2: 2Ø        M.21     .HS 2Ø
92E3: 2Ø        M.22     .HS 2Ø
92E4: 2Ø        M.23     .HS 2Ø
92E5: 2Ø        M.24     .HS 2Ø
92E6: 2Ø        M.25     .HS 2Ø
92E7: 2Ø        M.26     .HS 2Ø
92E8: 2Ø        M.27     .HS 2Ø
92E9: 2Ø        M.28     .HS 2Ø
92EA: 2Ø        M.29     .HS 2Ø
92EB: 2Ø        M.3Ø     .HS 2Ø
92EC: 2Ø        M.31     .HS 2Ø
92ED: 2Ø        M.A      .HS 2Ø
92EE: 2Ø        M.B      .HS 2Ø
92EF: 2Ø        M.C      .HS 2Ø
92FØ: 2Ø        M.D      .HS 2Ø
92F1: 2Ø        M.E      .HS 2Ø
92F2: 2Ø        M.F      .HS 2Ø
92F3: 2Ø        M.G      .HS 2Ø
92F4: 2Ø        M.H      .HS 2Ø
92F5: 2Ø        M.I      .HS 2Ø
92F6: 2Ø        M.J      .HS 2Ø
92F7: 2Ø        M.K      .HS 2Ø
92F8: 2Ø        M.L      .HS 2Ø
92F9: 2Ø        M.M      .HS 2Ø
92FA: 2Ø        M.N      .HS 2Ø
92FB: 2Ø        M.O      .HS 2Ø
92FC: 2Ø        M.P      .HS 2Ø
92FD: 2Ø        M.Q      .HS 2Ø
92FE: 2Ø        M.R      .HS 2Ø
92FF: 2Ø        M.S      .HS 2Ø
93ØØ: 2Ø        M.T      .HS 2Ø
93Ø1: 2Ø        M.U      .HS 2Ø
93Ø2: 2Ø        M.V      .HS 2Ø
93Ø3: 2Ø        M.W      .HS 2Ø
93Ø4: 2Ø        M.X      .HS 2Ø
93Ø5: 2Ø        M.Y      .HS 2Ø
93Ø6: 2Ø        M.Z      .HS 2Ø
```

Macro source code checksums

```
92ØØ: A9 ØB AØ 92 85 38 84 39   $56E8
92Ø8: 4C EA Ø3 2C 56 92 1Ø 35   $ØD2B
921Ø: 8D 58 92 AD 58 92 2Ø 1B   $8492
9218: FD C9 89 DØ 38 AD 58 92   $EACD
922Ø: 2Ø 1B FD C9 A1 9Ø EC C9   $D512
9228: DB BØ E8 38 E9 A1 ØA A8   $E416
923Ø: B9 59 92 85 18 C8 B9 59   $ØAB8
9238: 92 85 19 AØ ØØ 8C 56 92   $89A4
924Ø: 8C 57 92 FØ C6 AC 57 92   $3FBF
9248: B1 18 3Ø Ø5 Ø9 8Ø 8D 56   $4843

925Ø: 92 C8 8C 57 92 6Ø 8Ø ØØ   $695C
9258: ØØ CD 92 CE 92 CF 92 DØ   $22CA
926Ø: 92 D1 92 D2 92 D3 92 D4   $283Ø
9268: 92 D5 92 D6 92 D7 92 D8   $5A6E
927Ø: 92 D9 92 DA 92 DB 92 DC   $7ØD4
9278: 92 DD 92 DE 92 DF 92 EØ   $12D2
928Ø: 92 E1 92 E2 92 E3 92 E4   $D8A8
9288: 92 E5 92 E6 92 E7 92 E8   $6A76
929Ø: 92 E9 92 EA 92 EB 92 EC   $8Ø4C
9298: 92 ED 92 EE 92 EF 92 FØ   $Ø2DA

92AØ: 92 F1 92 F2 92 F3 92 F4   $882Ø
92A8: 92 F5 92 F6 92 F7 92 F8   $7A7E
92BØ: 92 F9 92 FA 92 FB 92 FC   $DØC4
92B8: 92 FD 92 FE 92 FF 92 ØØ   $F2A2
92CØ: 93 Ø1 93 Ø2 93 Ø3 93 Ø4   $1Ø98
92C8: 93 Ø5 93 Ø6 93 2Ø 2Ø 2Ø   $8ØFC
92DØ: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø   $ØØBC
92D8: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø   $8ØFC
92EØ: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø   $ØØBC
92E8: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø   $8ØFC

92FØ: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø   $ØØBC
92F8: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø   $8ØFC
93ØØ: 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø      $2C78
```

core's Macro Editor: BASIC program

```
10 REM ***********************
20 REM **                   **
30 REM ** MACRO             **
40 REM **        EDITOR ][  **
50 REM **                   **
60 REM **   BY TIM LEWIS    **
70 REM **                   **
80 REM ***********************
90 REM
100 HIMEM: 37375 : DEF FN HI(X ) = INT (X / 256 )
    : DEF FN LO(X ) = X - INT (X / 256 ) * 256
110 PR# Ø : CALL 1ØØ2 :PTR = 37465 :TXT = 37581
    :FIN = 38399 :MAX = FIN - TXT + 1
120 TEXT : HOME : IF PEEK (37376 ) < > 169 THEN
    PRINT CHR$ (7 ) "MACRO^ SYSTEM^ NOT^
    INSTALLED!" CHR$ (7 ) : STOP
130 VTAB 1Ø : PRINT "RETRIEVING^MACROS^NOW.^
    PLEASE^WAIT..."
140 DIM ST$(57 ) :NUM = Ø :CURR = TXT :K$ = "DESPQ"
150 FOR NUM = Ø TO 57 : VTAB 1Ø : HTAB 38 : PRINT NUM
160 CURR = PEEK (PTR + 2 * NUM ) + 256 * PEEK (PTR
    + 2 * NUM + 1 )
170 P = PEEK (CURR ) :ST$(NUM ) = ST$(NUM ) + CHR$
    (P ) :CURR = CURR + 1 : IF P > 127 THEN 17Ø
180 NEXT
190 BYT = Ø : FOR N = Ø TO 57 :BYT = BYT + LEN (ST$(N
    ) ) : NEXT
200 HOME : HTAB 8 : PRINT "MACRO^EDITOR^BY^TIM^
    LEWIS"
210 POKE 34 ,4 : HOME : VTAB 7 : PRINT "COMMANDS^
    AVAILABLE^FOR^MACRO^EDITOR:" : PRINT
220 PRINT : PRINT "D)ISPLAY^CURRENT^LIST^OF^
    MACROS"
230 PRINT "E)DIT^SOME^OF^THE^MACROS"
240 PRINT "S)TORE^MACROS^IN^MEMORY"
250 PRINT "P)UT^MACROS^IN^MEMORY^ON^DISK"
260 PRINT "Q)UIT^THE^PROGRAM"
270 VTAB 23 : PRINT "MAX^BYTES:" ;MAX; "^^BYTES^
    USED:" ;BYT :FLAG = 1
280 VTAB 16 : HTAB 1 : PRINT "YOUR^CHOICE:" ; :
    GET A$
290 FOR A = 1 TO 5 : IF A$ < > MID$ (K$ ,A ,1 ) THEN
    NEXT : GOTO 28Ø
300 ON A GOTO 47Ø ,56Ø ,69Ø ,78Ø
310 REM QUIT PROGRAM
320 TEXT : HOME : END
330 REM INPUT A MACRO
340 H1 = H : IN$ = "" :L = Ø :CTRL = Ø : VTAB 15 : HTAB
    1
350 PRINT "TYPE^ <CTRL>O^ BEFORE^ CONTROL^
    CHARACTERS"
360 VTAB V : HTAB H1 + 1 : GET A$ : IF A$ = CHR$ (15
    ) AND CTRL = Ø THEN CTRL = 1 : GOTO 36Ø
370 IF CTRL THEN CTRL = Ø : GOTO 42Ø
380 IF A$ = CHR$ (27 ) OR A$ = CHR$ (13 ) THEN CALL
    64578 : RETURN
390 IF A$ = CHR$ (8 ) AND H1 > H THEN L = L - 1 :V
    = V - 1 * (H1 = Ø ) :H1 = H1 - 1 + 4Ø * (H1 = Ø
    ) : IN$ = MID$ (IN$ ,1 ,L ) : GOTO 36Ø
400 IF A$ = CHR$ (21 ) THEN A$ = CHR$ ( SCRN( H1
    ,2 * (V - 1 ) ) + 16 * SCRN( H1 ,2 * V - 1 ) )
    : GOTO 42Ø
410 IF A$ < "^" THEN 36Ø
420 L = L + 1 : IF L > ML THEN L = L - 1 : GOTO 36Ø
430 V = V + 1 * (H1 = 39 ) :H1 = H1 + 1 - 4Ø * (H1 = 39 )
440 IN$ = IN$ + A$ : IF A$ < "^" THEN INVERSE :A$
    = CHR$ ( ASC (A$ ) + 64 )
450 PRINT A$; : NORMAL : GOTO 36Ø
460 REM DISPLAY LIST OF MACROS
470 HOME : VTAB 6 : PRINT "SEND^LIST^TO^THE^
    PRINTER?^(Y/N):^N" CHR$ (8 );
480 GET A$ : PRINT A$ : ON A$ < > "Y" AND A$ < > "N"
    AND A$ < > CHR$ (13 ) GOTO 48Ø : IF A$ = "Y"
    THEN PRINT CHR$ (4 ) "PR#1"
490 VTAB 8 : FOR N = Ø TO 57 : HTAB 1 : INVERSE :
    PRINT CHR$ (N + 33 ); : NORMAL : PRINT ">^" ;
500 GOSUB 52Ø : PRINT : PRINT : NEXT : POKE - 16368
    ,Ø : GET A$ : PRINT : PRINT CHR$ (4 ) "PR#Ø"
    : TEXT : GOTO 2ØØ
510 REM PRINT A MACRO
520 FOR M = 1 TO LEN (ST$(N ) ) :P$ = MID$ (ST$(N
    ) ,M ,1 )
530 IF P$ < "^" OR (P$ > CHR$ (127 ) AND P$ < CHR$
    (16Ø ) ) THEN INVERSE :P$ = CHR$ ( ASC (P$
    ) + 64 )
540 PRINT P$; : NORMAL : NEXT : RETURN
550 REM EDIT MACROS
560 HOME : VTAB 5 : PRINT "EDIT^SOME^MACROS"
570 VTAB 8 : HTAB 1 : PRINT "EDIT^THE^MACRO^FOR^
    WHAT^CHARACTER?^" ; : GET A$ : IF A$ < "!"
    OR A$ > "Z" THEN 21Ø
580 PRINT A$; :N = ASC (A$ ) - 33 :L1 = LEN (ST$(N
    ) )
590 VTAB 11 : HTAB 1 : GOSUB 52Ø :V = 11 :H = Ø :ML
    = FIN - (TXT + BYT ) + L1
600 ML = 119 * (ML > 119 ) + ML * (ML < = 119 ) : GOSUB
    34Ø : IF A$ = CHR$ (27 ) THEN 21Ø
610 IF IN$ = "" THEN IN$ = "^"
620 VTAB 14 : HTAB 1 : CALL - 958
630 IF BYT + 1 + LEN ( IN$ ) - L1 > MAX THEN VTAB 15
    : PRINT "MACRO^TOO^LONG" : GOTO 65Ø
640 BYT = BYT + LEN ( IN$ ) :ST$(N ) = IN$
650 VTAB 2Ø : HTAB 1 : PRINT "EDIT^ANOTHER^MACRO?^
    (Y/N):^N" CHR$ (8 );
660 GET A$ : ON A$ < > "Y" AND A$ < > "N" AND A$ <
    > CHR$ (13 ) GOTO 66Ø : IF A$ = "Y" THEN 56Ø
670 TEXT : GOTO 19Ø
680 REM STORE MACROS IN MEMORY
690 HOME : VTAB 1Ø : PRINT "STORING^MACROS^NOW.^
    PLEASE^WAIT..."
700 CURR = TXT
710 FOR N = Ø TO 57 : VTAB 1Ø : HTAB 35 : PRINT N
720 POKE PTR + N * 2 , FN LO(CURR ) : POKE PTR + N
    * 2 + 1 , FN HI(CURR )
730 IF LEN (ST$(N ) ) - 1 = Ø THEN 76Ø
740 FOR M = 1 TO LEN (ST$(N ) ) - 1 :P = ASC ( MID$
    (ST$(N ) ,M ,1 ) )
750 POKE CURR ,P + 128 * (P < 127 ) :CURR = CURR
    + 1 : NEXT
760 P = ASC ( RIGHT$ (ST$(N ) ,1 ) ) :P = P - 128
    * (P > 127 ) : POKE CURR ,P :CURR = CURR + 1
    : NEXT : GOTO 21Ø
770 REM SAVE MACROS ON DISK
780 HOME : VTAB 6 : HTAB 12 : PRINT "PUT^MACROS^
    ON^DISK" : VTAB 18 : PRINT SPC( 8 ) "RETURN^
    DOES^A^DIRECTORY"
790 VTAB 12 : PRINT "FILENAME=>" ; : FLASH :
    PRINT "^" CHR$ (8 ); : NORMAL : WAIT - 16384
    ,128
800 IF PEEK ( - 16384 ) = 155 THEN GET A$ : GOTO 21Ø
810 INPUT "" ;F$ : IF PEEK (512 + LEN (F$ ) ) THEN
    VTAB PEEK (37 ) : CALL 64578 : FOR A = 512 +
    LEN (F$ ) TO 768 : IF PEEK (A ) THEN F$ = F$
    + CHR$ ( PEEK (A ) ) : NEXT
820 IF F$ = "" OR LEFT$ (F$ ,1 ) = "," THEN HOME
    : PRINT CHR$ (4 ) "CATALOG" F$ : PRINT : GET
    A$ : GOTO 78Ø
830 PRINT CHR$ (4 ) "BSAVE" F$ ",A$92ØØ,L$4ØØ"
    : GOTO 21Ø
```

The Macro Editor Checksums

1Ø	- $BADD	43Ø	- $2C19
2Ø	- $9B13	44Ø	- $23D8
3Ø	- $4D3B	45Ø	- $F215
4Ø	- $AD92	46Ø	- $5136
5Ø	- $C899	47Ø	- $D3D6
6Ø	- $FF65	48Ø	- $CC22
7Ø	- $A3BF	49Ø	- $969A
8Ø	- $A9ØØ	5ØØ	- $E9F9
9Ø	- $924D	51Ø	- $1DE4
1ØØ	- $CADA	52Ø	- $977B
11Ø	- $B6E4	53Ø	- $DED4
12Ø	- $5F48	54Ø	- $DF87
13Ø	- $ACE5	55Ø	- $4C29
14Ø	- $F1Ø4	56Ø	- $9F33
15Ø	- $71DD	57Ø	- $B647
16Ø	- $892B	58Ø	- $C843
17Ø	- $D1A8	59Ø	- $66F1
18Ø	- $D2B6	6ØØ	- $CF25
19Ø	- $DØ28	61Ø	- $3EF1
2ØØ	- $2Ø34	62Ø	- $5A77
21Ø	- $FD8A	63Ø	- $1EØE
22Ø	- $A5E5	64Ø	- $D7E2
23Ø	- $13B7	65Ø	- $Ø5EE
24Ø	- $F995	66Ø	- $E1C6
25Ø	- $1137	67Ø	- $3Ø25
26Ø	- $Ø98C	68Ø	- $2665
27Ø	- $Ø98A	69Ø	- $Ø84A
28Ø	- $23CC	7ØØ	- $E371
29Ø	- $532C	71Ø	- $35EØ
3ØØ	- $8Ø27	72Ø	- $9ØFB
31Ø	- $D361	73Ø	- $A34D
32Ø	- $831F	74Ø	- $3BAØ
33Ø	- $E3D8	75Ø	- $1399
34Ø	- $ØCDØ	76Ø	- $F9F1
35Ø	- $8AØB	77Ø	- $4Ø2Ø
36Ø	- $417Ø	78Ø	- $637C
37Ø	- $BC96	79Ø	- $D7Ø4
38Ø	- $6Ø29	8ØØ	- $481F
39Ø	- $C42E	81Ø	- $A382
4ØØ	- $1CØE	82Ø	- $83Ø6
41Ø	- $65AB	83Ø	- $A9CC
42Ø	- $CØ77		

Isn't It Time You Graduated?

Enhance your Apple //e experience with the **SENIOR PROM**. Instantly double the ROM memory in your Apple //e and add software enhancements without losing ANY compatibility!

Capabilities provided by the **SENIOR PROM** include:

- A Complete Menu of Functions Available from the RESET key:
  - -Ability to enter the monitor at any time.
  - -Copy the entire contents of Main memory to Auxiliary memory, and vice versa.
  - -Move Main or Auxiliary upper 16k RAM memory to easily accessible lower memory.
  - -Copy volitile memory areas (normally destroyed upon Reset) to easily accessible memory.
  - -Write a test pattern into Main or Auxiliary memory to study the disk booting process.
- ROM Resident DOS with a Complete set of DOS 3.3 Utilities that Include:
  - -Initialize disks; copy all or selected portions of a disk.
  - -A sophisticated ROM resident track, sector and memory detective and editor.
  - -Alter DOS recognition (header) bytes to examine copy protected software.
- Non-Maskable Interrupt (NMI) Support which allows:
  - -Capture all 64k of Main RAM memory to Auxiliay memory or a normal DOS 3.3 disk.
  - -Re-load and restart any program captured on disk up to 64k.

Although Senior is completely menu driven, it comes with an extensive user's manual which gives operating principles, examples of use, and a complete guide to understanding copy protection. Unlock your Apple //e mysteries with the **senior prom**! Specify "Standard" or "Upgraded" //e ROMs.

Cutting Edge Enterprises. Box 43234 Ren Cen Station, Detroit, MI 48243. 313-349-2954 Modem.

$85.95 ($79.95 money order, check discount). 317-743-4041 visa, mc orders 10-5.

Attention Franklin Ace 100, 1000 owners: Add Composite Color Video only $25.95 ($19.95 if Money Order, Check).

By Hackers
For Hackers

- **ELITE BOARD DOWNLOADS**
- **CRACKING TIPS**
- **PHREAKING SECTION**
- **GAME CHEATS**
- **PARMS**
- **PROGRAMS**
- **INTERVIEWS**
- **ADVENTURE TRIPS**
- **HACKING TIPS**
- **MYSTERY SECTIONS**

Published on both sides of an Apple diskette - 4 times a year.

The BOOT-LEGGER
MAGAZINE

Subscribe Now!

Send 25 Bucks for a 1-Year Subscription THE BOOT LEGGER, 3310 Holland Loop Road, Cave Junction, Oregon 97523. Overseas Subscriptions $50. Canadian $30 U.S. Currency.

FOR AD INFO. & QUESTIONS
CALL BOOTLEG AT (503) 592-4461

NEW SUBSCRIBER?

of Hardcore COMPUTIST

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

BACK ISSUES

Issue	Mag	Disk
	$4.75	$9.95
HC 21	☐	☐
HC 20	☐	☐
HC 19	☐	☐
HC 18	☐	☐
HC 17	☐	☐
HC 16	☐	☐
HC 14	☐	☐
HC 13	☐	☐
HC 11	☐	☐
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:

Name ____________________

ID# ____________________

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.

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

Hardcore COMPUTIST 13: Softkeys for Laf Pak, Beyond Castle Wolfenstein, Transylvania, 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 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.

This is your last

WARNING

You have entered
DANGEROUS
territory

Use this special order form for

The
Book Of Softkeys
Vol. 1

Send me copies of the Book Of Softkeys. I have enclosed $20 PLUS shipping, handling, and taxes for each copy.

Name ______________________

ID# ______________________

Address ______________________

City ______________________

State __________ Zip ______________

Phone ______________________

VISA/MC _____ - _____ - _____ - _____

Exp ______________________

Signature ______________________

Send check or money order to:
Book Of Softkeys
Softkey Publishing
PO Box 110816-T
Tacoma, WA 98411
Washington residents add 7.8% sales tax.
Foreign orders add 20% shipping and handling.
US funds drawn on US bank.

REMEMBER!
Always be prepared whenever you enter the adventure-filled world of...

EAMON

Six different text-adventures for the Apple.
All for only $20.00!

☐ Please send me the EAMON Introductory Offer. I have enclosed $20.00 plus applicable sales tax and shipping & handling. I understand that Public Domain Software is not commercial quality and is supplied as-is and that orders are filled on double-sided disks.

Name ______________________

Address ______________________

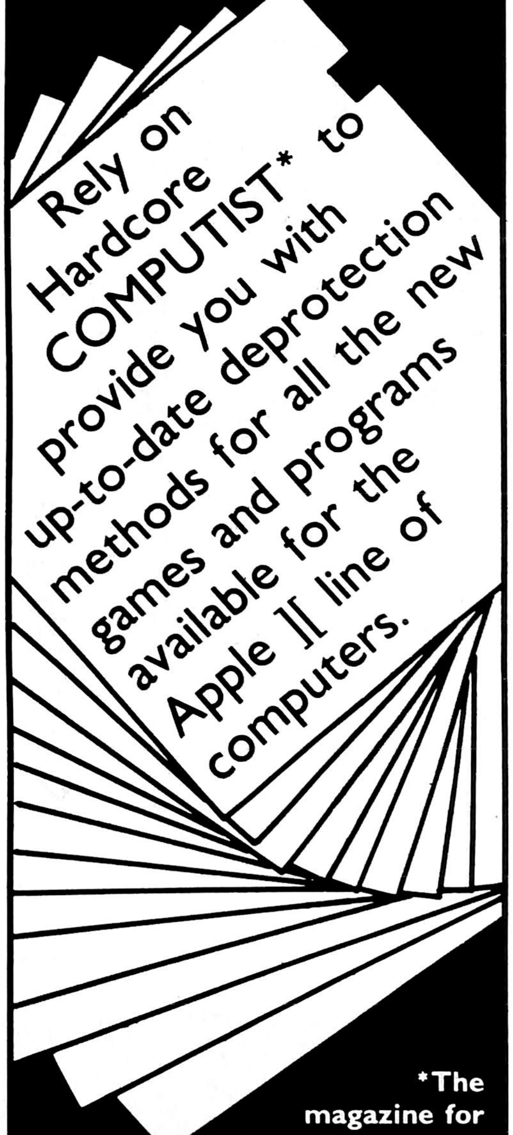
City __________ State __________ Zip __________

Country __________ Phone __________

VISA/MC _____ - _____ - _____ Exp __________

Signature ______________________

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



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

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

Subscription rates (for six issues):
Please check one

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

( ) Yes! Start my subscription now.

Name ______________________

Address ______________________

City __________ St __________ Zip __________

Country ______________________

VISA/MC _____ - _____ - _____ Exp __________

Signature ______________________

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

NEW DON LANCASTER RELEASES

Available ONLY from Synergetics. All software open and unlocked.

ABSOLUTE RESET (IIc/old IIe/new IIe) $19.50
Get back in total control. No more hole blasting! Access any part of any program at any time for any reason. Passes all diagnostics. Invisible till activated. Includes EPROM burner adapter details, service listings, and a free bonus book.

APPLEWRITER™ IIe TOOLKIT $39.50
EIGHT diskette sides include a complete disassembly script, self-prompting glossaries, Diablo microjustify and proportional space, patches for NULL, shortline, IIc detrashing, answers to lots of help line questions, source code capturing, two columns, WPL secrets, space on disk, keyword indexing, more.

ProDOS APPLEWRITER™ 2.0 TOOLKIT $39.50
Another EIGHT diskette sides crammed full. Includes a neat automatic source code capturer, a complete disassembly script, dozens of patches, prefix hassle fixes, Grappler and shortline repairs, NULL diversion, two columns, auto indexing, plus Don's unique self-prompting glossaries, Diablo microjustify, etc.

Both TOOLKITS $59.50

APPLEWORKS™ DISASSEMBLY SCRIPT $49.50
TEN diskette sides chock full of Applework's innermost secrets, using Don's unique and powerful "tearing method". Primarily for gonzo hackers and definitely NOT for the faint of heart.

COMPANION DISKETTES:

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

ASSORTED GOODIES:

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

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

FREE
VOICE HELPLINE
(602) 428-4073

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

Apple][,][+, //e, Franklin users:

Do you have problems backing-up your copy-protected programs?

Do you lack parameters for your copy programs?

Are you looking for programs that you can AFFORD?

Are you hesitating to upgrade your equipment due to expensive prices quoted in other ads?

It's simple now. Just drop us a line.

Send $1.00 U.S. funds to:

Reliant
P.O. Box 33610
Sheungwan, Hong Kong

IMPORTANT: We have over 1000 PC name-brand programs and various hardware offers. Programs @ U.S. $8.00/PC includes the disk and registered air-mail handling.

Now-the ultimate back-up system!

EDD III® and TRAK STAR™

COMPLETE PACKAGE: 159.95*

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

PRICED SEPARATELY:

TRAK STAR 99.95

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

Documentation: $3 Refundable with the purchase of TRAK STAR

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

Save copying time with nibble programs

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

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

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

Midwest Microsystems

To order, phone: 913 676-7242

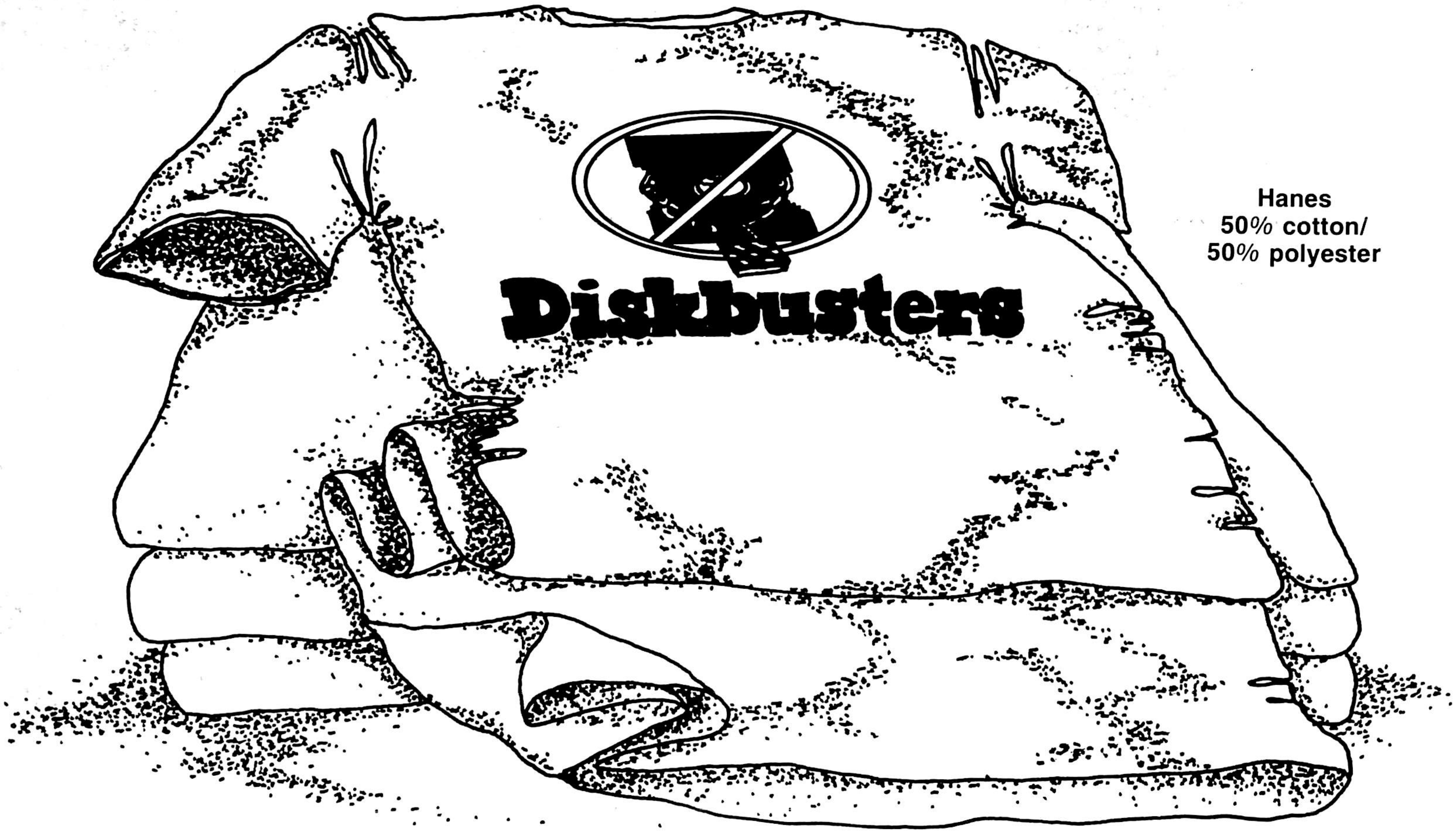
9071 Metcalf / Suite 124
Overland Park, KS 66212

it's summer.

Only $9.95

That means T-SHIRT weather!

What better way to show your friends *just* what you think of software copy-protection than to wear the ORIGINAL Hardcore COMPUTIST **DISKBUSTERS t-shirt?!!!**



Hanes 50% cotton/ 50% polyester

Hardcore COMPUTIST says:

"We ain't afraid of no disk!"

YOU CAN, TOO!

Rush me ________ (total) DISKBUSTERS T-shirts in the sizes indicated below. I have enclosed $9.95 (plus applicable tax & shipping) for each shirt.					
ADULT MENS:	____Small	____Med.	____Large	____X-Large	
Name					
Address					
City		St		Zip	
Phone					
VISA/MC	____-____-____-____			Exp	
Signature					

Send check or money order to: Hardcore T-shirts, PO Box 110816, Tacoma, WA 98411. Washington residents add 7.8% sales tax. Foreign orders add 20% shipping and handling. US funds drawn on US bank.

Use your VISA/MC, personal check or money order.

Shirts available in adult sizes ONLY! (black & red on gray).

Hardcore COMPUTIST

PO Box 110816 Tacoma, WA 98411