For The Serious User Of Apple][Computers

COMPUTIST

May 1986

Issue No. 31 $3.75

Softkeys For:

Trivia Fever

Lifesaver

Synergistic Software

Blazing Paddles

Zardax

Time Zone

Tycoon

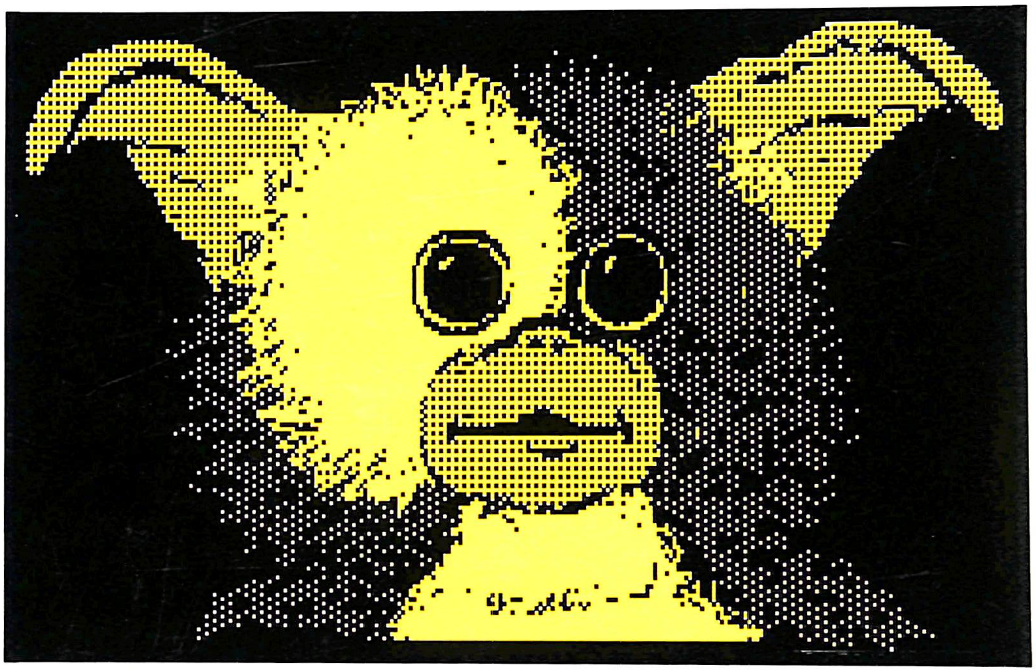
Earthly Delights

Core:

Black Box

Feature:

A little help with The Bard's Tale



(Page 23)

COMPUTIST
PO Box 110846-T
Tacoma, WA 98411

Many of the articles published in 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.

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

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

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

```
PR#6
```

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

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

```
6⌃P
```

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

Requirements: Most of the programs and softkeys which appear in COMPUTIST require one of the Apple][series of computers and at least one 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 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 COMPUTIST No. 22 as a back issue or order Program Library Disk No. 22.

RESET Into The Monitor: Some 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 COMPUTIST No. 6 or the "Dual ROM's" article in COMPUTIST No. 19.

Apple //e - Apple //c: Install a modified CD ROM on the computer's motherboard. Clay Harrell's company (Cutting Edge Ent.; Box 43234 Ren Cen Station-HC; Detroit, MI 48243) sells a hardware device that will give you this ability. 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 Pieter Lechner, 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 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 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 (ᐞ). All other spaces in a 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 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 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 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 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 COMPUTIST No. 1 and The Best of Hardcore Computing. An update to CHECKSOFT appeared in COMPUTIST No. 18. If the checksums these programs create on your computer match the checksums accompanying the program in the magazine, then you keyed in the program correctly. If not, the program is incorrect at the line where the first checksum differs.

1) To compute CHECKSOFT checksums:

```
LOAD filename
BRUNCHECKSOFT
```

Get the checksums with

```
&
```

And correct the program where the checksums differ.

2) To compute CHECKBIN checksums:

```
CALL -151
BLOAD filename
```

Install CHECKBIN at an out of the way place

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

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

```
xxx.xxx⌃Y
```

And correct the lines at which the checksums differ.

Coping with COMPUTIST

Welcome to 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.

renew your freedom

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

if you're moving...

Let us know right away or at least 30 days in advance so that you won't miss a single issue. Just write your new address below, and include your present address label. Issues missed due to non-receipt of this Change-of-Address may be aquired at the regular back-issue rates.
Please remember, the Post Office does not forward third class mail unless requested.

We are NOT PIRATES

but we're not fools, either.

We're serious programmers and software users who just want to have backup copies of any software we own. COMPUTIST magazine shows us HOW TO MAKE BACKUPS OF COMMERCIAL SOFTWARE regardless of the maker's attempt to stop us from having legal copies. Don't let them stop you from protecting your own rights.
Remove copy-protection from your valuable library of expensive software. The publisher of COMPUTIST has been showing subscribers how to unlock and modify commercial software for the past 4 years. Don't be one of the users abused by user-FIENDLY locked-up software. Subscribe.

6 issue SUBSCRIPTION RATES:
U.S.: $20 first class: $24
Canada, Mexico: $34
Other Foreign: $60
SAMPLE COPY:
U.S.: $4.75 Foreign: $8.00
US funds drawn on U.S. bank.
In Washington add 7.8% tax.
Send check or money order to:

COMPUTIST
PO Box 110846-T
Tacoma, WA 98411

□ NEW subscriber □ Renew my subscription □ New address

Name ______________________________ ID# ________

Address ______________________________

City ______________ State ________ Zip ____________

Country ______________ Phone ______________

VISA ________-________-________-________ Exp. ________

Signature ______________________________ CP31

ATTENTION ADVENTURERS

Adventure Tips ....and Clues

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.

We prefer that these hints not be a dead giveaway to solutions of dilemmas presented by the particular game. Adventure tips should contain just enough information to nudge the stumped adventurer towards the answer.

How & Where

So, if you know how to open the jewel-encrusted egg, how to plug the hole on 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:

COMPUTIST TIPS
PO Box 110846
Tacoma, WA 98411

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. The grateful sighs of all the readers you have helped shall be your just payment. Maybe your Charisma will go up, too!

BULLSHIRT!

by bobby
PHD, phD, Phd, PhD,
BHT, BLT, MSG, McDLT

Requirements
some $$$
postage stamp
Envelope
Pen or compatible word processor
scissors (optional)

Now that HARDCORE has changed its name to plain old 'computist', I've been assigned the regrettably redundant responsibility of changing all the Diskbusters T-shirts. Instead, using my own imaginary initiative, I've come up with a reasonable facsimile of an original idea...

"Why not," I asked myself, "call all the old Diskbusters T-shirt something catchy like: Hardcore Classic?"

"Wow!" I answered myself, "Then we could sell people the NEW computist T-shirts this summer!"

(I've heard that some people suspect that there's a rumor that such an artifact does not yet exist...VAPORWEAR...)"

If you like my idea, then PLEASE order the CLASSIC Hardcore Computist DiskBusters T-shirt (simply known as CLASSIC T-shirt)

1) Fill out the order form below using the Pen and then insert the order form with appropriate $$$ or acceptable substitutes into the envelope, and then pen-in the address onto the envelope, and then attach the stamp and that's it.

Enjoy!

I want the CLASSIC! so rush me _______ (total) Classic DISKBUSTERS T-SHIRTS in the sizes indicated below. I have enclosed $9.95 (plus tax and shipping) for each shirt.

ADULT MENS: _______Small _______Medium _______Large _______Xtra large

Name____ ID#________

Address____

City_________________________ State__________ Zip_____________

Country _________________ Phone _________________

VISA ☐ ☐ _______-_______-_______-_______ Exp. ________

Signature____ CP31

Send check or money order to: Bullshirt; PO Box 110816-T; Tacoma, WA 98411. Washington orders add 7.8% sales tax. Foreign orders add 20% shipping and handling. US funds drawn on US bank. OFFER GOOD WHILE SUPPLIES LAST.

$5 SAVE $5

The Book of Softkeys

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

Coupon Expires June 15, 1986

$5 SAVE $5

Isn't it Time You Graduated..?

...to the Senior PROM Version 2.0

Allows your Apple //c or //e to **halt** a program and:

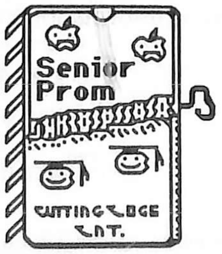
- Instantly swap between two different 64k programs.
- Enter the Monitor at any time, from any program.
- Examine where in memory the program was running.
- Disassemble, view or save memory, even $00-7FF.
- Examine the Stack for return subroutine addresses.
- Alter any memory to examine running effects.

The program may then be **restarted** exactly where it was interrupted or at any other location, or be **saved** to disk in normal B-files and later restarted.

Also includes a sophisticated Sector Editor and Memory/Disk Detective, and operating system with disk copy, format, edit, and protected disk utilities, all **without** booting DOS first! Assembly Language utilities include Step & Trace, an Assembler, & more.

All the **Senior PROM's** utilities are in ROM, instantly available when needed. Undetectable by any software or hardware, does not use a peripheral slot! Extensive documentation & guide to copy protection.

Economically priced at **$79.95** for prepaid orders with check or money order. Credit card orders available for **$88.95**. Please specify //c, or //e Standard or Enhanced ROMs.



Cutting Edge Enterprises
Box 43234 Ren Cen Station
Detroit, MI 48243

For ☐ VISA orders call
317-743-4041, 10-5 E.S.T.
or 313-349-2954 Modem 24 hrs.

not intended for illegal use.

COMPUTIST

May 1986 Issue 31

Publisher/Editor: Charles R. Haight **Managing Editor:** Ray Darrah
Technical Editor: Robert Knowles **Circulation:** Michelle Frank, Debbie Holloway
Advertising: (206) 474-5750 **Printing:** Valco Graphics Inc., Seattle, WA
COMPUTIST is published monthly by SoftKey Publishing, 5233 S. Washington, Tacoma, WA 98409
Phone: (206) 474-5750

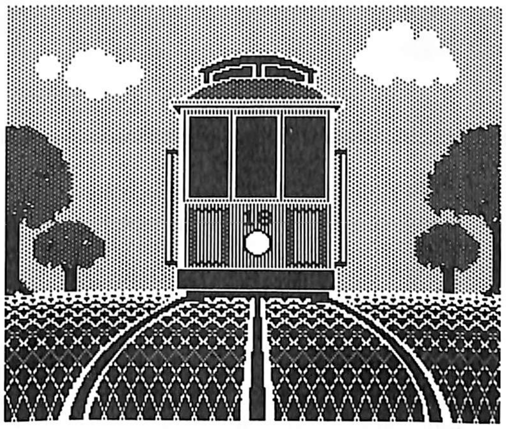
softkeys:

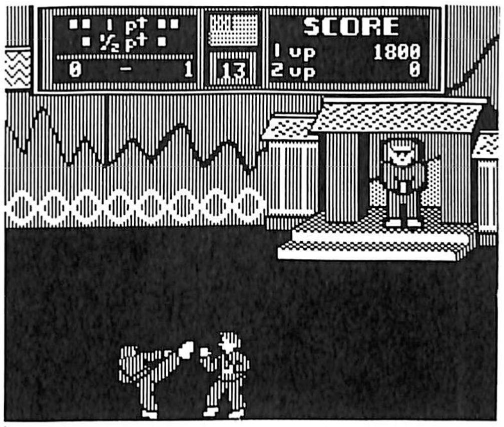
feature:

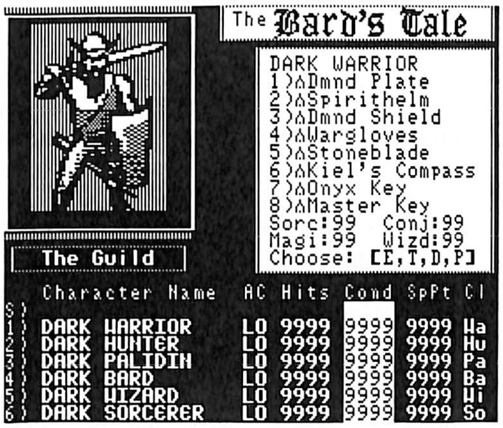
core:

quickie:

departments:







```
The Bard's Tale

DARK WARRIOR
1)^Dmnd Plate
2)^Spirithelm
3)^Dmnd Shield
4)^Wargloves
5)^Stoneblade
6)^Kiel's Compass
7)^Onyx Key
8)^Master Key
Sorc:99  Conj:99
Magi:99  Wizd:99
Choose: [E,T,D,P]

The Guild

   Character Name    AC Hits Cond SpPt Cl
S)
1) DARK WARRIOR      LO 9999 9999 9999 Wa
2) DARK HUNTER       LO 9999 9999 9999 Hu
3) DARK PALIDIN      LO 9999 9999 9999 Pa
4) DARK BARD         LO 9999 9999 9999 Ba
5) DARK WIZARD       LO 9999 9999 9999 Wi
6) DARK SORCERER     LO 9999 9999 9999 So
```

This month's cover:
Graphics created with Baudville's "Blazing Paddles."

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

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

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

Apple usually refers to an Apple][computer and is a trademark of Apple Computers, Inc.

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

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

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

input

Please address letters to:

COMPUTIST
Editorial Department
PO Box 110846-K
Tacoma, WA 98411

Include your name, address and phone number.

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.

Our technical staff is available for phone calls between 1:30 pm and 4:30 pm (PST) on Tuesdays and Thursdays only.

Bored With Milton Bradley

As a professional educator, I'm always irritated with software that I have to let students use and is either heavily disk interactive, has *students* store information on disk or has information that I would like to modify from year to year (for various reasons such as changes in curriculum, or test sections on which students occasionally pass information from year to year.) For this reason, when our special education people had trouble with a Milton-Bradley disk and asked me for help, I agreed to try to at least salvage their student record file, even though I knew precious little about non-standard format disks. With a bit of luck, I was able to open up the whole disk, as well as several other Milton-Bradley programs and they are now able to both easily backup the disks and recover trashed files. I'd like to share the process with your readers. (I've only recently become one, myself.) The process is a little cumbersome, probably because this is the first time I've opened up a disk, but it both works and helps the graphics portions load faster.

The Softkey

This softkey works on Milton-Bradley's "Vocabulary Skills," "Punctuation Skills," "Ratios & Proportions," "Mixed Numbers" and "Building Better Sentences."

1) Boot the system master disk and execute COPYA.

RUN COPYA

2) When the program comes up, break into Applesoft.

CTRL C

3) Enter the monitor and disable DOS errors.

CALL -151
B942:18

4) Get Back into BASIC and delete line 7Ø. (This keeps COPYA from re-loading the machine language part of the program.) The restart COPYA.

3DØG
7Ø
RUN

5) Follow the COPYA menu and copy the disk.

That's it! You may wish to put a fast DOS on the disk such as Hyper DOS or Pronto DOS. In my experience, the boot program is called BOOT. (That was a really tough one to figure out.) If it's different, you may have to try booting different programs until you get the right one, but at least now you can find the directory.

On a similar subject, I'm looking for a method to modify the files on an old but excellent oil well search simulation program called Geology Search, by McGraw-Hill. Unfortunately, the location of the wells is set and lists are now being passed from year to year. If anyone could help, I appreciate it.

John E. Wanner
Sullivan, WI

Copy Card vs. Integer Card

For starters, I would like to let readers know that they can easily backup "The Print Shop Companion" by Broderbund and "The Graphics Expander Vol. 1" by Springboard by using current parameters (dated 11/15/85) from Copy][Plus's Bit Copy program.

To copy Print Shop companion, choose Print Shop from Central Point's list of parameters. To copy the Graphics Expander Vol. 1 just choose Newsroom (alternate #1). Both parameter files easily copy these programs.

I also have a question concerning copy cards. What's the difference between an Integer Card and a Copy Card such as Wildcard and Replay? If you had a choice what type would you obtain? Finally, if the Integer Card is better, where do you obtain one?

I really enjoy your magazine and look forward to receiving each issue.

Frank Hahn
Muscatine, IA

***Mr. Hahn:** While an Integer card is a useful tool, there is no comparison between one and a copy card. The copy card is unquestionably more useful. The only added ability an Integer card gives is you is the ability to Reset into the monitor. A copy card on the other hand does this plus a whole lot more. For example, with a copy card you can find out where the program in memory was executing at the time you pressed the button. This is impossible with an Integer card.*

More F-15 Strike Eagle

The author of the *F-15 Strike Eagle* softkey is obviously a novice. The disk has a catalog on track 15. The DOS is a standard, virtually unmodified 64K Diversi-Dos. The protection routines are kept in the files "AS", and "SIDTRANC.OBJ3." Because there are at least three versions that I have personally seen, I can tell you with complete authority that no two versions have the exact same code, and that each uses different sectors on the disk. This virtually invalidates your printed softkey. I will not describe the versions I have seen, because there are doubtless others; anyone who owns a copy should:

1) Boot normal DOS.

2) Get into the monitor, and change ACØ1:15.

3) Put F-15 Strike Eagle in and catalog it.

4) BLOAD AS,A$2ØØØ. Examine it for the references to $2Ø1 as mentioned in the softkey. This file cannot be BSAVED, because it normally loads at $2Ø1. Find out what must be changed, get a sector editor and change it.

5) BLOAD SIDTRANC.OBJ3,A$2ØØØ. Examine this file too - it gets interesting after about $6Ø bytes. I can't give much help here; but watch for calls to $F944 or other RWTS locations followed by a "BCS". Replace the BCS with EA's. Also, don't let it store anything at $6AØØ. Because the normal load location for this is $ØØØØ, it would be best to edit it on disk.

input

6) Get a sector editor and make the changes. If it won't boot, AS is in trouble; if you still get "mechanical failures," check on SIDTRANC.OBJ3 again.

I hope this helps.

M. M. McFadden
Sunnyvale, CA

Print Shop White Line Fix

Here is how I print cards, etc. from PRINT SHOP without those dreaded white lines they say you must live with. I have an AMUST80 printer (a so-called EPSON compatible) drived by an EPSON APL card in slot 1.

1) Make a copy of PRINT SHOP.

2) Use a track-sector editor to change the following on the copy:

TRACK	SECTOR	BYTE	FROM	TO
07	0E	0B	97	9E

```
Change bytes A2-B1 to

     20 BB 1A
     A9 41
     20 BD 1A
     A9 06
     20 BD 1A
     4C 97 18
```

That's it!

Explanation

Track 07, Sector 0E is the beginning of a binary file called PRCOMS (printer commands). [**ed:** The start of this file will probably be on a different sector of *your* disk.] If you have an EPSON MX-FX-RX or clone printer the area used by this piece of code is skipped over. So I was able to insert extra commands to send <ESC> (20 BB 1A), which sets the line feed to 6/72 inch before returning to PRINT SHOP's bit-mapped graphics routine (4C 97 18).

Now if you have chosen some option other than the first on the printer setup list, you will have to return to Track 07, Sector 0E and make another change: bytes 9E-A1 become:

```
for option 2: C9 02 F0 10
           3: C9 03 F0 4A
           4: C9 04 F0 46
           5: C9 05 F0 36
           6: C9 06
```

If you don't do this, the code inserted will cause a crash.

Ferg Brand
Lane Cove, Australia

Death in the Caribbean Fix

I agree with Nick Galbreath that *Death in the Caribbean* is a great adventure and having never successfully backed-up my copy, I followed his procedure in COMPUTIST No. 23. Well, apparently Nick had never gotten as far as the back side of the disk because my backup thought that side 2 was just another copy of side 1 and refused to let me play further, although side 2 did contain the correct picture information. I had to track down the problem.

The "Hello" program loaded and ran the binary file "INPUT>OBJ1" at $4000. In the vicinity of $40D0, I learned that the program checked to see if the volume found was 254 and if not printed "TURN DISK OVER AND PRESS <SPC>". I CATALOGed side 2 and found that it was a volume 222.

The solution to the problem is to use Nick's softkey but when unlocking the backside (side 2) the disk must be INITialized with volume "222" when asked for a volume number. Now, "Death in the Caribbean" is finally totally cracked.

Gary Kowalski
Anaheim, CA

Zapping the New PFS

The new ProDOS versions of the PFS series are a big improvement over the old in that they can be installed on a hard disk or the new Uni-Disk 3.5 and the file size is limited only by the storage device. In addition, there have been a number of improvements in the use of control keys; the ability to print to disk, etc. However, Software Publishing has bucked the no protection trend in ProDOS software by forcing the user to have the original disk in the slot 6 drive as a key disk in the boot up process.

Luckily, the code that checks for this is easily circumvented by the following sector edits. Just use any copy program such as COPYA, FILER, etc. and then make these edits on the copy.

PFS FILE & PLAN

TRACK	SECTOR	BYTE	FROM	TO
1	D	CA	20	4C
1	D	CB	15	7D
1	D	CC	0F	10
1	D	EF	20	4C
1	D	F0	15	7D
1	D	F1	0F	10

PFS WRITE & REPORT

TRACK	SECTOR	BYTE	FROM	TO
4	1	CA	20	4C
4	1	CB	15	7D
4	1	CC	0F	10
4	1	EF	20	4C
4	1	F0	15	7D
1	1	F1	0F	10

These edits completely deprotect the disk (specifically, the xxxx.SYSTEM file on each). The files may now be moved with any ProDOS file utility. Hard disk and Uni-Disk 3.5 users will not need the key disk and floppy users will be able to back up without problems.

Robert James
Mt. Sinai, NY

EDD III Re-revisited (again)

Although COMPUTIST has dedicated quite a number of articles on the cracking of EDD III, I feel you should know about the simplified technique that finally worked for my version 3 dated May 25, 1984. This had been the only program that I had never been able to copy by any means and even though I am a registered owner, I was reluctant to use this powerful utility very much for fear of zapping the precious disk. Thanks to the fine publication known as "COMPUTIST" and Beagle Brothers' memory maps, this fear was finally put to rest. Below, I will outline the procedure.

This procedure uses Steve Dietz's letter "The Quicker Aux" on page 6 of COMPUTIST No. 25 and M. A. Todd's letter "The EDD Syndrome" on page 4 of COMPUTIST No. 28. This was accomplished on an Apple //e with an Apricorn 80 column video display with 64K on board. (This is an extended 80 column card.)

I followed Mr. Dietz's procedure until the last step where he tells you to move DOS and RWTS into main memory starting at $800.

input

Since DOS is already in main memory, there is no point in putting it somewhere else. What you want is the EDD moved from auxiliary memory to main memory. EDD will not execute when you do this so to move EDD from auxillary memory to main memory, type:

```
800<800.5B06CTRL Y
```

This adds a little garbage to the end of the program, but it is purged later.

Next I used a shortened Todd technique by combining steps 3 & 4 to:

```
3EE8: 60 60
```

Since they are redundant and unnecessary, I omitted steps 5, 6, and 7.

Finally, steps 8 & 9 can be combined so line 4020 reads:

```
4020: 0A A9 60 85 0B 4C 53 09
```

When this EDD III is BSAVEd to a Pronto DOS initialized disk with a turnkey HELLO file, the program loads in 8 or 9 seconds. As far as I can tell, the program works as well as the original except for the command #8, the boot disk feature; but that is not useful anyway.

Many thanks to COMPUTIST, its staff and contributors for making the Apple][series some of the most useful computers anywhere.

John Gubbins
Aurora, CO

A Difficult TASC

The BASIC compiler "TASC" by Microsoft is a useful utility that makes Applesoft programs run much faster and more efficiently.

Having used it for a (to me) very important program, I have lost the original and need to make amendments.

I phoned Microsoft to obtain an UNTASC (a program which would convert code created by TASC back to Applesoft) and they flatly refused. I now call on some enthusiastic user to do just this: Create UNTASC.

Thru your goodselves or other readers, how about cracking this code. I have started but find that time is my enemy.

It is a challenging "TASK."

Basil Brand
South Africa

Mr. Brand: TASC is supplied in a COPYAable format.

We at COMPUTIST believe that all software should have its source code included with it when you purchase the program. Perhaps someday this dream will come true but for now you just have to wait for someone else to create such code.

Microzine & Wizard of Words

I received COMPUTIST No. 27 and was pleased to see the softkey for the Microzine series. All attempts with my bit copiers failed on the microzine series. The bit copy would try to boot only to hang on a full page of inverse letters.

I remembered the same result was true for another educational piece that I had. It is the "Wizard of Words" by Advanced Ideas. So I thought I would try the Microzine 7-9 softkey and it worked on the Wizard of Words.

To deprotect Wizard of Words:

1) Boot DOS 3.3

2) Execute COPYA from the system master.

```
RUN COPYA
```

3) Break out of the program.

```
CTRL C
```

4) Enter the monitor and tell the object code to ignore errors.

```
CALL-151
5942:18
```

5) Go back to Applesoft and startup COPYA.

```
3D0G
RUN
```

6) Copy both sides of Wizard of Words.

7) Search the copied disk for the sequence C6 2A D0. You should find it on track 1, sector 5 just as in the Microzine series. Change these bytes to 4C 86 02.

Thanks to Phil Pattengale for the Microzine softkey that enabled me to back up Wizard of Words.

The Pretzel
Bloomington, IL

bugs

COMPUTIST No. 27:

Apple Logo II Softkey:

The last sector edit should read:

Track	Sector	Byte	From	To
$02	$09	$8B	$84	$21

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 COMPUTIST readers informed of the programs for which softkeys are MOST needed. Send your requests to:

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 COMPUTIST readers and earning MONEY at the same time. Send the information to us in article form on a DOS 3.3 diskette.

Flight Simulator][	Sublogic
Mouse Calc	Apple Computer
Apple Business Graphics	Apple Computer
Jane	Arktronics
Visiblend	Microlab
Catalyst	Quark, Inc.
Gutenburg Jr. & Sr.	Micromation LTD
Prime Plotter	Primesoft Corp.
The Handlers	Silicon Valley Systems
The Apple's Core: Parts 1-3	The Professor
Fun Bunch	Unicorn
Willy Byte ...	Data Trek
Terrapin Logo V2.00	Terrapin Software
Conan	Datasoft
Cycloid	Sirius Software
Adventure	Microsoft
Cranston Manor	Sierra On-Line
Snoggle	Broderbund
Robot War	Muse
ABM	Muse
Mychess II	Datamost
E-Z Learner	Silicon Valley Systems
Story Tree	Scholastic
Agent U.S.A.	Scholastic
Handicapping System	Sports Judge
Dollars & Sense	Monogram
Echo Plus	Agranat Systmes
Great Cross Country Road Race	Activision
Raster Blaster	Budge Inc.
GATO v1.3	Spectrum Holobyte
Odin	Odesta

readers' softkey & copy exchange

Danny Pollak's softkey for...

Time Zone

Sierra On-Line, Inc.
10398 Rockingham Dr., Ste. 12
Sacramento, CA 95827
$99.95

Requirements:
Apple][Plus or equivalent
Time Zone V1.1
Super IOB 1.5
Six double-sided (notched) blank disks

Time Zone is one of the (if not THE) largest adventure games ever created. There are nine different time zones to visit, with up to seven different places to travel within each zone. The main objective of the game is to stop the evil Neburites from demolishing the Earth by destroying their ray gun. To do so, you must first solve all the other puzzles and then travel to Neburon on the last leg of your quest.

The Protection

The game is supplied on six double-sided disks. All of the disks are written in standard format. The only protection that can be found is a nibble count on track $00 of side A (they use self-modifying code to try to hide it).

Step by Step

To copy side A:

1) Install the controller below into Super IOB 1.5.

2) Run Super IOB.

3) Follow the prompts and initialize the blank disk with a volume number of 2.

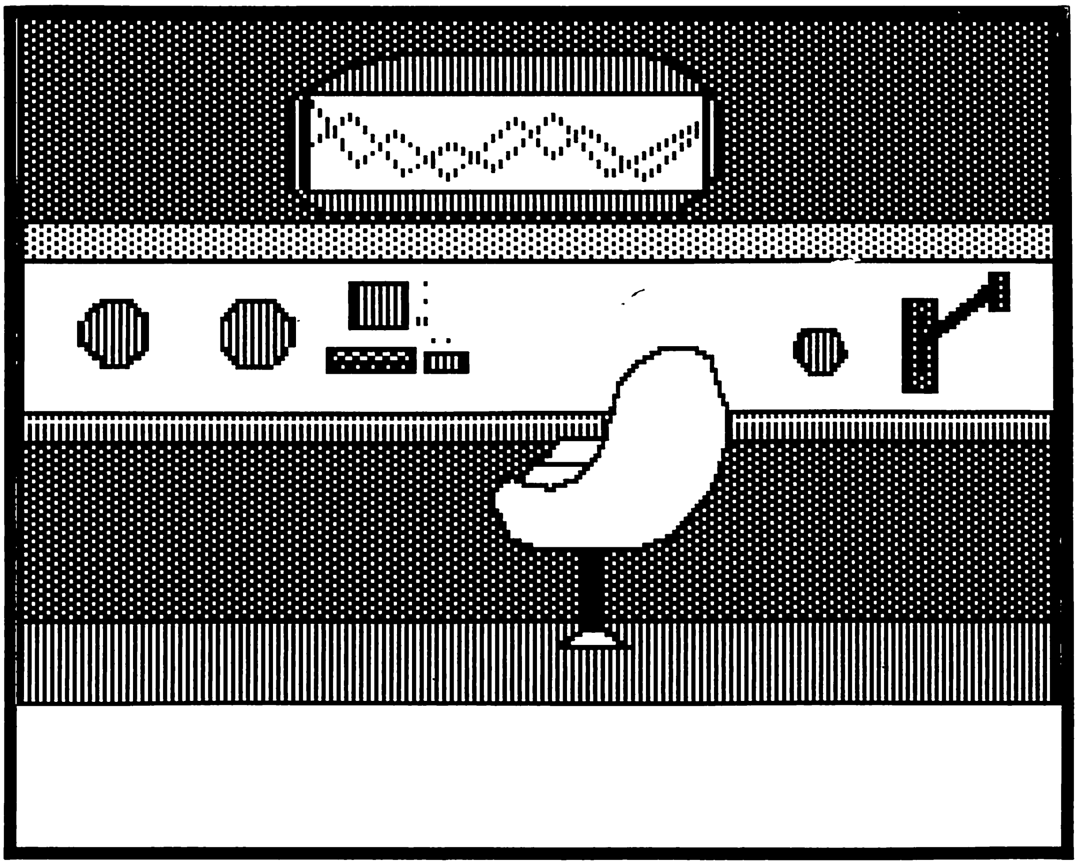
When Super IOB is finished, the following sector edits will have been already performed.

Track	Sector	Byte:	from:	To:
3	Ø	$D9	$FC	$ØØ
		$DA	$Ø8	$13

5) To copy sides B-L, use the built-in copier that is supplied with the program. You can also use most other whole disk copiers like COPYA.

controller

```
1ØØØ REM TIME ZONE V1.1
1Ø1Ø TK = Ø :LT = 35 :ST = 15 :LS = 15 :CD = WR :FAST
     = 1
1Ø2Ø GOSUB 49Ø : GOSUB 61Ø :T1 = TK :TK = PEEK (TRK
     ) - 1 : GOSUB 31Ø :TK = T1
1Ø3Ø GOSUB 49Ø : GOSUB 61Ø : IF PEEK (TRK ) = LT
     THEN 1Ø5Ø
1Ø4Ø TK = PEEK (TRK ) :ST = PEEK (SCT ) : GOTO 1Ø2Ø
1Ø5Ø HOME : PRINT "DONE" : END
5ØØØ DATA 2^ CHANGES
5Ø1Ø DATA 3 ,Ø ,217 ,Ø
5Ø2Ø DATA 3 ,Ø ,218 ,19
```



controller checksums

```
1ØØØ  - $356B     1Ø5Ø  - $3BØB
1Ø1Ø  - $2544     5ØØØ  - $3EØØ
1Ø2Ø  - $C314     5Ø1Ø  - $838Ø
1Ø3Ø  - $7EC5     5Ø2Ø  - $2AF1
1Ø4Ø  - $881C
```

Charles Taylor's softkey for...

Tycoon

Blue Chip Software
6744 Eton Avenue
Canoga Park, CA 913Ø3

Requirements:
Tycoon program disk
Two blank disk sides
FID
A sector editor
Copy][Plus (optional)

Tycoon is a simulation of the commodity market from Blue Chip Software.

1) INIT two sides of a disk with the name HELLO. Delete HELLO from both sides.

2) Copy all the files from side one of Tycoon to side one of the copy, *except* the files ENTRA and CON.

3) Get out your sector editor and look in the catalog of the copy for something like the filename CCDDIIIINNCC (it's hard to represent in print) and change the name to HELLO.

You can do this by looking for the sequence C3 83 C4 84 C9 89 C9 89 CE 8E C3 83 in track $11, probably sector $E, and replacing it with C8 C5 CC CC CF AØ AØ AØ AØ AØ AØ AØ. Write the sector back out.

An alternate method would be to use the Copy][Plus utility to change the name to HELLO.

4) Copy all the files from side two of the original to side two of the copy.

Done! This procedure can also be used to deprotect Squire and Baron also.

readers' softkey & copy exchange

Patrick Ford's softkey for...

Earthly Delights

Requirements:
At least 64K
COPYA
Sector editor
Blank disk

Earthly Delights is a detective adventure based on the search of a stolen painting, after which the game is named. It was originally written for the IBM PC in PASCAL around 1981 by Roger Webster and Dan'l Leviton. The rights were sold to Datamost who promptly transferred it to Apple's USCD PASCAL then never did much afterward. After years of neglecting the game, Datamost filed for bankruptcy and Dan'l Leviton has acquired the distribution rights.

When booting the game, it is obvious that it is copy protected because it spends several seconds jumping the drive head from track to track. Initially, I thought it was synchronized, because I could find no obvious changes to the track organization. After failing to make a working backup with Locksmith 5.Ø, EDD, or Copy][Plus 5.Ø, I decided to try and locate the offending code.

Fortunately, unlike Sundog, the directory is intact. I examined the code files for assembly subroutines and functions that accessed the disk. I quickly found two identical pieces with the same location, except on the opposite sides of the disk. I have yet to figure out what it is looking for.

Fortune was with me, because instead of returning a parameter to the P-system, (who has a P-code dissassembler?) the routine appears to just return if successful. So all we need to do is insert a RTS as the first statement.

1) Copy both sides of the disk with COPYA.

2) Make the following sector edit to both sides:

track	sector(3.3/PASCAL)	byte	was	to
1	$C / $3	$ØØ	$68	$6Ø

Patrick Ford's softkey for...

Jingle Disk

Thoughtware, Inc.
2699 South Bayshore Drive
Coconut Grove, FL 33133

Requirements:
Super IOB 1.5
A blank disk

Jingle Disk is a Christmas holiday musical story and card maker from Thoughtware that retails for approximately ten dollars.

Jingle Disk has a modified DOS that looks for a data field epilogue of AA DE EB instead of the normal DE AA EB. In addition, the VTOC and catalog have been moved. So while it is easy to deprotect the disk and copy with any nibble copier, the catalog cannot be read with normal DOS.

1) Put the controller below into Super IOB. This controller will use the altered epilogues to read the disk and write with normal ones. During the copy process, it will make the following sector edits so their DOS will be able to read the new format.

2) Copy the disk with Super IOB. Put both the original and copy away until December.

controller

```
1ØØØ REM JINGLE DISK CONTROLLER
1Ø1Ø TK = Ø : LT = 35 : ST = 15 : LS = 15 : CD = WR : FAST
     = 1
1Ø2Ø RESTORE : GOSUB 17Ø : GOSUB 49Ø : GOSUB 61Ø
1Ø25 T1 = TK : TK = PEEK (TRK ) - 1 : GOSUB 31Ø : TK
     = T1
1Ø3Ø 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 ,17Ø ,222
5Ø1Ø DATA 2^ CHANGES
5Ø2Ø DATA Ø ,1 ,227 ,222
5Ø3Ø DATA Ø ,1 ,237 ,17Ø
```

controller checksums

```
1ØØØ  - $356B      1Ø5Ø  - $67C8
1Ø1Ø  - $2544      5ØØØ  - $7FC2
1Ø2Ø  - $73BØ      5Ø1Ø  - $1B35
1Ø25  - $Ø5A4      5Ø2Ø  - $BF8D
1Ø3Ø  - $3E1B      5Ø3Ø  - $53B1
1Ø4Ø  - $CE2Ø
```

Danny Pollak's softkey for...

Crystal Caverns

Hayden Software
6ØØ Suffolk Street
Lowell, MA Ø1854

Requirements:
Apple][Plus or equivalent
Crystal Caverns
One blank disk
Muffin from DOS 3.3 System Master

readers' softkey & copy exchange

Crystal Caverns is a (old) text adventure in which the object of the game is to find the treasures and stash them. The protection scheme uses a version of DOS 3.2 which has been modified to read address headers of BB AA B5. We can use a modified version of Muffin to deprotect this program.

The softkey for Crystal Caverns is as follows:

1) Boot up DOS 3.3 and initialize the blank disk.

```
INIT HELLO
DELETE HELLO
```

2) Insert your System Master and BLOAD MUFFIN.

```
BLOAD MUFFIN
```

3) Enter the monitor and modify Muffin to read an address header of BB AA B5.

```
CALL -151
1A76:BB
```

4) Execute Muffin by typing:

```
803G
```

5) Now all that must be done is to transfer the files from the original disk to the initialized disk. When you run the program, you should notice a significant decrease in the time it takes for the program to load. Happy Adventuring!

Philip Goetz's softkey for...

Karate Champ

Data East USA, Inc.
470 Gianni St.
Santa Clara, CA 95054

Requirements:
48K Apple with one disk drive
Karate Champ
A blank disk
A sector editor
COPYA

You may have seen this game in the arcades. Two opponents, either of which may be controlled by the computer or a human, try to flatten each other as often as possible. Unfortunately, none of my nibble copiers seemed to be up to copying it.

Inspection of the disk with a nibble edit program shows that Karate Champ has very short sync gaps of about 7 bytes in length. Locksmith in default mode could not even recognize them as sync gaps. I suspect that these sync bytes may have more 0 bits tacked on the end than the normal 111111110 0 sync byte, thus a nibble copy cannot get into sync before reading a sector.

The nibble editor also shows that Karate Champ has normal address and data headers, with both epilogues being FF FF EB, rather than DE AA EB. Thus we can copy it with any program designed for unprotected disks by first telling DOS to ignore bad checksums and epilogues.

```
CALL-151
B942:18
RUN COPYA
```

When you boot the copy, it will access the disk and reboot. By removing the drive cover, watching where the drive head is at just before the reboot, and comparing it to the locations that the head is at when reading from various tracks with a sector editor, we see that the head was on track $11. (See "Getting on the Right Track" in COMPUTIST No. 5 for a detailed explanation). Examining track $11 with a nibble editor shows, between two sectors, a string of $E7 bytes. Surely a nibble count routine is looking for these $E7s and not finding them. So we search the disk for C9 E7, which disassembles to CMP #$E7. This is on track 0 sector 5, along with a lot of disk access code. By inspecting the code on track 0, sector 5, we see that it appears to start at the beginning of the page.

So, we simply change bytes 0 and 1 of that sector from $A0 00 (LDY #$00) to $18 60 (CLC followed by RTS) and write it back to the disk. Voila!

Here are the steps to unlock Karate Champ.

1) Boot a normal disk and modify DOS to ignore checksum and epilogue errors.

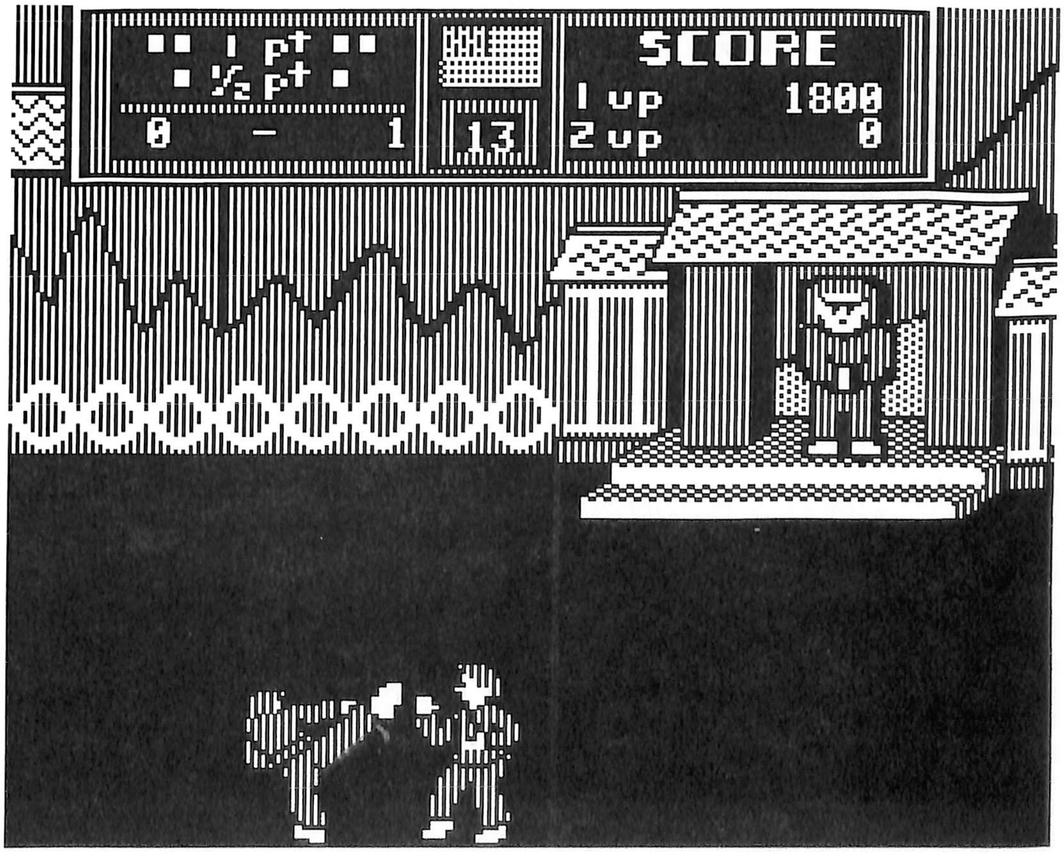
```
CALL-151
B942:18
```

2) Copy Karate Champ to a blank disk using a COPYA-type program.

3) Make the following change to the copy with a sector editor.

Track	Sector	Byte	From	To
0	5	$00	$A0	$18
0	5	$01	$00	$60

Finished!



softkey for...

Trivia Fever

by Fred Monroe

Professional Software, Inc.
P.O. Box 533
Needham, MA Ø2194
$39.95

Requirements:
An Apple][Plus or better
Trivia Fever disk
Ability to RESET into monitor (I used The Senior Prom)
A blank double-sided disk
A sector editor
A disk search utility

Trivia Fever is a game very much like the popular board game Trivial Pursuit (and THEY call US pirates). When you first boot the disk you will see the title page, then the usual boring "All rights reserved don't you dare touch this disk" messages. Next you will be asked if you want directions, and finally it will tell you to flip the disk over and press the space bar so it can load the questions. After some really annoying music and some "impressive" flashing between the text and lo-res pages you can finally play the game.

Because the game used files and an apparently normal DOS, I figured it would be your typical 5 minute job. I was pleasantly surprised to find out that I was wrong.

The first thing I always do when working with a disk is to watch it boot. I have the cover on my drive removed so that I can adjust the drive speed for nibble copies, and so I can tell which tracks a disk is accessing. When I booted my copy of the original side of Trivia Fever and watched it boot, the first thing that it did was jump to track 3, analyze that and crash out. Using my Senior Prom nibble viewer I looked at track 3. It appeared to be just a lot of self-sync bytes ($FF) with a few imbedded values, which is probably what the protection was looking for. There is a bright side to all of this, however. The disk check is done before the DOS is loaded in and the Hello program is run, so we can just bypass this part by copying everything except the DOS, on side 1, and putting our own DOS on.

1) INITialize both sides of a disk. Side one needs the filename "HELLO".

2) Use something like Super IOB (plain controller) or Copy][Plus to copy tracks $4-$22 of side one of Trivia Fever to side one of the new disk. Copy tracks $Ø-$21 of side two.

Now you've got a COPYAable disk with the slight drawback that it won't work. When I booted mine in this state I got a syntax error at 673. When I listed the program all I got was garbage. It appears that the HELLO program is encrypted. I will save you some time by telling you that the only encrypted programs are the Applesoft ones. There are two of them, "HELLO" and "TF". Here's how to capture them.

3) Boot side 1 of your original disk.

4) Reset into the monitor after the title page disappears (DOS is in memory).

5) Clear the "run flag" ($D6) and re-enter BASIC through DOS.

```
D6:ØØ
3DØG
```

For those of you who might not know, $D6, the run flag, controls the state of a BASIC program in memory. When the high bit is set at $D6 (i.e. a value of $8Ø or more) entry to BASIC will cause the program to execute immediately. It will also cause all statements like LIST, NEW, etc. to run the program.

6) Now you must clear memory and load in the HELLO file with Trivia Fever's own DOS.

```
NEW
LOAD HELLO
```

Fortunately, we don't have to get a normal DOS in memory because the SAVE command doesn't encrypt the program. Insert the copy disk and type

```
SAVE HELLO
```

Repeat this step for the file "TF".

I'm going to take this opportunity to warn you of one of the nasty little tricks that the protection on this disk likes to pull. If you ever see something like this flash onto your screen:

YOU HAVE A PROBLEM
-KING OF TRIVIA

then you should immediately hit RESET, pull your computer's plug, etc., because this is the program's little way of letting you know that it knows you've been messing around and it will proceed to erase tracks $ØØ-$Ø2 (your DOS).

That is exactly what happened to me when I tried to boot the copy in its present state. I was able to make it to the second side where the disk had asked for my name, and names of anyone playing, the subjects I wanted questions on etc. Just as it was searching the disk for a question I got the above message. The disk check actually happens long before this point, back when you are flipping your disk over.

In looking for the protection subroutine I decided to take advantage of the fact that whoever had produced this game had decided to be as annoyingly cute as a Care Bear and sent me the "KING OF TRIVIA" message. In fact, those are the exact words that I used in my Senior Prom ASCII search of the disk (any disk searcher will do). Lo and behold the offending routine turned up at $95BB.

I used the memory search function of the S.P. again and found a JMP $95BB at two locations, $929A and $94BB. They looked like this:

```
94A6- SEI            ;JSR to DOS'
94A7- JSR $BDØØ      ;sector read subr.
94AA- BCS $94B2      ;if there's an error
.                    ;then goto $94B2
                     ;
.
94B2- PLP            ;Get P-register
94B3- JSR $FF3A      ;Ring the bell
94B6- DEC $94DD      ;decrement error cntr.
94B9- BNE $94A1      ;Not Ø? Then try
94BB- JMP $95BB      ;again, else crash
```

7) So it would seem that this routine is an error handler. If an I/O error is encountered a certain number of times in reading the drive then it jumps to the "King of Trivia" message. It may not be necessary to change this routine, but better safe than sorry. Use a disk search utility to find it. I found it at track $12, sector $ØB:

```
change bytes $7B-$7D  from:CE DD 94
                        to:4C A1 94
```

Now it will continue trying to read the disk without the protection subroutine if a number of read errors occur. This next routine seemed to be the more important of the two. I think this is the one I was having trouble with.

```
9292- JMP $94DE     ;End last routine
9295- LDA $94DC     ;Get value
9298- BPL $929D     ;If positive then
929A- JMP $95BB     ;skip over protection
929D- LDA $D5       ;routine.

etc.
```

This seems easy enough to deal with. Just NOP the JuMP to $95BB. However, when I used my Senior Prom's disk search utility I was unable to find it. Because I couldn't find the second subroutine I decided to check elsewhere. I searched for the hex values DC 94 from the line LDA $94DC above. They turned up at a couple of places around $95EØ. This turned out to be a "nibble count" routine. The part that we're concerned with, though, is at the end of this nibble count routine.

```
958B- LDA $CØ8C,X  ;look at data latch
958E- BPL $958B    ;anything new? no, rtn
959Ø- PHA          ;otherwise push data
9591- PLA          ;pull it back
9592- CMP #$FF:    ;is it $FF?
9594- BEQ $958B    ;Yes, go to $95BB or
9596- CMP #$D5     ;no: Is it $D5?
9598- BNE $95B2    ;no, goto $95B2. or...
959A- LDA $CØ8C,X  ;yes get another byte
959D- BPL $959A    ;data yet? no, go back
959F- CMP #$FF     ;is it $FF?
95A1- BNE $95B2    ;no, goto $95B2
95A3- SEC          ;SEC for subtract
95A4- LDA $94DB    ;Get a count
95A7- SBC #$1Ø     ;Does it = $1Ø?
95A9- BNE $95B2    ;no, go to $95B2 or...
95AB- STA $94DC    ;STA at $94DC (see ML
95AE- LDA $CØ88,X  ;above) and turn off
95B1- RTS          ;drive motor and RTS.
95B2- LDA $CØ88,X  ;Turn off motor.
95B5- LDA #$FF     ;LDA with a NEGATIVE
95B7- STA $94DC    ;STA at $94DC (see
95BA- RTS          ;above) and RTS.
```

Whenever things don't match up quite right it seems to JuMP to $95B2, which loads an $FF in the Accumulator and stores it at $94DC. The same value that the second subroutine above uses to determine whether it should jump to the "King of Trivia" routine. The first thing that I tried doing was having this program load a $#ØØ instead of an #$FF at the end. That ended up causing problems, because when you got past a certain number of questions the program would lock up in a loop trying to read the disk. Apparently some of the questions are on tracks that are supposed to leave the nibble count routine via $95B2. However, we do know that:

a) The routine at $9295 requires $94DC to be positive (less than 128).

b) The first time that the nibble count is accessed, after flipping the disk, it will exit through $95B2 because things don't match up on the copy.

c) The "King of Trivia" routine, code that we definitely don't need, is located at $95BB just after the nibble count.

As I told you earlier, I was unable to find the routine located at $9295 in memory. I was, however, able to find the nibble count and the "King of Trivia" routines. They are both located (on my disk) on side 1, track $12, sector $A. "So," I said to myself, "why not just tack on a little more code at the end of the nibble count that will wipe out the JuMP at the $9295 routine. I know that it will be executed before the program ever gets to that routine so it should work." Unlike a number of things that I think should work, this did. Here's what I did:

8) Search the disk for the end of the nibble count. The last RTS is located at track $12, sector $A, byte $7F.

9) At that point I keyed in the following:

```
A9 EA 8D 9A 92 8D 9B 92 8D 9C 92 6Ø
```

Which translates in assembly as:

```
LDA #$EA   ;Hex code for NOP
STA $929A  ;store the value
STA $929B  ;at the locations for
STA $929C  ;the JMP $95BB
RTS        ;and continue as normal
```

I thought that this is all I would need to do but there is still one more bridge to cross. If you were to boot the disk at this point it would begin to read in some of the data but ends up stuck in a loop spinning the disk. The reason for this can again be found in the nibble count routine. Here is the code we have to alter:

```
955Ø- LDY #$ØØ        ;LDY #Ø and store at
9552- STY $94DB       ;nibble ctr. ($94DB)
9555- LDA $CØ8C,X     ;Go to disk for a byte
9558- BPL $9555       ;
955A- CMP #$D5        ;is it $D5?
955C- BEQ $956D       ;Yes goto $956D or...
955E- CMP #$F7        ;is it $F7?
956Ø- BNE $9563       ;if no then don't INY
9562- INY             ;in either case add to
9563- CLC             ;value at $94DB & save
9564- ADC $94DB       ;new value to same
9567- STA $94DB       ;location.
956A- JMP $9555 ***;Start over at $9555
956D- TYA             ;Y=Ø if no $F7's found
956E- BEQ $9555 ***;and in that case start
                      ;over at $9555
```

10) Obviously, on a normal disk there aren't going to be F7's in the self sync fields (which is what this routine checks) so it is stuck in a loop because things don't match up. What we do is change the two lines indicated with asterisks above. Just NOP the JuMP (EA EA EA) and set the branch distance to ØØ (BEQ $956E, or FØ ØØ). This routine should be in the same sector as the end of the nibble count was above. In my case, track $12, sector $A. Here's what I did:

```
Side 1, track $12, sector $A
        Bytes $2F-31  From:4C 55 95
                        To:EA EA EA
        Byte $34      From:EØ
                        To:ØØ
```

Just write this back to your disk and you will now have a fully deprotected Trivia Fever disk.

If I were you I would write protect both sides because of the nature of the protection and since the original comes write protected.

The Trivia Fever volume 2 disk can be copied just as if it was side 2 of the original Trivia Fever disk (i.e. tracks $Ø-21 normal; no DOS necessary).

Cookbook Version

1) INIT both sides of a disk with your favorite DOS using the filename HELLO.

2) Copy tracks $4-$22 of Trivia Fever side 1 to the blank disk's side 1. If using Super IOB let TK=4 and LT=35 in line 1Ø1Ø of a standard controller.

Copy tracks $Ø-$21 of side 2. (TK=Ø and LT=34.)

3) Boot your original Trivia Fever disk side one.

4) Reset into the monitor after the title page is up.

5) Disable the run flag and enter BASIC.

D6:ØØ
3DØG

6) Get the Applesoft files HELLO and TF in their decoded forms and save them to the copy.

NEW
LOAD HELLO

Insert the copy disk.

SAVE HELLO

Re-insert the original.

NEW
LOAD TF

Insert the copy.

SAVE TF

7) Perform the following sector edits on side one:

```
---------------------------------------
Track $12, Sector $B, bytes $7B-$7D
FROM- CE DD 94
  TO- 4C A1 94
---------------------------------------
Track $12, Sector $A, bytes $7F-$8A
FROM- 6Ø AD 54 CØ AD 51 CØ AD 81 CØ 2Ø 58
  TO- A9 EA 8D 9A 92 8D 9B 92 8D 9C 92 6Ø
---------------------------------------
Track $12, sector $A, bytes $2F-$31
FROM- 4C 55 95
  TO- EA EA EA
---------------------------------------
Track $12, Sector $A, byte $34
FROM- EØ
  TO- ØØ
---------------------------------------
```

While I have your attention, I'd like to say that for any of you who don't already have a means of dropping into the monitor at will or a memory search utility, I whole-heartedly recommend the Senior Prom. It has so many useful utilities due to the ROM's RWTS that it's worthwhile even if you don't softkey disks. Give their modem line a call at (313)-349-2954 and tell them Fred sent you.

The Original Boston

by Clay Harrell

Scarborough Systems, Inc.
25 N. Broadway
Tarrytown, NY 1Ø591

Requirements:
Any Apple][or clone
A sector editor
COPYA or other normal DOS copy program
4 blank disks
The Original Boston Computer Diet

The Original Boston Computer Diet (OBCD from here on) is an excellent computer-based diet system. The documentation provided is very extensive and well written, and provides much dietary information. I highly recommend this program to anyone with an Apple computer and who is a concerned about their diet (whether they are overweight or not).

The problem with the OBCD is it can only handle one individual's diet at a time. Since it is copy protected, you can not make a second copy and let another member of the family use the backup disks for their diet needs. None of the powerful bit copy programs could produce a working copy either. Hence this softkey was developed.

All four sides of the OBCD copy with COPYA or Locksmith 5.Ø Fastcopy. Of course, the copied program will not run. Instead, after booting you receive the message "Defective Disk", memory clears, and the program halts.

After booting the Copied OBCD several times, I was able to time where the Disk Check was occurring, and interrupt the program before receiving the "Defective Disk" message. Using the Senior PROM's "D" command after interrupting the program (which gives a disassmebly of ten instructions before and after where the program is currently executing), I could easily see a check disk routine was at location $77A2. For the curious, here is the code:

```
77A2- LDA $B7EC    ;Save the current track
77A5- PHA          ; on the Stack.
77A6- LDA $B7F4    ;Save current RWTS
77A9- PHA          ; command on the Stack.
77AA- LDA $BE4D    ;Save location $BE4D
77AD- PHA          ; on the Stack.
77AE- LDA #$ØE     ;
77BØ- STA $B7EC    ;Set to read track $ØE.
77B3- LDA #$ØØ     ;Set RWTS command
77B5- STA $B7F4    ; to $ØØ (seek).
77B8- LDA #$6Ø     ;
77BA- STA $BE4D    ;Put an RTS at $BE4D.
77BD- LDY #$E8     ;Use the RWTS Parm List
77BF- LDA #$B7     ; starting at $B7E8.
77C1- JSR $B7B5    ;Seek head to track $ØE.
77C4- PLA          ;Pull original values
77C5- STA $BE4D    ; of $BE4D, command, and
77C8- PLA          ; track from the Stack
77C9- STA $B7F4    ; restore them to their
77CC- PLA          ; previous states.
77CD- STA $B7EC    ;
77DØ- LDA $CØ8C,X  ;Read a byte from disk.
77D3- BPL $77DØ    ;Repeat until MSB = 1.
77D5- PHA          ;Push and pull this byte
77D6- PLA          ; on Stack (timing!).
77D7- CMP #$D5     ;Is it a $D5?
77D9- BNE $77DØ    ;If not found, go back.
77DB- LDY #$ØØ     ;Clear Y register.
77DD- STY $783A    ;Store a $ØØ at $783A.
77EØ- LDA $CØ8C,X  ;Read a byte from disk.
77E3- BPL $77EØ    ;Repeat until MSB = 1.
77E5- CMP #$D5     ;Is it a D5?
77E7- BEQ $77F8    ;If so goto $77F8. This
                   ; is the first byte
                   ; we are looking for.
77E9- CMP #$F7     ;Is it a $F7? (2nd byte)
77EB- BNE $77EE    ;If not, goto $77EE.
77ED- INY          ;If so, increment Y-reg.
                   ;This is very important,
                   ; since if no $F7 is
                   ; found, Y is never
                   ; INCed and the routine
                   ; continues in a loop.
77EE- CLC          ;Prepare for adding.
77EF- ADC $783A    ;Add $F7 to $783A.
77F2- STA $783A    ;Store sum at $783A.
77F5- JMP $77EØ    ;Repeat search for $F7's
77F8- TYA          ;Transfer Y-reg to Acc.
77F9- BEQ $77DB    ;If Acc = $ØØ
  .                ; (unsuccessful search)
  .                ; start over and look
  .                ; for a $D5 F7 again.
;Else, continue with program...
```

In other words, this routine saves the current parameters of the RWTS (current track and command code), seeks the read/write head to track $ØE, and then looks for the byte sequence $D5 F7.

To test my theory about the Check Disk routine, I put a $6Ø (RTS instruction) at location

Computer Diet

$77F2, and restarted the program with the Senior PROM. Sure enough, the program restarted and ran perfectly, just as if the original disk was in the drive.

So all I needed to do was to put a $6Ø (RTS) at location $77A2 on the disk with a sector editor, and the OBCD would be deprotected!

Now the bad news. Searching through all the disks for the code at $77A2 with the Senior PROM's disk search utility brought nothing. Obviously, the OBCD anticipated this action and thoughtfully encoded the routine on the disk.

As far as I could tell, the OBCD used RWTS to load in the code, this meant that each page in memory ($1ØØ hex bytes or 256 decimal bytes) represented a sector on the disk. Since the Disk Check code was somehow encoded, I decided to search the disk for the byte sequence at location $77ØØ, the beginning of the page where the Disk Check routine resided. Hopefully *this* code wasn't encoded. This code was found on track $Ø8, sector $3, side A of the OBCD. Great! Now we just need to figure out how the code was encoded and then make the appropriate change. I found the encoding routine at location $7783 (right before the Disk Check routine):

```
7783-  38           SEC
7784-  6E 87 77     ROR $7787
7787-  AØ F2        LDY #$F2
7789-  6E 8C 77     ROR $778C
778C-  6E 98 77     ROR $7798
778F-  6E 92 77     ROR $7792
7792-  6E 99 77     ROR $7799
7795-  6E AØ 77     ROR $77AØ
7798-  98           TYA
7799-  59 A1 77     EOR $77A1,Y
779C-  99 A1 77     STA $77A1,Y
779F-  88           DEY
77AØ-  DØ F6        BNE $7798
```

Since this routine was obviously written to confuse hackers, I took the brute force approach and just tried changing the code at $77A2 wholesale (change 1Ø bytes to $ØØ's). Then I rebooted the program, interrupted it and examined the results.

The code from $77A2 was unchanged, no matter what I changed the code on track $Ø8, sector $Ø3 to. Back to the trail, since I had to be changing the wrong sector!

My next approach was to use a track monitor and watch when track $Ø8 was being accessed. When track $Ø8 appeared in the track monitor, I interrupted the program (via the Senior PROM), and checked location $B7F1 (command code, 1=read, 2=write), $B7EC (the current track being read), location $B7ED (the current sector being read), and location $B7F1 (the current page being read into). Sure enough track $Ø8, sector $Ø3 was being read into $77ØØ-77FF. I then restarted the program and let it continue to load.

Once again using the Track Monitor and the Senior PROM, when track $Ø8 was being accessed I interrupted the program. This time I found that track $Ø8, sector $Ø4 was being read into $77ØØ-77FF! This must be it...

Loading track $Ø8, sector $Ø4 with the Senior PROM sector editor brought up a very strange looking sector. Regardless, I changed locations $A2-AF of this sector to $ØØ's (I changed more than 1 location just to make finding the change in memory easier) and re-booted the program. The program bombed into the monitor at location $77F4 because the code encountered the $ØØ's (a Break instruction).

It's real strange that a change in the sector at byte $A2 would effect the code at byte $F4! For more strangeness, three bytes have to be changed in the sector to effect one byte in memory! After much experimentation with editing track $Ø8, sector $Ø4 and then viewing the effects in memory, I found that replacing bytes $6Ø-62 from the original $83 77 71 to $6Ø 6Ø 6Ø would put a single $6Ø (Return from Subroutine instruction) at location $77A2 in memory. That's the only change needed to deprotect the OBCD!

Cookbook

1) Boot normal DOS 3.3 and run your favorite normal DOS copy program (i.e. COPYA, Locksmith 5.Ø Fast Copy, etc.).

2) Copy all four sides of the Original Boston Computer Diet to some blank disks.

3) Run your favorite sector editor and make the following change to your copy of the OBCD:

```
Track $Ø8, sector $4, byte $6Ø, side A
           From  $83 77 71
           To    $6Ø 6Ø 6Ø
```

4) Write the sector back out to your copy of the OBCD. And you're all done!

A little help with..

The Bard's Tale

by Ben Youngdahl

Requirements:
The Bard's Tale
A good sector editor
A premade party on a character disk
A blank disk for backing up your characters

Face it, your party is a bunch of wimps compared to what The Bard's Tale is going to throw at you. If you've been to the castle and seen how the Old Man fights, you know they're out to get you. Why should you be forced to spend gold needlessly to cure a party of hurt wimps when you'd rather use it to buy new weapons?

If you have been wondering if it was time to find an easier hobby like moving buildings, I have just the thing for you. Read on to discover how to make your party as powerful as you wish.

How I Found Out

I was about to give up the game entirely when I decided to look through the disk to see if I could find anything useful. I'm glad I took the few minutes to boot up the sector editor and start looking around because on track $ØØ of the character disk there are some names that looked too familiar to be coincidental.

The next thing I did was examine the bytes following the name. It appeared to me that each sector had two characters on it, and a lot of data concerning the characters as well. It looked like it was going to take some trial and error (T & E) to sort all this out.

The first part was easy; the names seemed to be stored on bytes $ØØ to $ØF of the sector. Unused spaces were indicated by FF's, and the names were in standard text format as well.

Bytes $1Ø to $13 turned out after T & E to be the ability scores. The following bytes through $1F represented the experience of a character, I soon learned, and the next two bytes still remained a mystery for me.

For the first time so far I ran into a problem; I was not able to discover the use of bytes $22 and $23. After a while it became evident the characters had progressed a lot of levels and I had not noticed. I concluded that bytes 22 and 23 dealt with the level status.

Bytes $24 to $2F deal with the amount of gold you possess; it seemed to follow the same lines as experience points. A little farther on, I found that this was one of the easier discoveries.

I changed bytes $3Ø through $37 to FF's and received a higher hit point max and a higher amount of spell points. Not until later did I discover that $3Ø-$33 dealt specifically with hit points and $34-$37 dealt only with spell points.

Class status was dealt with by byte $38. So after some T & E I came up with Table I. Your race turned out to be held in byte $39. This is outlined in Table II.

Soon after one of my characters died (yes, even with 9,999 hit points!) I searched for the byte that would allow me to achieve immortality. In a way I found it. Byte $3E, when changed to ØØ from whatever it was before, will bring your character back to life. I tried this and now Joe the Bard is safe and sound.

Still, a few things that I felt needed to be done had not been accomplished. I still had rotten weapons and armour, and did not have the number of spells I needed to survive.

I found that bytes $48 to $4B controlled the number of spells allowed, so I changed them all to FF's and achieved total spell capability.

At long last, I finally discovered how the computer stored weapon data for your character. This took quite a bit of time because it was surprisingly hard to figure out that weapon data is stored in pairs starting at $5Ø. The first byte in the pair represents the status of the weapon (or armour or whatever), and the second byte is the number of the item, as in Table IV.

It took me an hour to come up with table IV, and it was an hour spent with constant discovery. Many of these items I had not even heard about before. I was very anxious to try them out so there may be flaws in my data.

After three days of on and off hacking I finally had finished what I had begun. I have created a super party but still have not won. On level three of the castle I can't find the stairs up. Someone help me!

How To Make Your Super Party

Make a back-up of your character disk with any normal copy program. Then take out your best sector editor and read in track $Ø sector by sector until you see the name of your character in ASCII. If it is in the upper half of the sector, use the byte numbers in the first column. If it is in the lower half, use the byte numbers in parentheses. All numbers in the tables are in hex. Make the changes you require to produce the effect. For example:

Joe the Bard has only 1Ø hit points. To make that 9,999+ change bytes $3Ø-33 to $FF FF FF FF. If Joe the Bard is on the bottom half of the sector you would change $BØ-B3 to $FF FF FF FF.

I hope you enjoy this game more with the added plus of a great party. I find it more interesting and more entertaining to make it past some of those guardian statues now!

Table Ø

Byte(s)	Effect
ØØ-ØF (8Ø-8F)	Name
1Ø-13 (9Ø-93)	Ability Scores
14-1F (94-9F)	Experience Points
22-23 (A2-A3)	Level
24-2F (A4-AF)	Gold
3Ø-33 (BØ-B3)	Hit Points
34-37 (B4-B7)	Spell Points
38 (B8)	Class (Table I)
39 (B9)	Race (Table II)
3E (BE)	Status (ØØ=O.K.)
4Ø-43 (CØ-C3)	Spells
5Ø-5F (DØ-DF)	Items held:

-Even bytes (5Ø,52,...) contain the status of the item (Table III)
-Odd bytes (51,53,...) contain the number of the item (Table IV)
-Order of byte pair is status first, object number second.

Table I

ØØ	Warrior
Ø1	Wizard
Ø2	Sorcerer
Ø3	Conjurer
Ø4	Magician
Ø5	Rogue
Ø6	Bard
Ø7	Palidin
Ø8	Hunter
Ø9	Monk

Table II

ØØ	Human
Ø1	Elf
Ø2	Dwarf
Ø3	Hobbit
Ø4	Half-Elf
Ø5	Half-Orc
Ø6	Gnome

Table III

ØØ	Object is there
Ø1	Object is equipped
Ø2	Object is unusable (but not if changed to Ø1)
Ø3	Object is unknown

Table IV

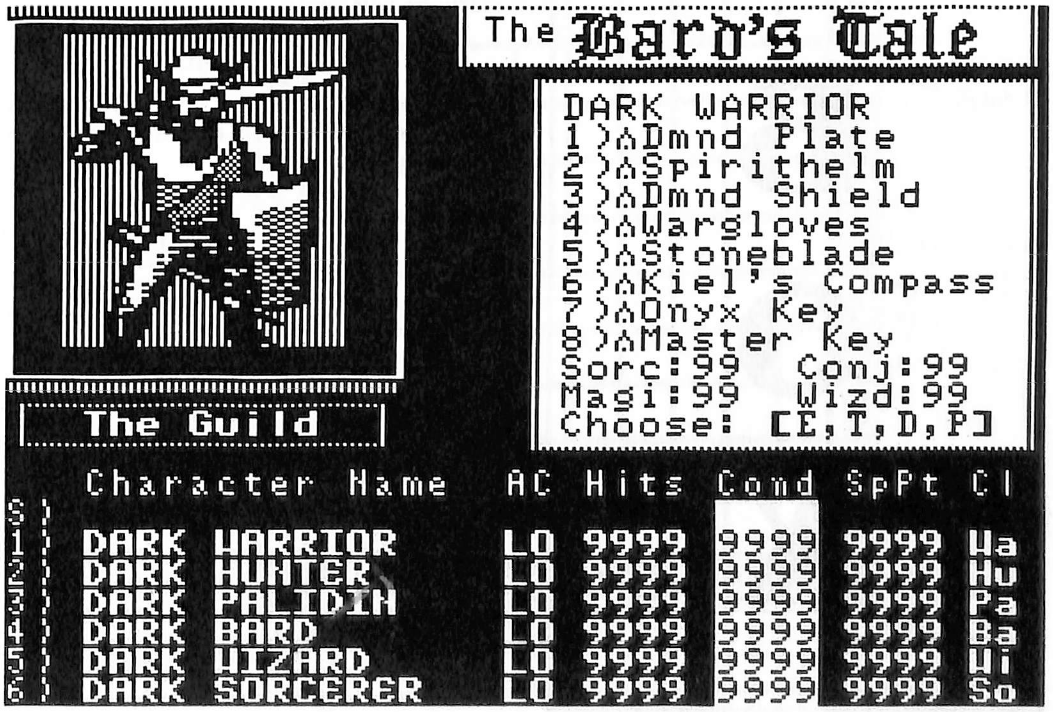
ØØ	Nothing
Ø1	Torch
Ø2	Lamp
Ø3	Broadsword
Ø4	Short Sword
Ø5	Dagger
Ø6	War Axe
Ø7	Halbard
Ø8	Mace
Ø9	Staff
ØA	Buckler
ØB	TowerShield
ØC	Leather Armour
ØD	Chain Mail
ØE	Scale Mail
ØF	Plate Mail
1Ø	Robes
11	Helm
12	Leather Gloves
13	Gauntlets
14	Mandolin
15	Harp
16	Flute
17	Mthr Sword
18	Mthr Shield
19	Mthr Chain
1A	Mthr Scale
1B	Samurai Fgn
1C	Bracers (6)
1D	Bardsword
1E	Fire Horn
1F	Lightwand
2Ø	Mthr Dagger
21	Mthr Helm
22	Mthr Gloves
23	Mthr Axe
24	Mthr Mace
25	Mthr Plate
26	Ogre Fgn
27	Lak's Lyre
28	Shield Ring
29	Dork Ring
2A	Fin's Flute
2B	Kael's Axe
2C	Blood Axe
2D	Dayblade
2E	Shield Staff
2F	Elf Cloak
3Ø	HawkBlade
31	Admt Sword
32	Admt Shield
33	Admt Dagger
34	Admt Helm
35	Admt Gloves
36	Admt Mace
37	Broom
38	Pureblade
39	Exorwand
3A	Ali's Carpet
3B	Magic Mouth
3C	Luckshield
3D	Giant Fgn
3E	Admt Chain
3F	Admt Scale
4Ø	Admt Plate
41	Bracers (4)
42	Arcshield
43	Pureshield
44	Magestaff
45	Warstaff
46	Thief Dagger
47	Soul Mace
48	Wither Staff
49	Sorcer Staff
4A	Sword of Pak
4B	Heal Harp
4C	Galt's Flute
4D	Frost Horn
4E	Dmnd Sword
4F	Dmnd Shield
5Ø	Dmnd Dagger
51	Dmnd Helm
52	Golem Fgn
53	Titan Fgn
54	Conjur Staff
55	Arc's Hammer
56	Staff of Lor
57	Power Staff
58	Mournblade
59	Dragon Shield
5A	Dmnd Plate
5B	War Gloves
5C	Lorehelm
5D	Dragonwand
5E	Kiel's Compass
5F	Speedboots
6Ø	Flame Horn
61	Truth Drum
62	Spirit Drum
63	Pipes of Pan
64	Ring of Power
65	Deathring
66	Ybarra Shield
67	Spectre Mace
68	Dag Stone
69	Arc's Eye
6A	Ogrewand
6B	Spirithelm
6C	Dragon Fgn
6D	Mage Fgn
6E	Troll Ring
6F	Troll Staff
7Ø	Onyx Key
71	Crystal Sword
72	Stoneblade
73	Travelhelm
74	Death Dagger
75	Mongo Fgn
76	Lich Fgn
77	Eye
78	Master Key
79	Wiz Wand
7A	Silvr Square
7B	Silvr Circle
7C	Silvr Triang
7D	Thor Fgn
7E	Old Man Fgn
7F	Spectre Snare

Some Hints

The word "skull" is something to keep in mind if you come across someone who is old enough to be one.

The eye will make a harmless statue become as mad as a god! I prefer to snare him with a spectre snare and bind him to the party.

The throne prefers bards. Bards can turn out most useful in the end!



```
The Bard's Tale
                    DARK WARRIOR
                    1)ΔDmnd Plate
                    2)ΔSpirithelm
                    3)ΔDmnd Shield
                    4)ΔWargloves
                    5)ΔStoneblade
                    6)ΔKiel's Compass
                    7)ΔOnyx Key
                    8)ΔMaster Key
                    Sorc:99   Conj:99
                    Magi:99   Wizd:99
  The Guild         Choose: [E,T,D,P]
  Character Name     AC Hits Cond SpPt Cl
S)
1) DARK WARRIOR      LO 9999 9999 9999 Wa
2) DARK HUNTER       LO 9999 9999 9999 Hu
3) DARK PALIDIN      LO 9999 9999 9999 Pa
4) DARK BARD         LO 9999 9999 9999 Ba
5) DARK WIZARD       LO 9999 9999 9999 Wi
6) DARK SORCERER     LO 9999 9999 9999 So
```

Probing the Unknown with...

BLACK BOX

by Ray Darrah

Requirements:
48K Apple][Plus or better
One disk drive
A few hours of typing time or library disk #31
The Animator from COMPUTIST No. 29

Black Box is a board game distributed by Parker Bros in which a player fires several "electrons" into an imaginary black box containing one or more "atoms." The player chooses the starting position (and therefore direction) of the electron to be fired. Sensors on the perfect-vacuum black box tell the player what happened to the electron. By observing what happens to electrons fired from several positions, the player is to deduce *where* exactly in the black box the atoms are located.

My first version of this game was created due to the encouragment of a teacher named Marvin Jacobson. My original program was very long and text oriented. After creating the "Animator" appearing in COMPUTIST No. 29, I decided to make a hi-res version of Black Box utilizing the capabilities of it.

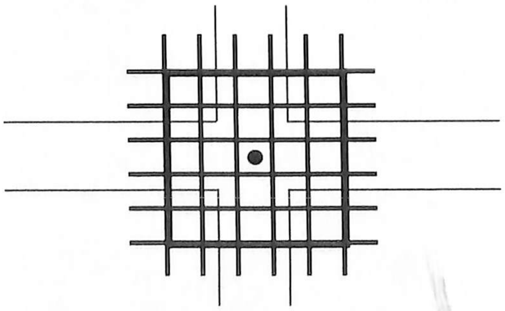
Of Atoms and Electrons

Electrons always interact with atoms in the same way. The following is an account of this interaction:

To simplify the game, the Black Box is divided into a ten by ten grid. You may fire an electron into the box from the outer edge. The electron will travel along a straight path unless it encounters an atom. When an electron encounters an atom, one of three possible reactions will occur.

Deflection

An electron is deflected when it encounters an atom placed diagonally from it. Consider the following diagram:

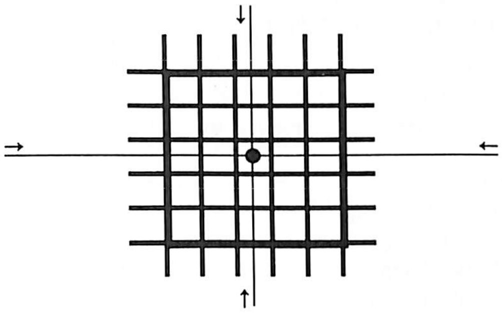


The heaviest black line designates the border of a small black box. The thick lines represent the black box's subdivisions and the dot represents an atom. The thin lines show the paths electrons would take if they were fired from various positions. This sort of "bouncing off of" is called deflection.

Markers placed around the very edges of the box are used to indicate where an electron entered the box and where it exited. Markers are also used for other indications. For a complete list of markers see the section entitled "Markers and Symbols."

Absorption

An atom can absorb an electron if the electron is fired along the same row or column as the atom. This is illustrated in the following diagram:



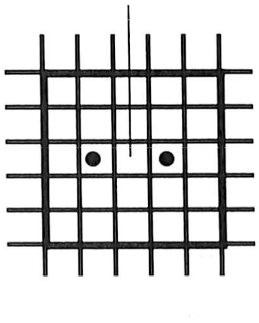
When an electron is aligned either horizontally or vertically with an atom, the atom absorbs the electron. That is, the electron enters the box but doesn't come out. A special marker is used to indicate such an occurrence.

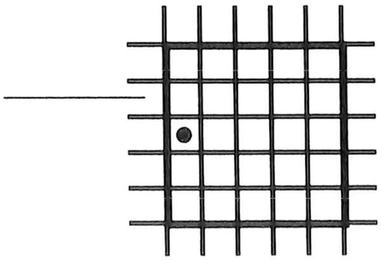
Absorption has precedence over deflection. That is, if two atoms are side by side (or one atop the other) and an electron is fired directly in line with either of the atoms, it will be absorbed rather than deflected off of the other atom.

Absorption also has precedence over reflection.

Reflection

Reflection is a 180° turn about of the electron and therefore the electron comes out of the box exactly where it entered. Two situations can cause a reflection. They are illustrated in the following diagrams:





In the first diagram, the atom on the left wants to deflect the electron to the right and the atom on the right wants to deflect the electron to the left. The result is that the electron is reflected and exits the box exactly where it entered.

In the second diagram, an atom has been placed right next to the border of the box. Since the electron has been fired one row above the atom, (one row below would also work) the atom would normally deflect the electron upward. But this deflection would occur before the electron has even entered the box. The result is a reflection.

Multiple Atomic Encounters

Keep in mind that when more than one atom is in the box, there is a potential for a multiple atomic encounter. That is, an electron could react to more than one atom.

For example, an electron could be deflected off of one atom and its new path could send it directly at another atom. In this case, an absorbed marker would be placed where the electron entered the box.

Markers and Symbols

Greek letters are used to identify electrons that enter the box and exit it somewhere else. Eight capital greek letters have been defined. They are (and appear in this order) omega, theta, lambda, pi, sigma, upsilon, phi and psi. After these eight letters have been used, the sequence starts over again with inverse greek letters. All of the greek symbols can be seen in the diagram at the bottom right of this page.

The symbol for absorption and reflection is also indicated in the diagram at the bottom right of this page.

Using the Program

Use "I, J, K and M" to move your electron gun (represented by a spy gun) around the edge of the black box. To fire into the box, use the space bar.

If you should move the electron gun into the box, your position will be indicated by a finger. Once inside the box you can designate where you think an atom is located by pressing the space bar. A question mark will then appear in that sector of the box. Pressing the space bar while pointing to an occupied space will make the question mark go away.

When you think you have solved the puzzle and know where all the atoms are hidden, press return. This activates the program's scoring and evaluation sequence. For every question mark you have correctly positioned, the computer will replace it with an atom shape. Every question mark that you positioned incorrectly will remain a question mark. An inverse atom will be drawn in the sectors where you should have had a question mark.

Scoring

The program considers how many atoms you were looking for and bases your score on how many non-absorbed and non-reflected shots it took to discover where the atoms were.

The program also considers the number of atoms you placed on the board incorrectly and affects your score in a negative manner for each incorrectly placed question mark.

Your score is displayed as a percentile. The best score you can get is 100%. Because incorrectly placed question marks affect your score in a negative manner, it is possible to get a negative score.

Modifications

As the program is now, you can hide a maximum of five atoms. You should find this quite adequate as five atoms in a ten by ten grid is sometimes very perplexing. However, by changing the "5" in lines 240, 250 and 260 to some other number (such as "9"), the program will let you hide more atoms.

The Parker Bros version of this game requires two players (or one player with a split personality). Player one hides the atoms while player two hunts them down. Player one also tells player two what happens to each electron that player two fires. My progam assumes all of the responsibilities of player one. However, you may occasionally want to involve someone else in the game. You could add a subroutine where another player would hide the atoms for you to hunt down.

Typing It In

Black Box is comprised of three parts. A BASIC program and two hexdumps. Key in the BASIC program using the instructions on the inside front cover of this issue and save it with:

```
SAVE BLACK BOX
```

Key in the object code hexdump (the second hexdump) using the instructions on the inside front cover of this magazine and save it with:

```
BSAVE OBJ.BLACK BOX,A$300,L$17
```

To save magazine space (and your fingers), the first hexdump (containg all the shapes used by Black Box) has been condensed. To key it in, first clear the memory where it resides with the following:

```
6000: 00
6001<6000.69C0M
```

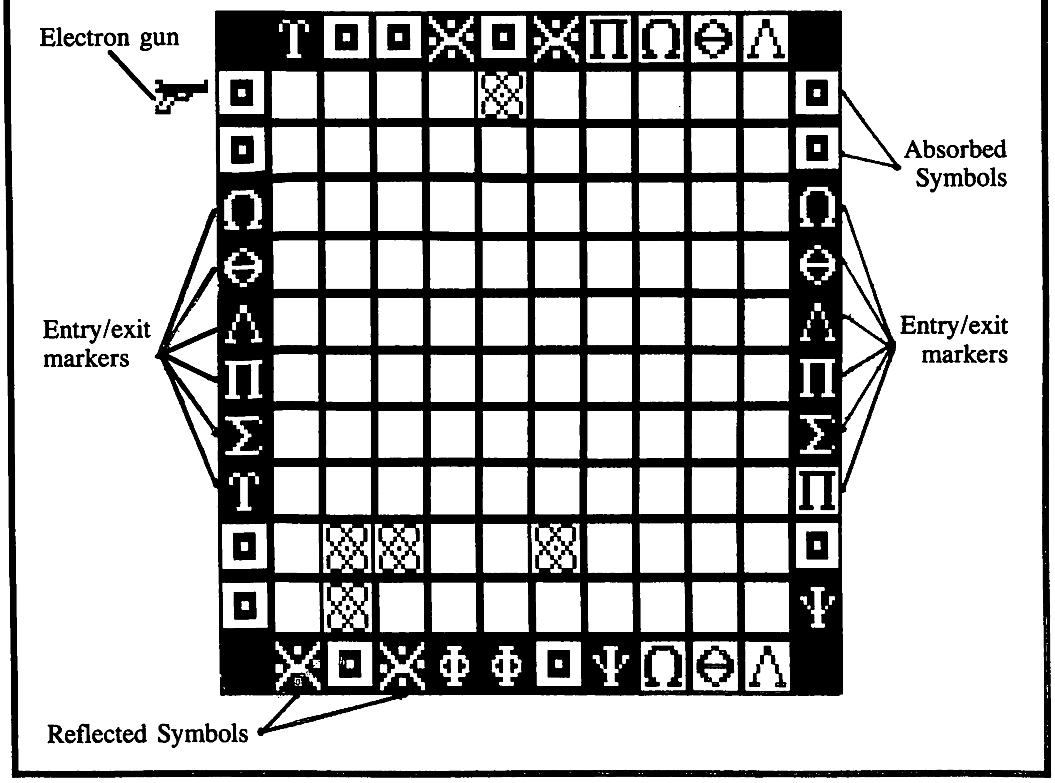
Next, key in the hexdump as you would normally. Note that the addresses of each line aren't consecutive. This is because large chunks of the hexdump containing only zeros have been omitted. These deletions occur only where the hexdump has a blank line in it. When you're finished typing in this hexdump (paying careful attention to the addresses at the left), save it with:

```
BSAVE SH.BLACK BOX,A$6000,L$9C0
```

Black box uses 26 shapes and each consecutive block of this hexdump is one shape. The last line of the hexdump was included so that you could easily match your final checksum.

The order of the shapes used by Black Box is: gun pointing right, gun pointing down, gun pointing left, gun pointing up, absorbed symbol, reflected symbol, pointing finger, omega marker, theta marker, lambda marker, pi marker, sigma marker, upsilon marker, phi marker, psi marker, inverse greek markers, question mark, normal atom symbol and inverse atom symbol.

Paths of electrons and Symbols:



Black Box Shape Hexdump

```
6000: 03 40 00 00 7E 7F 00 00   $1FA8
6008: 7F 7F 00 00 3C 06 00 00   $EFAB
6010: 64 03 00 00 12 00 00 00   $A160
6018: 09 00 00 00 0F 00 00 00   $568C

6060: 23 01 00 00 65 01 00 00   $F128
6068: 79 00 00 00 73 00 00 00   $98B4
6070: 74 00 00 00 78 00 00 00   $0E14
6078: 68 00 00 00 68 00 00 00   $BEF4
6080: 78 00 00 00 70 00 00 00   $7A74
6088: 60 00 00 00 60 00 00 00   $9AB4
6090: 60 00 00 00 60 01 00 00   $7F76

60C0: 01 60 00 00 7F 3F 00 00   $398B
60C8: 7F 7F 00 00 30 1E 00 00   $B774
60D0: 60 13 00 00 00 24 00 00   $7615
60D8: 00 48 00 00 00 78 00 00   $F6AD

6120: 60 01 00 00 60 00 00 00   $079F
6128: 60 00 00 00 60 00 00 00   $F7AF
6130: 70 00 00 00 78 00 00 00   $73FF
6138: 68 00 00 00 68 00 00 00   $D3EF
6140: 78 00 00 00 74 00 00 00   $2D8F
6148: 73 00 00 00 79 00 00 00   $500B
6150: 65 01 00 00 23 01 00 00   $DB87

6180: 7E 3F 00 00 7E 3F 00 00   $CB93
6188: 0E 38 00 00 0E 38 00 00   $07EB
6190: 4E 39 00 00 4E 39 00 00   $DF33
6198: 0E 38 00 00 0E 38 00 00   $534B
61A0: 7E 3F 00 00 7E 3F 00 00   $33AF

61E0: 02 20 00 00 46 31 00 00   $385D
61E8: 4C 19 00 00 18 0C 00 00   $635D
61F0: 30 36 00 00 66 33 00 00   $09C1
61F8: 76 07 00 00 18 0C 00 00   $0F3E
6200: CC 19 00 00 46 31 00 00   $B252
6208: 02 20 00 00 00 00 00 00   $4362

6248: 00 00 00 00 60 00 00 00   $B3E2
6250: 40 01 00 00 00 03 00 00   $3FE7
6258: 00 06 03 00 00 4D 01 00   $4BAE
6260: 00 5F 01 00 40 7F 01 00   $97A6
6268: 40 7F 01 00 00 7F 02 00   $0C3D
6270: 00 3E 03 00 00 5C 01 00   $CD22
6278: 00 60 00 00 00 00 00 00   $5D32

62A0: 00 00 00 00 70 07 00 00   $88DF
62A8: 18 0C 00 00 0C 18 00 00   $5A05
62B0: 0C 18 00 00 0C 18 00 00   $02D5
62B8: 0C 18 00 00 0C 18 00 00   $DA05
62C0: 18 0C 00 00 3C 1E 00 00   $4830

6300: 00 00 00 00 40 01 00 00   $ED32
6308: 30 06 00 00 18 0C 00 00   $A99F
6310: 0C 18 00 00 7C 1F 00 00   $4422
6318: 0C 18 00 00 18 0C 00 00   $2E30
6320: 30 06 00 00 40 01 00 00   $F3C1

6360: 00 00 00 00 40 01 00 00   $56C3
6368: 40 01 00 00 20 02 00 00   $1F94
6370: 20 02 00 00 10 04 00 00   $074F
6378: 10 04 00 00 08 08 00 00   $CB69
6380: 08 08 00 00 1C 1C 00 00   $D93B

63C0: 00 00 00 00 7C 1F 00 00   $82FA
63C8: 18 0C 00 00 18 0C 00 00   $2242
63D0: 18 0C 00 00 18 0C 00 00   $827A
63D8: 18 0C 00 00 18 0C 00 00   $22C2
63E0: 18 0C 00 00 3C 1E 00 00   $B047
```

```
6420: 00 00 00 00 7C 0F 00 00   $ABAE
6428: 30 10 00 00 60 00 00 00   $B346
6430: 40 01 00 00 00 03 00 00   $3F03
6438: 40 01 00 00 60 00 00 00   $0ED1
6440: 30 10 00 00 7C 0F 00 00   $6DE0

6480: 00 00 00 00 30 06 00 00   $8D2F
6488: 48 09 00 00 48 09 00 00   $79CF
6490: 40 01 00 00 40 01 00 00   $FD5F
6498: 40 01 00 00 40 01 00 00   $39CF
64A0: 40 01 00 00 60 03 00 00   $E5DA

64E0: 00 00 00 00 60 03 00 00   $185D
64E8: 40 01 00 00 70 07 00 00   $2C22
64F0: 58 0D 00 00 48 09 00 00   $8070
64F8: 58 0D 00 00 70 07 00 00   $7879
6500: 40 01 00 00 60 03 00 00   $F40C

6540: 00 00 00 00 60 03 00 00   $098B
6548: 40 01 00 00 4C 19 00 00   $9357
6550: 48 09 00 00 50 05 00 00   $CB49
6558: 60 03 00 00 40 01 00 00   $7F68
6560: 40 01 00 00 60 03 00 00   $C3CD

65A0: 7E 3F 00 00 0E 38 00 00   $9614
65A8: 66 33 00 00 72 27 00 00   $B4AA
65B0: 72 27 00 00 72 27 00 00   $1C5E
65B8: 72 27 00 00 72 27 00 00   $34EA
65C0: 66 33 00 00 42 21 00 00   $56FB
65C8: 7E 3F 00 00 00 00 00 00   $E806

6600: 7E 3F 00 00 3E 3E 00 00   $DDF0
6608: 4E 39 00 00 66 33 00 00   $89E9
6610: 72 27 00 00 02 20 00 00   $74A0
6618: 72 27 00 00 66 33 00 00   $0E06
6620: 4E 39 00 00 3E 3E 00 00   $C303
6628: 7E 3F 00 00 00 00 00 00   $6DFE

6660: 7E 3F 00 00 3E 3E 00 00   $4808
6668: 3E 3E 00 00 5E 3D 00 00   $F13B
6670: 5E 3D 00 00 6E 3B 00 00   $19C4
```

```
6678: 6E 3B 00 00 76 37 00 00   $2586
6680: 76 37 00 00 62 23 00 00   $C7F0
6688: 7E 3F 00 00 00 00 00 00   $A9FD

66C0: 7E 3F 00 00 02 20 00 00   $B238
66C8: 66 33 00 00 66 33 00 00   $0234
66D0: 66 33 00 00 66 33 00 00   $B2F8
66D8: 66 33 00 00 66 33 00 00   $02F4
66E0: 66 33 00 00 42 21 00 00   $8085
66E8: 7E 3F 00 00 00 00 00 00   $DE18

6720: 7E 3F 00 00 02 30 00 00   $B565
6728: 4E 2F 00 00 1E 3F 00 00   $5DE9
6730: 3E 3E 00 00 7E 3C 00 00   $2188
6738: 3E 3E 00 00 1E 3F 00 00   $E03E
6740: 4E 2F 00 00 02 30 00 00   $732B
6748: 7E 3F 00 00 00 00 00 00   $DD56

6780: 7E 1F 00 00 4E 19 00 00   $3DAD
6788: 36 16 00 00 36 16 00 00   $59B9
6790: 3E 1E 00 00 3E 1E 00 00   $4D9D
6798: 3E 1E 00 00 3E 1E 00 00   $19F9
67A0: 3E 1E 00 00 1E 1C 00 00   $5558
67A8: 7E 1F 00 00 00 00 00 00   $9BC5

67E0: 7E 3F 00 00 1E 3C 00 00   $86C6
67E8: 3E 3E 00 00 0E 38 00 00   $42DD
67F0: 26 32 00 00 36 36 00 00   $1EAB
67F8: 26 32 00 00 0E 38 00 00   $16C6
6800: 3E 3E 00 00 1E 3C 00 00   $6A97
6808: 7E 3F 00 00 00 00 00 00   $14AA

6840: 7E 3F 00 00 1E 3C 00 00   $B919
6848: 3E 3E 00 00 32 26 00 00   $3371
6850: 36 36 00 00 2E 3A 00 00   $7B9B
6858: 1E 3C 00 00 3E 3E 00 00   $DF0E
6860: 3E 3E 00 00 1E 3C 00 00   $735F
6868: 7E 3F 00 00 00 00 00 00   $DD62

68A0: 00 00 00 00 60 03 00 00   $B0C5
68A8: 30 06 00 00 00 06 00 00   $3029
```

Black Box Source Code

```
                    .OR $300
                    .TF OBJ.BLACK BOX

* ------------------------------------------------------------------------ *
*                          CALL THE ANIMATOR                               *
* ------------------------------------------------------------------------ *

Ø3ØØ: A2 ØØ                 LDX #Ø        GET BACKGROUND NUMBER
Ø3Ø2: AØ ØØ                 LDY #Ø        GET SHAPE NUMBER
Ø3Ø4: A9 Ø1                 LDA #1        ALWAYS ENTER AS SPRITE DRAW
Ø3Ø6: 4C B8 83              JMP $83B8     CALL ANIMATOR

* ------------------------------------------------------------------------ *
*                             MAKE A NOTE                                  *
* ------------------------------------------------------------------------ *

Ø3Ø9: A2 ØØ                 LDX #Ø        DURATION OF NOTE
Ø3ØB: AØ ØØ          LOOP2  LDY #Ø        PITCH OF NOTE
Ø3ØD: AD 3Ø CØ              LDA $CØ3Ø     CLICK
Ø31Ø: 88             LOOP1  DEY
Ø311: DØ FD                 BNE LOOP1
Ø313: CA                    DEX
Ø314: DØ F5                 BNE LOOP2
Ø316: 6Ø                    RTS
```

```
68B0: 00 03 00 00 40 01 00 00   $C488
68B8: 40 01 00 00 00 00 00 00   $F52A
68C0: 40 01 00 00 40 01 00 00   $31AA

6900: 1D 0E 00 00 23 11 00 00   $57EB
6908: 43 10 00 00 23 11 00 00   $9EE5
6910: 15 0A 00 00 49 04 00 00   $B89E
6918: 15 0A 00 00 23 11 00 00   $1A1D
6920: 43 10 00 00 23 11 00 00   $4373
6928: 1D 0E 00 00 00 00 00 00   $5DE5

6960: 62 31 00 00 5C 2E 00 00   $B374
6968: 3C 2F 00 00 5C 2E 00 00   $220A
6970: 6A 35 00 00 36 3B 00 00   $DC01
6978: 6A 35 00 00 5C 2E 00 00   $26F2
6980: 3C 2F 00 00 5C 2E 00 00   $A7EC
6988: 62 31 00 00 00 00 00 00   $2796

69B8: 00 00 00 00 00 00 00 00   $2796
```

Black Box Object Code

```
0300: A2 00 A0 00 A9 01 4C B8   $9E64
0308: 83 A2 00 A0 00 AD 30 C0   $42B8
0310: 88 D0 FD CA D0 F5 60      $680D
```

Black Box BASIC Program

```
10 REM     /-----------/
20 REM    /           /!
30 REM   / BLACK BOX / !
40 REM  /           /  !
50 REM /-----------/   !
60 REM \    BY     \   !
70 REM  \           \  !
80 REM   \ RAY DARRAH\ !
90 REM    \           \!
100 REM    \-----------\
110 REM
120 TEXT : HOME : LOMEM: 16384 : CLEAR : IF PEEK
     (775 ) = 184 THEN 150
130 PRINT CHR$ (4 ) "BLOAD^ OBJ.BLACK^ BOX" :
     PRINT CHR$ (4 ) "BLOAD^ SH.BLACK^ BOX"
140 PRINT CHR$ (4 ) "BLOAD^ OBJ.ANIMATOR" : POKE
     33798 ,85
150 DIM B(11 ,11 ) ,PB(11 ,11 ) :PX = Ø :PY = 2 :A
     = 1 :K$ = "IJKM^ " + CHR$ (13 ) + CHR$ (27 )
     :M = 7 :W = 1
160 POKE 230 ,32 : CALL 62450 : HCOLOR= 3 : FOR
     A = 56 TO 223 STEP 14
170 HPLOT A ,18 TO A ,173 : HPLOT A + 13 ,18 TO A
     + 13 ,173 : NEXT
180 FOR A = 18 TO 173 STEP 13 : HPLOT 56 ,A TO 223
     ,A : HPLOT 56 ,A + 12 TO 223 ,A + 12 : NEXT
190 FOR A = 19 TO 29 : HPLOT 56 ,A TO 223 ,A : HPLOT
     56 ,A + 143 TO 223 ,A + 143 : NEXT
200 FOR A = 57 TO 68 : HPLOT A ,18 TO A ,173 : HPLOT
     A + 154 ,18 TO A + 154 ,173 : NEXT
210 POKE 778 ,1Ø : FOR A = 255 TO W STEP - W : POKE
     78Ø ,A : CALL 777 : NEXT
220 PRINT TAB( 16 ) "BLACK^ BOX" : PRINT : PRINT
     : PRINT TAB( 14 ) "BY^ RAY^ DARRAH"
230 VTAB 22 : PRINT TAB( 1Ø ) "PRESS^ ANY^ KEY^
     TO^ PLAY" : WAIT - 16384 ,128 : GET A$ : IF
     A$ = CHR$ (27 ) THEN 77Ø
240 HOME : VTAB 1Ø : PRINT "HOW^ MANY^ ATOMS^
     SHALL^ I^ HIDE^ FOR^ YOU?" : PRINT "(Ø^ -^
     5,^ Ø^ =^ RANDOM^ AMOUNT)"
250 PRINT : INPUT B :B = INT (B ) : IF B < Ø OR B >
     5 THEN 24Ø
260 NA = B : IF B = Ø THEN B = INT ( RND (W ) * 5
     ) + W :NA = 5
270 FOR A = W TO B
280 X = RND (W ) * 1Ø + W :Y = RND (W ) * 1Ø +
     W : IF B(X ,Y ) THEN 28Ø
290 B(X ,Y ) = W : NEXT
300 POKE - 16297 ,Ø : POKE - 163Ø2 ,Ø : POKE -
     163Ø4 ,Ø
310 POKE 771 , (PY = Ø ) + 2 * (PX = 13 ) + 3 *
     (PY = 13 ) : POKE 769 ,Ø : GOSUB 78Ø :B = PEEK
     (771 )
320 GET A$ : FOR A = W TO 7 : IF A$ < > MID$ (K$ ,A
     ,W ) THEN NEXT : GOTO 32Ø
330 POKE 771 ,B : ON A GOTO 34Ø ,36Ø ,38Ø ,4ØØ ,6ØØ
     ,7ØØ ,77Ø
340 GOSUB 78Ø : IF PY = 2 THEN PY = Ø :PX = 2 * (PX
     = Ø ) + 11 * (PX = 13 ) : GOTO 42Ø
350 PY = PY - (PY > Ø ) : GOTO 42Ø
360 GOSUB
370 PX = PX - (PX > Ø ) : GOTO 42Ø
380 GOSUB 78Ø : IF PX = 11 THEN PX = 13 :PY = 2 *
     (PY = Ø ) + 11 * (PY = 13 ) : GOTO 42Ø
390 PX = PX + (PX < 13 ) : GOTO 42Ø
400 GOSUB 78Ø : IF PY = 11 THEN PY = 13 :PX = 2 *
     (PX = Ø ) + 11 * (PX = 13 ) : GOTO 42Ø
410 PY = PY + (PY < 13 )
420 IF PY = 13 OR PY = Ø OR PX = 13 OR PX = Ø THEN 31Ø
430 PX = PX + (PX = W ) - (PX = 12 ) :PY = PY + (PY
     = W ) - (PY = 12 )
440 POKE 771 ,6 :B = W
450 GOSUB 8ØØ :B = W - B : FOR A = W TO 2Ø : IF PEEK
     ( - 16384 ) < 128 THEN NEXT : GOTO 45Ø
460 IF B = Ø THEN GOSUB 8ØØ
470 GET A$ : FOR A = W TO 7 : IF MID$ (K$ ,A ,W ) <
     > A$ THEN NEXT : GOTO 44Ø
480 ON A GOTO 49Ø ,5ØØ ,51Ø ,52Ø ,55Ø ,7ØØ ,77Ø
490 PY = PY - W - (PY = 2 ) : GOTO 53Ø
500 PX = PX - W - (PX = 2 ) : GOTO 53Ø
510 PX = PX + W + (PX = 11 ) : GOTO 53Ø
520 PY = PY + W + (PY = 11 )
530 IF PX = Ø OR PX = 13 OR PY = Ø OR PY = 13 THEN 31Ø
540 GOTO 44Ø
550 IF PB(PX - W ,PY - W ) THEN 59Ø
560 IF NA = AB THEN 44Ø
570 AB = AB + W :PB(PX - W ,PY - W ) = W
580 POKE 771 ,23 : GOSUB 8ØØ : GOTO 44Ø
590 PB(PX - W ,PY - W ) = Ø :AB = AB - W : GOTO 58Ø
600 XS = (PX = Ø ) - (PX = 13 ) :YS = (PY = Ø ) - (PY
     = 13 ) :X = PX - W - (PX = 13 )
610 X = PX - W - (PX = 13 ) :X = (X < > - W ) * X :Y
     = PY - W - (PY = 13 ) :Y = (Y < > - W ) * Y :
     IF PB(X ,Y ) THEN 32Ø
620 PB(X ,Y ) = W : POKE 778 ,7 : FOR A = 1Ø TO 8Ø
     : POKE 78Ø ,A : CALL 777 : NEXT :SX = X :SY = Y
630 IF X + XS < Ø OR X + XS > 11 OR Y + YS < Ø OR Y
     + YS > 11 THEN POKE 771 ,M :M = M + W : GOSUB
     81Ø :PB(X ,Y ) = W :X = SX :Y = SY : GOSUB 81Ø
     : GOTO 32Ø
640 IF B(X + XS ,Y + YS ) THEN X = SX :Y = SY : POKE
     771 ,4 : GOSUB 81Ø : GOTO 32Ø
650 IF B(X + (YS < > Ø ) + XS ,Y + (XS < > Ø ) + YS
     ) AND B(X - (YS < > Ø ) + XS ,Y - (XS < > Ø ) +
     YS ) THEN X = SX :Y = SY : POKE 771 ,5 : GOSUB
     81Ø : GOTO 32Ø
660 IF B(X + (YS < > Ø ) + XS ,Y + (XS < > Ø ) + YS
     ) THEN XS = - ( ABS (XS ) < > W ) :YS = - ( ABS
     (YS ) < > W ) : GOTO 68Ø
670 IF B(X - (YS < > Ø ) + XS ,Y - (XS < > Ø ) + YS
     ) THEN XS = ( ABS (XS ) < > W ) :YS = ( ABS (YS
     ) < > W )
680 IF ((X = Ø OR X = 11 ) AND YS < > Ø ) OR ((Y =
     Ø OR Y = 11 ) AND XS < > Ø ) THEN POKE 771 ,5
     :X = SX :Y = SY : GOSUB 81Ø : GOTO 32Ø
690 X = X + XS :Y = Y + YS : GOTO 63Ø
700 IF PEEK (771 ) < 4 THEN GOSUB 78Ø
710 FOR Y = W TO 1Ø : FOR X = W TO 1Ø : IF PB(X ,Y
     ) = Ø AND B(X ,Y ) = Ø THEN 74Ø
720 IF PB(X ,Y ) AND B(X ,Y ) THEN POKE 771 ,23 :
     GOSUB 79Ø : POKE 771 ,24 : GOSUB 79Ø :S = S
     + 1ØØ / NA : GOTO 74Ø
730 IF B(X ,Y ) THEN POKE 771 ,25 :S = S - 1ØØ / NA
     : GOSUB 79Ø
740 NEXT : NEXT : POKE 778 ,128 : POKE 78Ø ,Ø :
     CALL 777 : GET A$
750 HOME : TEXT : VTAB 11 : PRINT "YOUR^ SCORE^
     IS^ " INT (S - 1Ø * ((M - 8 ) / NA ) ) "↑"
760 PRINT : PRINT "PRESS^ A^ KEY" : WAIT - 16384
     ,128 : GET A$ : RUN
770 TEXT : HOME : END
780 POKE 224 ,PX * 7 + 2Ø + 2 * (PX > Ø ) : POKE
     225 ,PY * 13 + 8 * (PY > Ø ) : CALL 768
     : RETURN
790 PX = X + W :PY = Y + W
800 POKE 224 , (PX - 2 ) * 7 + 35 : POKE 225 , (PY
     - 2 ) * 13 + 32 : CALL 768 : RETURN
810 POKE 224 ,28 + X * 7 : POKE 225 ,19 + Y *
     13 : CALL 768 : RETURN
```

Black Box checksums

```
10    - $BADD     420   - $2DØ7
20    - $9B13     430   - $E2D5
30    - $4D3B     440   - $Ø677
40    - $AD92     450   - $98B7
50    - $C899     460   - $8919
60    - $FF65     470   - $DFD4
70    - $A3BF     480   - $A429
80    - $A9ØØ     490   - $9FAB
90    - $924D     500   - $8F9C
100   - $CB63     510   - $ØE22
110   - $BD13     520   - $F8D5
120   - $3B38     530   - $613D
130   - $A13E     540   - $6CC1
140   - $94A2     550   - $Ø9B8
150   - $9EØ3     560   - $6778
160   - $E58E     570   - $Ø378
170   - $2FA8     580   - $BC4F
180   - $A5B2     590   - $EBØ3
190   - $C239     600   - $38B4
200   - $8AFC     610   - $DD8C
210   - $383D     620   - $6616
220   - $AØ57     630   - $849E
230   - $87CA     640   - $2A3Ø
240   - $7Ø66     650   - $41E2
250   - $6DØ9     660   - $54ØF
260   - $Ø854     670   - $7672
270   - $ECAE     680   - $C3A8
280   - $D63F     690   - $269F
290   - $CA96     700   - $C1E8
300   - $247C     710   - $3751
310   - $CØØE     720   - $5E85
320   - $E344     730   - $C464
330   - $5DØ3     740   - $211F
340   - $ØDC6     750   - $36CF
350   - $6797     760   - $1D1Ø
360   - $8ØDØ     770   - $732Ø
370   - $A5C5     780   - $11AB
380   - $7F63     790   - $8Ø72
390   - $F5A6     800   - $9D27
400   - $3692     810   - $256F
410   - $Ø71B
```

softkey for...

LIFESAVER

Life

by Tim Beckmann

MicroLab, Inc.
2699 Skokie Valley Road
Highland Park, IL 69935

Requirements:
64K
A modified ROM (see below)
A blank disk
Lifesaver disk

***Editor's Note:** this procedure requires the use of a modified F8 ROM similar to the one presented in COMPUTIST No. 19 to capture some sensitive memory used by the program (pages $ØØ-$Ø8) and save it in a binary file. If you have a 16K RAM card that can be moved (in an Apple][Plus) you can simulate the ROM with the method given in this article.*

Lifesaver is a utility that is very useful for recoving data from damaged disks. Anyone who has used Lifesaver to recover data knows how valuable it can be. But, ironically, this piece of software that is meant to recover damaged disks is protected so well that it cannot be backed up in case the original is damaged. I attempted to copy the disk with Copy][Plus and EDD, but both failed to do the job. It appears that Microlab used a foreign DOS with some other very difficult protection that is hard to bypass. This means that Super IOB cannot be used to unprotect the disk. Luckily, Lifesaver is a single load program, so we will be able to capture all of it from memory. One minor drawback, however, is that it will require 64K to run because we need DOS in memory to load the files.

The first thing you should do is to INITialize a disk with a blank HELLO program. It will greatly decrease the loading time if you use a fast DOS.

1) Initialize a blank disk with a normal or fast DOS.

```
NEW
INIT HELLO
```

Those of you with a modified ROM or a card that can move memory may skip to step 4. If you are the owner of a][Plus without a modified ROM, The first step to take toward cracking this program is to move your 16K card to slot 1. This is done so we can install an image of the modified F8 ROM in the RAM card and not have it erased by the program. Unfortunately, those of you who have a //e or //c will need a modified ROM installed because you cannot move your RAM card.

2) Turn off your computer and move your RAM card to slot 1.

3) Next you need to install the image of the modified F8 ROM in the RAM card.

```
BLOAD F8 ROM,A$28ØØ
CALL-151
CØ91 N CØ91 N F8ØØ<28ØØ.2FFFM
DØØØ<DØØØ.F7FFM N CØ9Ø
```

Now with the modified F8 ROM working the next step is to boot the Lifesaver disk.

4) Insert the Lifesaver disk and boot it with C6ØØG from the monitor or PR#6 from BASIC.

5) After the drive stops, Lifesaver should be up and running. Now press Reset followed by the "S" key to move pages $ØØ through $Ø8 to pages $2Ø through $28. Insert your blank initialized disk. Then move some more memory to insure the data remains safe and boot the disk.

```
3ØØØ<89ØØ.BFFFM
C6ØØG
```

6) Save the program in three files as follows:

```
BSAVE LIFESAVER.2,A$2ØØØ,L$8FF
BSAVE
LIFESAVER.3,A$4ØØØ,L$26FF
BSAVE LIFESAVER.4,A$3ØØØ,L$FFF
```

Saver

LIFESAVER

7) Once again boot the Lifesaver disk and press Reset followed by "M" to enter the monitor. Then again reboot the normal disk to save a file.

```
C6ØØG
BSAVE LIFESAVER.1,A$9ØØ,L$7FFF
```

8) If you used the RAM card method turn off your computer and move your card back to slot Ø. The data capturing part of the softkey is over. It is now time to develop a program to reassemble it in memory and start it running. But first it is necessary to clarify a couple of points.

After some investigating and with a lot of luck, I found that the program's entry point is at $52ØA and, as a form of protection, uses the second text page for its display. So with these points in mind, I developed two binary programs to put the program into memory and start it up. The first binary program will load part of the files into main memory and the other part into the RAM card. It will then put pages $ØØ through $Ø8 back where they belong and jump to the second binary program that will be loaded at $F7ØØ. That program will finish moving memory to the correct pages, turn on the second text page with a LDA $CØ55, then jump to the start of the program at $52ØA.

Type in hexdump #1 and save it. Then load in the file LIFESAVER.3.

```
BSAVE LIFESAVER,A$92ØØ,L$EØ
BLOAD LIFESAVER.3,A$1ØØØ
```

Type in hexdump #2 and save it plus the old LIFESAVER.3 in a new LIFESAVER.3.

```
BSAVE
LIFESAVER.3,A$1ØØØ,L$273F
```

9) Time to make a HELLO program to boot Lifesaver automatically, so type in this one-liner and save it as HELLO.

```
1Ø PRINT CHR$(4);"BRUN LIFESAVER"
SAVE HELLO
```

You should now have a backup of Lifesaver, so in addition to being able to recover damaged disks, you don't have to worry about your original being damaged.

Hexdump #1

```
92ØØ: 2Ø BB 92 84 C2 CC CF C1   $615C
92Ø8: C4 CC C9 C6 C5 D3 C1 D6   $E868
921Ø: C5 D2 AE B1 AC C1 A4 B9   $2Ø54
9218: BØ BØ AØ 8D ØØ 2Ø BB 92   $C8F8
922Ø: 84 C2 CC CF C1 C4 CC C9   $5881
9228: C6 C5 D3 C1 D6 C5 D2 AE   $D58B
923Ø: B2 AC C1 A4 B8 B9 BØ BØ   $EB84
9238: AØ 8D ØØ AD 81 CØ AD 81   $26AE
924Ø: CØ AØ ØØ B9 ØØ F8 99 ØØ   $156F
9248: F8 C8 DØ F7 EE 45 92 EE   $1FF1

925Ø: 48 92 AD 45 92 DØ EC 2Ø   $FD3F
9258: BB 92 84 C2 CC CF C1 C4   $A59C
926Ø: CC C9 C6 C5 D3 C1 D6 C5   $32A5
9268: D2 AE B3 AC C1 A4 C4 BØ   $CEB8
927Ø: BØ BØ AØ 8D ØØ AD 89 CØ   $5C93
9278: AD 89 CØ 2Ø BB 92 84 C2   $ØAE1
928Ø: CC CF C1 C4 CC C9 C6 C5   $AA39
9288: D3 C1 D6 C5 D2 AE B4 AC   $93AF
929Ø: C1 A4 C4 BØ BØ BØ AØ AØ   $B988
9298: 8D ØØ AØ ØØ B9 ØØ 89 99   $3459

92AØ: ØØ ØØ C8 DØ F7 EE 9E 92   $7628
92A8: EE A1 92 AD A1 92 C9 Ø9   $4FAB
92BØ: DØ EA AD 8B CØ AD 8B CØ   $FAA9
92B8: 4C ØØ F7 68 85 ØØ 68 85   $5E61
92CØ: Ø1 E6 ØØ A5 ØØ DØ Ø2 E6   $EC68
92C8: Ø1 AØ ØØ B1 ØØ FØ Ø6 2Ø   $E6D7
92DØ: ED FD 4C C1 92 E6 ØØ A5   $12CB
92D8: ØØ DØ Ø2 E6 Ø1 6C ØØ ØØ   $5D15
```

Hexdump #2

```
37ØØ: AØ ØØ B9 ØØ DØ 99 ØØ 89   $CADØ
37Ø8: C8 DØ F7 EE Ø4 F7 EE Ø7   $B2Ø8
371Ø: F7 AD Ø7 F7 C9 99 DØ EA   $616F
3718: AØ ØØ AD 83 CØ AD 83 CØ   $1Ø14
372Ø: B9 ØØ DØ 99 ØØ 99 C8 DØ   $A8BF
3728: F7 EE 22 F7 EE 25 F7 AD   $17F5
373Ø: 25 F7 C9 CØ DØ EA AD 8Ø   $6EAC
3738: CØ AD 55 CØ 4C ØA 52      $EA78
```

softkey for...

Synergistic Software

by Danny Pollak

Synergistic Software
83Ø N. Riverside Dr., Ste. 2Ø1
Renton, WA 98Ø55

Requirements:
Apple][Plus or better
Microbe or Crisis Mountain
Super IOB 1.5
One blank disk

Microbe is an action packed adventure game in which a submarine and its entire crew are reduced to microscopic size and are then injected into the bloodstream of a critically ill patient. Your ultimate mission is to work your way through the body to the brain and repair the damage located there. There is a time limit of 1 hour 1Ø minutes in which to accomplish this feat. When a higher level of play is selected, this game can definitely be a challenge to the best of adventurers.

Crisis Mountain is an arcade style game in which you must move around within the mountain diffusing bombs before they explode while trying to avoid flying, rolling, and falling boulders. There are several different types of boulders which must be avoided in order to stay alive, and at higher levels of play, you must also avoid a radioactive bat which can kill you with just one touch.

The Protection

Both of these games are protected in the same way. The folks at Synergistic Software decided to use a modified RWTS to read in the different sections of the game. During the boot, the program also performs a nibble count searching for the sequence of bytes EB D5 AA. With a slightly modified swap controller and a few sector edits, the copy protection on these disks easily fall to Super IOB. This procedure may possibly work with other software from Synergistic.

The Procedure

1) The first thing we must do is to capture the RWTS from one of the disks. Do this by entering the following commands:

```
CALL -151
96ØØ<C6ØØ.C7ØØM
96F8:A9 ØØ 8D 4A Ø8 4C Ø1 Ø8
96ØØG
19ØØ<B8ØØ.BFFFM
```

2) Boot an initialized (slave) disk and then save the RWTS with the following command:

```
BSAVE
SYNERGISTIC.RWTS,A$19ØØ,L$8ØØ
```

3) Install the controller at the end of the article into Super IOB 1.5. Follow the prompts being sure to answer YES when asked if you wish to initialize the duplicate disk.

4) When Super IOB is finished, you will have a deprotected copy of the program. The following sector edits are performed by the program:

Track	Sector	Byte	From	To
ØØ	Ø2	9E	FF	DE
ØØ	Ø2	A3	FF	AA
ØØ	Ø2	A8	FF	EB
ØØ	Ø3	35	FF	DE
ØØ	Ø3	3F	FF	AA
ØØ	Ø3	91	FF	DE
ØØ	Ø3	9B	FF	AA
ØØ	Ø4	29	AA	96
ØØ	Ø4	AA	ØØ	AA
ØØ	Ø5	26	F7	ØØ

If you are copying Microbe, the back side of the disk can be copied with either COPYA or the standard controller.

For some unexplainable reason, Microbe locks up if any information has previously been loaded into memory. The easiest way to solve this problem is to turn the machine off and then do a cold boot before starting the game.

A fix for Crisis Mountain

How many times have you wished that the title page sequence for Crisis Mountain didn't take so long to execute. Well, if you are like me, you would rather get on with the game than watch a little man jump back and forth over the name of the guy who created the game. Here's how to do it.

Break out your favorite sector editor and perform the following sector edits on the now unprotected disk:

Track	Sector	Byte	From	To
ØØ	ØB	AC	ØØ	18
ØØ	ØB	D3	ØC	3A
Ø7	ØC	EC	2Ø	4C
Ø7	ØC	ED	53	99
Ø7	ØC	EE	49	42
Ø7	ØD	4C	BC	B4
Ø7	ØD	4D	4C	41

The title page will now appear in record time and you can get on with the game.

controller

```
1ØØØ REM SYNERGISTIC SOFTWARE
1Ø1Ø TK=Ø :LT=35 :ST=15 :LS=15 :CD=WR :FAST
     =1
1Ø2Ø POKE 775 ,96
1Ø3Ø GOSUB 36Ø : GOSUB 49Ø : GOSUB 61Ø
1Ø4Ø T1=TK :TK=PEEK (TRK) - 1 : RESTORE : GOSUB
     31Ø :TK=T1
1Ø5Ø GOSUB 36Ø : GOSUB 49Ø : GOSUB 61Ø : IF PEEK
     (TRK) = LT THEN 1Ø7Ø
1Ø6Ø TK=PEEK (TRK) :ST=PEEK (SCT) : GOTO 1Ø2Ø
1Ø7Ø TK = Ø :LT = 1 :ST = 1 : POKE 47426 ,24
1Ø8Ø GOSUB 49Ø : GOSUB 61Ø
1Ø9Ø GOSUB 49Ø : GOSUB 61Ø
11ØØ HOME : PRINT "COPYDONE" : END
111Ø DATA 1Ø^ CHANGES
112Ø DATA Ø ,2 ,158 ,222
113Ø DATA Ø ,2 ,163 ,17Ø
114Ø DATA Ø ,2 ,168 ,235
115Ø DATA Ø ,3 ,53 ,222
116Ø DATA Ø ,3 ,63 ,17Ø
117Ø DATA Ø ,3 ,145 ,222
118Ø DATA Ø ,3 ,155 ,17Ø
119Ø DATA Ø ,4 ,41 ,15Ø
12ØØ DATA Ø ,4 ,17Ø ,17Ø
121Ø DATA Ø ,5 ,38 ,Ø
1ØØ1Ø PRINT CHR$ (4) "BLOAD^SYNERGISTIC.RWTS"
```

controller checksums

1ØØØ	-	$356B	112Ø	-	$ØAB3
1Ø1Ø	-	$2544	113Ø	-	$8927
1Ø2Ø	-	$6AF1	114Ø	-	$9ØB3
1Ø3Ø	-	$FBBD	115Ø	-	$DD6F
1Ø4Ø	-	$F863	116Ø	-	$6955
1Ø5Ø	-	$C1F5	117Ø	-	$3BF9
1Ø6Ø	-	$5Ø1D	118Ø	-	$1359
1Ø7Ø	-	$7AAF	119Ø	-	$3ØBC
1Ø8Ø	-	$ØA55	12ØØ	-	$B341
1Ø9Ø	-	$DØ48	121Ø	-	$EBED
11ØØ	-	$65FD	1ØØ1Ø	-	$38D3
111Ø	-	$B5CA			

Softkey for...

by Frank Carone

Baudville Software
1001 Medical Park Dr. SE
Grand Rapids, MI 49506
$38.00

Requirements:
64K Apple][Plus, //e, //c
One disk drive
A sector editor
COPYA
One blank disk
A disk searcher (optional)

Blazing Paddles is a great graphics drawing program that is simple to use, with a smooth flow between the many built-in features. There are shapes and fonts on the disk, plus you can purchase other disks for a greater variety. This program works with many different types of input devices (mouse, light pen, graphic tablet, etc.). It also interfaces with a fair amount of older and newer models of printers. This is one of the things I particularly like about it. I have Print Shop, a great program, but it will not work with my IDS 46Ø printer.

When I recieved Blazing Paddles, the first thing I tried to do was back it up. First I tried to COPYA it and all seemed to go well until I tried to boot it. The program placed "BLAZING PADDLES" and the serial number at the top of the screen, then the disk sounded as though it was looking for something to do but could not figure out what. Next I went through all my back issues of COMPUTIST and every parameter list for all the copy programs I have, which are many, and found one for EDD III. I tried EDD without any success. Then I warmed up my Snapshot card. I captured the program up to the first menu, which is past the protection on the disk, took out this disk and inserted my COPYAed disk, and had full use of the program. Swapping disks is a hassle, however, and would not do.

Back in COMPUTIST No. 25, I read a softkey by Clay Harrell for Take 1, also by Baudville. I said "Great! Here is my softkey. My program is doing the same thing. All I have to do is make the changes at the same place and I am home free!!" WRONG. It does not work that easily.

What I did instead was boot my COPYAed disk (this is the disk we will work with from here on). When it placed "BLAZING PADDLES" and the serial number on the screen, and the disk drive went into never-never land, I froze it in memory with the Snapshot card. I dropped into the monitor, which showed me that the program was running at $B944 in memory. Notice that this is the routine that checks the D5 AA 96 (prologue) and DE AA (epilogue) address markers, which are of course standard on every DOS 3.3 sector. These tell DOS where data begins and ends. Like Take 1, all seems normal with this routine, so we will have to backtrack and find the calling routine.

This can be accomplished by any memory search utility. I used the Inspector and found the JuMP to $B944 at $BBØ7. Type BBØØL from the monitor so you can look at it.

Now we're HOT! Notice at $BBØA the LDA $2D (load accumulator with what is at $2D) and then CMP #$ØØ (compare with the number $ØØ) followed by a BNE (branch if not equal) to $BBØ7. Here is our infinite loop. If the subroutine at $B944 returns with a non-zero value in $2D, the BNE $BBØ7 will occur. To defeat this routine, all we would have to do is find it on the disk and replace the bytes $DØ F7 (BNE $BBØ7) with $EA EA (NOPs) at $BBØE.

Warm up your disk searcher and search for the bytes $2Ø 44 B9 A5 2D C9 ØØ DØ F7. I found this at track Ø, sector $B, byte $Ø7. Replace the DØ F7 at byte $ØE with $EA EA. That disables the branch back to $BBØ7. Of course, like Take 1, the copy did not work yet.

Looking a bit further at the routine at $BBØØ, I noticed another JSR $B944 at $BB1D, followed by another JSR, a CoMPare, and a BNE back to $BBØ2. Another loop? Yes, apparently this is a second check of the disk before it will load in the rest of the program. These bytes live in the same sector as the last loop. Just replace the $DØ D5 (BNE $BBØ2) with $EA $EA and you have a COPYAable, deprotected Blazing Paddles.

To do the above, I used my Apple][Plus with all the goodies such as a Snapshot card and a Trackstar (a device that monitors the disk drive arm to show what track it is on). I also have a plain Apple //c that has given me trouble in the past with softkeys because of the differences between the disk drive controller ROMs and other hardware problems.

To get in and examine this program on a //c (or another Apple][), you can just start by booting the COPYAed copy of Blazing Paddles. When it hangs up, open the drive door and press Reset. The computer will attempt to reboot. Press Reset again and this should get you into BASIC. Enter the monitor with CALL -151 and type BBØØL. Here you are at the protection routine, and you can continue the above softkey from there.

Step by Step

1) Copy Blazing Paddles with COPYA or a similar program.

2) Use your sector editor to make the following changes to the copy.

Track	Sector	Byte	From	To
Ø	$B	$ØE	$DØ	$EA
		$ØF	$F7	$EA
		$2B	$DØ	$EA
		$2C	$D5	$EA

3) Write the sector back to the disk and you're done!

In addition, the shapes & fonts library disk can be copied straight across with COPYA.

Deprotecting
Deprotecting
Deprotecting
Deprotecting
Deprotecting
Deprotecting
Deprotecting
Deprotecting
Deprotecting

Deprotecting

by William Hinger

Computer Solutions

Requirements:
Zardax (with write-protect tab)
COPYA
2 blank disks

There are any number of reasons for unprotecting disks. Zardax emphasizes three of these reasons quite well. First, the program is very expensive. Second, if one of the two program disks is damaged, there is a $30.00 replacement fee. And third, you have to reconfigure the program for each combination of printer/display that you wish to use. Since I use two separate computers with very different video characteristics and three different printers, this would mean either not using some of the equipment (unsatisfactory), or reconfiguring each time I changed equipment (also unsatisfactory). After removing the protection, I hoped to be able to configure one disk for each combination and pick the disk I wished to use based on the equipment combination.

My first impression was that removing the protection should be relatively easy. After all, I thought, you can boot a normal DOS 3.3 disk and then just RUN HELLO to start the program. Well, first impressions can be misleading. I am now convinced that the BASIC/machine code combination can be the most difficult code there is to trace. Opinions aside, let's get on with the break.

About all that HELLO (or HELLOX, which is exactly the same as HELLO) does is BRUN KBDMACRO and RUN HELLO1. Since the first instruction in KBDMACRO is an RTS, BRUNning this file does nothing more than load it. HELLO1, on the other hand, is where the bulk of the problems are. If you just load HELLO1, the computer goes off into never-never land and you have to press reset to regain control. The first line of the program has a line number of 65535 ($FFFF) and the pointer to the next line points at itself so that if you try to list the program, it just keeps listing line 65535 forever (or until you interrupt it). If you are trying to repair this program in memory, there are countless traps to overcome. The best way to examine this program is to use a sector editor and track the program via its track/sector list.

With all the BLOADs and CALLs that this program does, there are basically only two things that it accomplishes (other than keeping you out). The first is the check to see if you have an autoboot ROM in the same bank as Applesoft. This means that if you have floating-point BASIC in ROM, you must have an autoboot ROM, and if you have Integer in ROM, the language card must contain Applesoft with an image of the autoboot ROM. Since we will eliminate this portion of the load in our softkey, this check is also eliminated and the program will then run with the old ROM.

The second item accomplished is a sector read. Sector $F from track $3 is read into page $B4 of memory. This sector, and this sector only, uses a modified data trailer (DE AD instead of DE AA). If you wish to examine this sector with a sector editor, you must first change the byte at $B93F from AA to AD to read the sector. After reading this sector, the

Zardax

Zardax

Zardax

Zardax

Zardax

Zardax

Zardax

Zardax

Zardax

Zardax

Zardax

program executes a JMP to $B401.

This routine points the reset vector at a memory destruction routine; puts some values in zero page from $EC to $EF; puts a couple of values near the bottom of the stack at $108 and $109; and puts an IOB pointer at $E8 and $E9. It then restores the data trailer to DE AA, sets parameters for the next read, and modifies the expected values for the data mark to D5 AB AD. The normal middle value of the data mark is AA and is found at $B8F1. Sectors $E, $D and $C from track $3 are then read into memory pages $B1, $B2 and $B3 respectively, and the new data is executed via a JMP $B100.

At $B100, the data mark is first normalized to D5 AA AD. Then $18 (24) sectors are read into memory. The order of the read is track $3, sector $7 down to sector $0, then track $4, sector $F down to sector $0. This data, which is mainly composed of your printer and video routines as found in the file LOWER, occupies memory from $0800 to $1FFF. These are also the sectors which are written to when you run SETUP to change parameters for your printer and/or video display.

Now the data from page $B3 is moved to memory page $20. This is executed by pushing a return address onto the stack so that the next RTS encountered causes a return to this routine. Specifically, $2000 is pushed onto the stack. When the move routine (at $FE2C) is JMPed to, the RTS at the end of the routine pops the top address ($2000) off the stack, adds one to it (now $2001) and starts executing code at that address. Incidentally, if you wish to try this yourself, here's how you do it. Take the address of the code you wish the RTS to go to, and subtract one from this address. Then push this address onto the stack most significant byte first (with the PHA instruction). The next RTS will "return" to this code.

This routine ($2001) now reads in the main ZARDAX program. This code is stored from track $5 to track $10 in a 4+4 format, which means that it takes two "disk bytes", each contributing four bits, to make up one "memory byte". This type of code loads very fast since you don't need an intermediate buffer, but is very wasteful of disk space. The code is read into memory from $2100 to $9FFF.

Execution then returns to $B134. What happens now is that the code determines whether or not you have a RAM card. If you have a RAM card, the code copies an image of the monitor ROM into the language card, moves the code segment $5C00.8600 into the RAM card starting at $D000, moves $8600.9700 to memory starting at $9FD0, pushes $F00D onto the stack as a return address, and the JMPs to common code at $B248.

If you don't have a RAM card, the code segment $2100.5C00 is relocated to memory starting at $7600, $950D is pushed onto the stack, and execution falls through to the common code at $B248.

The common code first reads sectors $E, $D, $C and $B from track $13 into memory starting at $2000. These sectors contain the data from the disk file USER, which contains data only if you use a special printer driver with your setup. Your USER file must occupy this location on your disk in order to work properly. If you follow the procedure in this softkey, all will be well. If you choose to modify it further and do not use a special printer driver, you may wish to eliminate this sequence from the boot code we will enter later.

After reading in this data, the pointers to the start of the program buffer space are adjusted so that they point just after the routines in the

USER file. Data from locations $EC to $EF are copied to $FC to $FF and the monitor move routine is used to fill $B1ØØ to $B5ØØ with $6Øs. Although this overwrites all of the load code, the computer is executing inside the move routine in the monitor while this happens, and the RTS at the end of the routine causes execution to "return" to the address we earlier pushed onto the stack, which starts actual program execution.

Here's How

What are the goals of this softkey? First, to convert the disk to a completely standard format so that it can be copied with any copy program (such as COPYA). Second, the copy should still determine RAM size and execute the corresponding code segment. Third; the code should still load and utilize the USER file. And last, the SETUP program should still work with the unprotected disk. So let's go.

1) Before we start, we are going to set up a work disk we'll call WORK to be able to load DOS without having a program overwrite any data we might have in memory, so insert your DOS 3.3 System Master and load DOS.

```
PR#6
```

2) Now, insert WORK and INIT it with an empty HELLO program

```
NEW
INIT HELLO
```

3) Tracks 17 through 34 of the ZARDAX original are in standard format, and contain files and data that our copy will need. What we will need to do later is copy these tracks from the original to COPY with a copy program that will allow us to copy selected tracks. We'll do this by modifying COPYA to allow us to copy selected tracks. So first we need to insert the DOS 3.3 System Master and load the copy program.

```
LOAD COPYA
BLOAD COPY.OBJØ
```

4) The machine code in COPY.OBJØ does not need to be modified, but it does need to be on the same disk as our modified copy program, so insert WORK and save the binary file.

```
BSAVE COPY.OBJØ, A$2CØ, L$1ØB
```

Now we need to add some lines to COPYA so that it will suit our purpose and then save it to disk.

```
171 INPUT "STARTING^ TRACK:^ " ;ST : INPUT
     "ENDING^TRACK:^" ;LT : IF (ST > 34 ) OR (LT
     > 34 ) OR (ST > LT ) THEN 17Ø
172 POKE 77Ø ,LT + 1 : POKE 863 ,LT + 1 : IF ST >
     Ø THEN POKE 721 ,ST - 1 : POKE 722 ,ST - 1
173 PRINT : PRINT : INPUT "FORMAT^THIS^DISK?^
     " ;I$ : IF ( LEN (I$ ) > Ø ) AND ( LEFT$ (I$
     ,1 ) = "Y" THEN 175
174 FT = 1
```

```
SAVE COPYA.MOD
```

5) We are going to enter the code for the boot sequence now and save it as a binary file. This way, we don't have to worry about the DOS environment and by saving the code as a separate file we can easily repair it if we make mistakes while entering it. This code is the first place you need to look if your copy does not work correctly after finishing the softkey.

When we are ready, we will position the code on the disk so that it is read in and executed as the second stage of the boot process. Then, instead of reading in the portion of DOS that normally resides from $9D00 to $B5FF, it will read in ZARDAX, reposition it and execute it. This means that the first boot stage will read in ten pages instead of nine, so that there is room for our routine, which will eventually execute at $B6ØØ. So, enter and save this routine at this time.

```
CALL -151

16ØØ: 8E E9 B7 8E F7 B7 8E 32    $5AF7
16Ø8: B7 2Ø 89 FE 2Ø 93 FE 2Ø    $D7E9
161Ø: 58 FC A9 B2 8D Ø8 Ø1 A9    $1243
1618: CF 8D Ø9 Ø1 A9 E8 85 E8    $ØB83
162Ø: A9 B7 85 E9 A2 Ø3 BD 2F    $ØØB3
1628: B7 95 EC 95 FC CA 1Ø F6    $BD56
163Ø: 2Ø 93 B7 AD 83 CØ AD 83    $4E29
1638: CØ A9 A5 8D ØØ DØ CD ØØ    $AAØD
164Ø: DØ DØ 3E 4A 8D ØØ DØ CD    $14C9
1648: ØØ DØ DØ 35 AD 81 CØ AD    $739A

165Ø: 81 CØ AØ Ø7 2Ø FF B6 AD    $577D
1658: 83 CØ AD 83 CØ AØ ØF 2Ø    $8CF7
166Ø: FF B6 AØ 17 2Ø FF B6 AD    $8763
1668: Ø1 EB 85 AB AD Ø2 EB 85    $7194
167Ø: AC A9 ØØ 8D Ø1 EB 8D Ø2    $647E
1678: EB A9 9D 8D 1E Ø8 4C AA    $32F3
168Ø: B6 AD 81 CØ AØ 1F 2Ø FF    $Ø88C
1688: B6 A9 ØØ 8D 27 Ø8 A9 Ø2    $E8ED
169Ø: 8D 28 Ø8 A9 4C 8D 26 Ø8    $ØBF2
1698: AD Ø1 91 85 AB AD Ø2 91    $5FAD

16AØ: 85 AC A9 ØØ 8D Ø1 91 8D    $6F87
16A8: Ø2 91 A5 AB DØ Ø2 C6 AC    $13AØ
16BØ: C6 AB A5 AC 48 A5 AB 48    $37EF
16B8: A9 Ø4 8D F2 B7 A9 13 8D    $4D76
16CØ: EC B7 A9 ØE 8D ED B7 A9    $5D33
16C8: 2Ø 8D F1 B7 A9 EE 8D AC    $1178
16DØ: B7 2Ø 93 B7 AD 1B Ø8 85    $Ø855
16D8: 5Ø 85 5A AD 1C Ø8 85 51    $DADØ
16EØ: 85 5B AØ Ø2 B1 5Ø 18 65    $92CC
16E8: 5A 69 Ø4 85 5A 8D 1B Ø8    $E2D7

16FØ: C8 B1 5Ø 65 5B 85 5B 8D    $6C27
16F8: 1C Ø8 A9 A5 85 1F 6Ø A2    $C8C9
17ØØ: Ø8 B9 ØF B7 95 3B 88 CA    $32ØØ
17Ø8: DØ F7 AØ ØØ 4C 2C FE ØØ    $16Ø9
171Ø: F8 FF FF ØØ ØØ ØØ F8 ØØ    $FA47
1718: 5C ØØ 86 ØØ ØØ ØØ DØ ØØ    $6FC1
172Ø: 86 ØØ 97 ØØ ØØ DØ 9F ØØ    $4425
1728: 21 ØØ 5C ØØ ØØ ØØ 76 Ø1    $8C8E
173Ø: ØØ ØF                      $2F11
3DØG
BSAVE BOOTCODE, A$16ØØ, L$132
```

Incidentally, you could enter the above code as a text file using ZARDAX. This would allow you to use the editing features of the word processor. Then, when you are sure the code is correct, you could reboot a DOS disk (such as WORK), enter the monitor with a CALL -151, and EXEC the code into memory. You would then resave the code as a binary file using the same parameters as above.

6) Now we need to actually start unprotecting the ZARDAX disk. Insure that WORK is in the drive and run our modified copy program.

```
RUN COPYA.MOD
```

7) Once the drive has stopped spinning and the screen is prompting you for the slot of the original, remove WORK from the drive and use the copy program to copy tracks 17 - 34 from the ZARDAX original to our disk COPY. You will need to answer 17 to "STARTING TRACK: ", 34 to "ENDING TRACK: " and Y to "FORMAT THIS DISK? ". Follow the program instructions and make the copy.

8) Now insert the ZARDAX original into drive one and read the first load sector (track $3, sector $F) into memory at $B4ØØ.

```
CALL -151
3ØØ:2Ø E3 Ø3 2Ø D9 Ø3 4C 59 FF
B93F:AD
B7EA:Ø1 ØØ Ø3 ØF
B7FØ:ØØ B4 ØØ ØØ Ø1 ØØ
3ØØG
```

9) Use this code to read in the rest of the protected portion of ZARDAX.

```
B485:ØØ Ø3
3ØØ:48 8A 48 98 48 18 A9 59
3Ø8:8D 35 B1 A9 FF 8D 36 B1
31Ø:68 A8 68 AA 68 4C ØØ B1
B417G
```

10) The drive will still be spinning, so turn it off now.

```
CØE8
```

11) Insert COPY and write this code to our copy.

```
B7EA:Ø1 ØØ Ø9 ØF
B7FØ:ØØ 97 ØØ ØØ Ø2 ØØ
B7E1:9Ø
3ØØ:2Ø 93 B7 4C 59 FF
3ØØG
```

12) Now we need to patch the boot sequence to execute our routine. We will do this by moving RWTS and boot code down to $15ØØ.1FFF. While doing this, we will leave room for our boot routine.

```
1ØØØ:ØØ
1ØØ1<1ØØØ.1FFEM
15ØØ<B6ØØ.B6FFM
1793<B793.BFFFM
17BØ:F2
17EA:Ø1 ØØ Ø9 ØF
17FØ:ØØ 97 9Ø ØØ Ø1 ØØ
```

13) It's now ready to take our boot routine so insert WORK and reboot.

```
6 CTRL P
```

14) And load that big hexdump from the WORK disk.

```
BLOAD BOOTCODE, A$16ØØ
```

15) Finally, insert COPY and write the code to track $Ø

```
CALL -151
3ØØ:2Ø 93 B7 4C 59 FF
B7EA:Ø1 ØØ ØØ ØA
B7FØ:ØØ 1F ØØ ØØ Ø2 ØØ
B7E1:ØB
15FE:B5 ØA
3ØØG
```

16) The disk will now boot and operate correctly, but in the process of standardizing the ZARDAX code, we also freed up tracks $A through $1Ø. What we want to do now is modify the VTOC (Volume Table Of Contents) at sector $Ø of track $11 to show that these tracks are now free.

```
3ØØ:2Ø E3 Ø3 2Ø D9 Ø3 4C 59 FF
B7EA:Ø1 ØØ 11 ØØ
B7FØ:ØØ 1Ø ØØ ØØ Ø1 ØØ
3ØØG
1Ø6Ø:FF FF
1Ø64:FF FF
1Ø68:FF FF
1Ø6C:FF FF
1Ø7Ø:FF FF
1Ø74:FF FF
1Ø78:FF FF
B7F4:Ø2 ØØ
3ØØG
3DØG
```

17) The disk is completely unprotected at this time. But one of our goals was to enable the SETUP program on the ZARDAX UTILITIES disk to change the parameters on our unprotected version. It turns out that this is very simple to do. The file LOWER.WRITER on our ZARDAX disk contains the code which is used to write our paramters directly to the disk. If you wish to look at these yourself, look at lines 9Ø and 942 of the SETUP program on the UTILITIES disk. All we need to do is change the code in this file so that it will write the parameter information to the correct sectors on our copy. Insert COPY in your disk drive and modify LOWER.WRITER.

```
BLOAD LOWER.WRITER, A$47ØØ
CALL -151
47Ø3:Ø2 Ø2 Ø7 5F 48
4771:38 E9
478Ø:38 E9
4788:BØ
3DØG
BSAVE LOWER.WRITER, A$47ØØ,
L$FØ
```

You can now delete most of the files on the ZARDAX COPY, since many of them were involved in the load process and will not work with the unprotected copy anyway. The only files you must retain are USER, GLOSSARY, LOWER.WRITER, LOWER, OK and KBDMACRO. If you use the software routines for displaying text on the hi-res screen, you will also need to retain the appropriate LCIO files on the disk. You can delete the file OK if you delete the lines in the SETUP program on the UTILITIES disk that make reference to this file. If you choose to do this, the actual name of the file is OK⌃A ⌃B.

That's it. You can now make as many copies of this disk as you need for your situation and configure each for your specific needs. In order to change paramters, you will need to boot the UTILITIES disk and run SETUP as the copy will no longer present that option during the load process.

Cookbook Instructions

1) Label your two blank disks WORK and COPY.

2) Insert your DOS 3.3 System Master and load DOS into memory:

```
PR#6
```

3) Insert the blank disk labeled WORK and initialize it with an empty HELLO program:

```
NEW
INIT HELLO
```

4) Insert the DOS 3.3 System Master and load the copy programs:

```
LOAD COPYA
BLOAD COPY.OBJØ
```

5) Insert WORK disk, modify COPYA, and save:

```
BSAVE COPY.OBJØ, A$2CØ, L$1ØB
```

Type in the modifications to COPYA made in the article.

```
SAVE COPYA.MOD
```

6) Type in the boot routine hexdump in the article and BSAVE it.

```
BSAVE BOOTCODE, A$16ØØ, L$132
```

7) Insert WORK and run our modified copy program:

```
RUN COPYA.MOD
```

8) When the program stops loading and displays the prompt for slot, remove WORK, insert ZARDAX and COPY and answer the prompts. Answer 17 to "STARTING TRACK: " , 34 to "ENDING TRACK: " and "Y" to "FORMAT THIS DISK? ".

9) Insert the ZARDAX original in the disk drive and read in the first load sector:

```
CALL -151
3ØØ:2Ø E3 Ø3 2Ø D9 Ø3 4C 59 FF
B93F:AD
B7EA:Ø1 ØØ Ø3 ØF
B7FØ:ØØ B4 ØØ ØØ Ø1 ØØ
3ØØG
```

10) Now use this code to read in the rest of the protected portion of ZARDAX:

```
B485:ØØ Ø3
3ØØ:48 8A 48 98 48 18 A9 59
3Ø8:8D 35 B1 A9 FF 8D 36 B1
31Ø:68 A8 68 AA 68 4C ØØ B1
B417G
```

11) Turn the drive off:

```
CØE8
```

12) Insert COPY and write this code to our copy:

```
B7EA:Ø1 ØØ Ø9 ØF
B7FØ:ØØ 97 ØØ ØØ Ø2 ØØ
B7E1:9Ø
3ØØ:2Ø 93 B7 4C 59 FF
3ØØG
```

13) Now move some code for the boot sequence:

```
1ØØØ:ØØ
1ØØ1<1ØØØ.1FFEM
15ØØ<B6ØØ.B6FFM
1793<B793.BFFFM
17BØ:F2
17EA:Ø1 ØØ Ø9 ØF
17FØ:ØØ 97 9Ø ØØ Ø1 ØØ
```

14) Insert WORK and reboot:

```
6⌃P
```

15) Retrieve our boot code:

```
BLOAD BOOTCODE, A$16ØØ
```

16) Insert COPY and write this code into the boot sequence:

```
CALL -151
3ØØ:2Ø 93 B7 4C 59 FF
B7EA:Ø1 ØØ ØØ ØA
B7FØ:ØØ 1F ØØ ØØ Ø2 ØØ
B7E1:ØB
15FE:B5 ØA
3ØØG
```

17) Fix the VTOC to show that tracks 1Ø through 16 are free:

```
3ØØ:2Ø E3 Ø3 2Ø D9 Ø3 4C 59 FF
B7EA:Ø1 ØØ 11 ØØ
B7FØ:ØØ 1Ø ØØ ØØ Ø1 ØØ
3ØØG
1Ø6Ø:FF FF
1Ø64:FF FF
1Ø68:FF FF
1Ø6C:FF FF
1Ø7Ø:FF FF
1Ø74:FF FF
1Ø78:FF FF
B7F4:Ø2 ØØ
3ØØG
3DØG
```

18) Change LOWER.WRITER so that SETUP will work:

```
BLOAD LOWER.WRITER, A$47ØØ
CALL-151
47Ø3:Ø2 Ø2 Ø7 5F 48
4771:38 E9
478Ø:38 E9
4788:BØ
3DØG
BSAVE LOWER.WRITER, A$47ØØ,
L$FØ
```

That's all.

UNRESTRICTED AMPERSAND

by Phil Goetz

The ampersand (&) command is very useful for calling a machine language subroutine from Applesoft. The only problem is that the ampersand vector at $3F5-3F7 can only point to one subroutine at a time. This would appear to mean that you can only use one ampersand routine at a time. With the technique presented here, however, you can use an unlimited number of ampersand commands at the same time.

How It Works

In order to determine which command is being used, all ampersand commands present at the same time must be distinguishable from one another. One may be just an &, but the others must be followed by a keyword or a symbol, i.e. &D. The technique here involves layering the ampersand routines.

For instance, say a routine activated by "&X" is in memory and you want to add one activated by "&Y". When setting it up, you would have your "&Y" routine load the current ampersand vector (pointing to "&X"), place it somewhere within "&Y', and store its own starting address in the & vector. Upon later & commands, BASIC would go to the ampersand vector at $3F5, which would point to the &Y routine. This routine would then fetch the next byte in the BASIC program without incrementing the CHRGET pointers. The CHRGET pointers must not be incremented during the initial check so that no characters will be skipped over. The byte may be a keyword like "HOME" or a symbol or ASCII character of your choosing. The routine would compare its value to $59, which is the ASCII value of "Y". If it were not the same, it would jump to the "&X" routine using the pointers which it stored within itself upon initial execution.

How to Use It

An unlimited number of ampersand commands can be stacked upon each other in this manner. The first one to be put in memory is set up normally, but every subsequent routine must follow a method similar to what was described above. The following listing is an example of the necessary code. Note that you must still be careful not to load two routines in the same place. Notice also that if a command is meant to be activated by just an &, it must be the first to be installed and thus the last in line upon execution of an &. This is because it would take several times as much work to determine whether the & was alone or followed by something, especially if it were at the end of a line.

```
* Example of ampersand routine stacking:

SETUP   LDA $3F6        ;Get old address
        STA NEWVEC      ;from $3F6-3F7
        LDA $3F7        ;and create new
        STA NEWVEC+1    ;vector for it

        LDA #$4C        ;Put JMP to new
        STA $3F5        ;& routine at
        LDA #TESTY      ;$3F5-$3F7
        STA $3F6
        LDA #/TESTY     ;(high byte)
        STA $3F7
        RTS             ;Return to caller

NEWVEC  .HS 0000        ;2 bytes reserved
                        ;for vector to next
                        ;& routine

TESTY   JSR $B7         ;get next char
        CMP #$59        ;ASCII "Y"?
        BEQ .1          ;Yes, do "&Y"
        JMP (NEWVEC)    ;No, next & vector
.1      JSR $B1         ;Increment CHRGET

AMPERY  .               ;New amper. routine
        .               ;starts here
```

Adventure Tips

PLANETFALL

- Search all desks.
- Show Floyd the computer room. He'll get an idea.
- Send Floyd into a small door and then listen to what he says.
- He's childish, but loyal, and he sometimes does what you ask him.
- Once you've fixed the computer and gotten through the lab, head for the room with the tasteless mural.

By Paul Wilson

SORCERER

- If you have anything to mail, put it in the receptacle before the mailman comes.
- The journal and infotater will tell you how to open the trunk in the cellar.
- Getting bitten when you try to get a scroll from the cannon? Check the infotater again for something that bites.
- The river can be dried up.
- Fly over the riverbank and drawbridge from the meadow.
- Say, like the guy you're looking for.
- Kill time in bozbar land for safety's sake (2 patrons in your system unnatural death).
- The infotater is a booklet that comes with their wares.
- I urge that you do not accept an offer of SORCERER from them.
(Input younger self the combination is xxx)

By Paul Wilson

WIZARD AND THE PRINCESS

- Use the beach where the anchor was.
- Remember the blanket is good to use for stuffing up holes.
- The anchor is a necessary tool in the recovery of the shovel.
- To find the rock to kill the snake you must first watch carefully as to which side all the rocks appear on, but please ingnore the first rock or you will die.
- Don't forget to fill the flask an ocean trip can make you very thirsty.

By Debbie Holloway

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

DuoDisk & IIc owners write.

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

Public Domain Software
SPECIAL
From The COMPUTER LEARNING CENTER!

GET Over 170

BUSINESS & FINANCE

Programs FOR ONLY

$28.^{00}

- Forecast your investment profits, and evaluate the worth of your stock portfolio
- Experiment with Visicalc formulas, real estate and stock plots and design a business or personal letter writing format
- Organize your household expenses
- Balance your checkbook, manage your loans, payroll

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

Annuity
Bond Price & Interest
Budget Monthly
Decision Matrix
Financial Pak
Keogh Savings Program
Mortgage Calculation
Regular Deposits I
Stocks
Calendar Personal
Letter Writer
Phone List

Real Estate Plot
Visicalc Formulas
Business Finance
Check Stub
Household Exp. Profile
Income Tax 1040 For 77
Inventory Company
Average Growth Rate
Check Book Balancer
Income Taxes
Interest Earned
Loan Balance

Savings Growth
Treasury Bill Valuation
Address File
Compound Int. Tables
File Manager
Home Accounting
Names Search
File Cabinet I
Family Finance
Gen Ledger Printer
Constar
etc...

ORDER NOW!

You will also receive a complete catalog of Public Domain Software available from the CLC 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-HC, 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 175 volumes, which will run on Apple][Plus computers and Apple compatibles. Most will also run on Apple //e and //c.)

Back Issues & Library Disks order form

Issue	Mag $4.75	Disk $9.95	Both $12.95
31	□	□	□
30	□	□	□
29	□	□	□
28	□	□	□
27	□	□	□
26	□	□	□
25	□	□	□
24	□	□	□
23	□	□	□
22	□	□	□
☆21 . .	□	□	□
20	□	□	□
19	□	□	□
☆18 . .	□	□	□
17	□	□	□
16	□	□	□
15	NA	□	NA
14	NA	□	NA
13	□	□	□
12	NA	□	NA
☆11 . .	NA	□	NA
10	NA	□	NA
9	NA	□	NA
*8	NA	□	NA
*7	□		□
6	NA	□	NA
*4	□	□	□
*3	NA		NA
*Core 2	□		□
*2	NA	□	NA
*1	□		□
*Core 1	□		NA
Core 3	□	□	□

Special "Both" disk & magazine combination orders apply to one issue and its corresponding disk.

* Some disks apply to more than one issue and are shown as taller boxes.

☆ We have a limited supply of these issues.

BACK ISSUES

of COMPUTIST *(formerly Hardcore COMPUTIST)*

are still available, though some issues (marked NA) are sold out, Library disks are available for ALL issues of COMPUTIST.

Don't *T Y P E* in programs that appear in COMPUTIST.

Order the Library Disk, instead!

For each issue of COMPUTIST, a Library Disk containing all the programs that appeared in that issue 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 order either the magazines or library disks or BOTH MAGAZINE AND DISK AND SAVE $1.75. Documentation for Library Disks is in the corresponding issue.

Send me the back issues and/or library disks indicated:

Name ______________________________ ID# ________

Address ______________________________

City ______________ State ________ Zip ____________

Country ______________ Phone ______________

VISA MasterCard ________-________-________-________ Exp. ________

Signature ______________________________ CP31

Send check or money order to:

COMPUTIST PO Box 110846; Tacoma, WA 98411

Most orders shipped UPS so please use street address. Offer good while supply lasts. In Washington state: add 7.8% sales tax.

Back Issue Rates For Foreign Orders (effective immediately)

FOREIGN MAGAZINE ORDERS

Price for each magazine includes shipping.

	1 - 2 copies	3 to 4 copies	5 or more copies
Canada/Mexico	$8.00 each	$7.00 each	$6.00 each
Other Foreign	$14.25 each	$13.25 each	12.25 each

FOREIGN DISK ORDERS

Disks are $11.94 each (includes shipping). Special "Both" disk and magazine combinations shown do NOT apply to Foreign orders. US funds drawn on US banks. All foreign orders sent AIR RATES.

Description of Available Back Issues

30 *Softkeys* | Millionaire | SSI's RDOS | Fantavision | Spy vs. Spy | Dragonworld | *Readers' Softkeys* | King's Quest | Mastering the SAT | Easy as ABC | Space Shuttle | The Factory | Visidex 1.1E | Sherlock Holmes | The Bards Tale | *Feature* | Increasing Your Disk Capacity | *Core* | Ultimaker IV, an Ultima IV Character Editor | ....

29 *Softkeys* | Threshold | Checkers v2.1 | Microtype | Gen. & Organic Chemistry Series | Uptown Trivia | Murder by the Dozen | *Readers' Softkeys* | Windham's Classics | Batter Up | Evelyn Wood's Dynamic Reader | Jenny of the Prairie | Learn About Sounds in Reading | Winter Games | *Features* | Customizing the Monitor by Adding 65C02 Disassembly | *Core* | The Animator | ....

28 *Softkeys* | Ultima IV | Robot Odyssey | Rendezvous | Word Attack & Classmate | Three from Mindscape | Alphabetic Keyboarding | Hacker | Disk Director | Lode Runner | MIDI/4 | *Readers' Softkeys* | Algebra Series | Time is Money | Pitstop II | Apventure to Atlantis | *Feature* | Capturing the Hidden Archon Editor | *Core* | Fingerprint Plus: A Review | Beneath Beyond Castle Wolfenstein (part 2) | ....

27 *Softkeys* | Microzines 1-5 | Microzines 7-9 | Microzines (alternate method) | Phi Beta Filer | Sword of Kadash | *Readers' Softkeys* | Another Miner 2049er | Learning With Fuzzywomp | Bookends | Apple Logo II | Murder on the Zinderneuf | *Features* | Daleks: Exploring Artificial Intelligence | Making 32K or 16K Slave Disks | *Core* | The Games of 1985: part II | ....

26 *Softkeys* | Cannonball Blitz | Instant Recall | Gessler Spanish Software | More Stickybears | *Readers' Softkeys* | Financial Cookbook | Super Zaxxon | Wizardry | Preschool Fun | Holy Grail | Inca | 128K Zaxxon | *Features* | ProEdit | *Core* | Games of 1985 part I | ....

25 *Softkeys* | DB Master 4.2 | Business Writer | Barron's Computer SAT | Take 1 | Bank Street Speller | Where In The World Is Carmen Sandiego | Bank Street Writer 128K | Word Challenge | *Readers' Softkeys* | Spy's Demise | Mind Prober | BC's Quest For Tires | Early Games | Homeword Speller | *Features* | Adding IF THEN ELSE To Applesoft | *Core* | DOS To ProDOS And Back | ....

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

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

22 *Softkeys* | Miner 2049er | Lode Runner | A2-PB1 Pinball | *Readers' Softkeys* | The Heist | Old Ironsides | Grandma's House | In Search of the Most Amazing Thing | Morloc's Tower | Marauder | Sargon III | *Features* | Customized Drive Speed Control | Super IOB version 1.5 | *Core* | The Macro System | ....

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

20 *Softkeys* | Sargon III | Wizardry: Proving Grounds of the Mad Overlord and Knight of Diamonds | *Reader' Softkeys* | The Report Card V1.1 | Kidwriter | *Feature* | Apple][Boot ROM Disassembly | *Core* | The Graphic Grabber v3.0 | Copy II+ 5.0: A Review | The Know-Drive: A Hardware Evaluation | An Improved BASIC/Binary Combo | ....

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

18 *Softkeys* | Scholastic Version of Bank Street Writer | Applewriter //e | SSI's Non-RDOS Disks | *Readers' Softkeys* | BPI Accounting Programs | DesignWare Programs | *Features* | Installing a Free Sector Patch Into Applewriter //e | Simple Copy Protection | *Core* | The Games of 1984: In Review | 65C02 Chips Now Available | Checksoft v2 | ....

17 *Softkeys* | The Print Shop | Crossword Magic | The Standing Stones | Beer Run | Skyfox | Random House Disks | *Features* | A Tutorial For Disk Inspection and the Use Of Super IOB | S-C Macro Assembler Directives (reprint) | *Core* | The Graphic Grabber For The Print Shop | The Lone Catalog Arranger v1.0 Part 2 | ....

16 *Softkey* | Sensible Speller for ProDOS | Sideways | *Readers' Softkeys* | Rescue Raiders | Sheila | Basic Building Blocks | Artsci Programs | Crossfire | *Feature* | Secret Weapon: RAMcard | *Core* | The Controller Writer | A Fix For The Beyond Castle Wolfenstein Softkey | The Lone Catalog Arranger Part 1 | ....

13 *Softkeys* | Laf Pak | Beyond Castle Wolfenstein | Transylvania | The Quest | Electronic Arts | Snooper Troops (Case 2) | DLM Software | Learning With Leeper | TellStar | *Core* | 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 | *Feature* | Locksmith 5.0 and Locksmith Programming Language | ....

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

4 *Softkeys* | Ultima II | Witness | Prisoner II | Pest Patrol | Adventure Tips for Ultima II & III | Copy II Plus PARMS Update | *Feature* | Ultima II Character Editor | ....

1 *Softkeys* | Data Reporter | Multiplan | Zork | *Features* | PARMS for Copy II Plus | No More Bugs | APT's for Choplifter & Cannonball Blitz | *'Copycard' Reviews* | Replay | Crackshot | Snapshot | Wildcard |

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 | Dragon Dungeon | ....

CORE 2 ....Utilites: 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 | ....

Back issues not listed are no longer available.

But disks are still available for ALL sold-out issues !

Use the order form on the other side of this page

Writer's Guide

COMPUTIST

is a monthly magazine dedicated to the serious user of the Apple (or compatible) computer. COMPUTIST welcomes articles on a variety of subjects in all levels of technical difficulty but requires accurate data, technical competence, correct English usage, readable style, and fully defined jargon and buzzwords.

MANUSCRIPT MECHANICS

All manuscripts must be typed or printed on one side of the paper. Text should be double-spaced.

Printouts should use a non-compressed font with both upper and lower case. A letter quality mode is preferred, with each page torn at the perforation only. Pages need not be stapled together.The cover page of each manuscript should contain the following data:

TITLE OF WORK
FULL NAME OF AUTHOR
ADDRESS
PHONE NUMBER

Each page of the manuscript and program listing should include the author's name, the title of the work, and the page number in the upper right hand corner.

The article and any accompanying program **SHOULD BE SUBMITTED AS A STANDARD TEXT FILE ON A DOS 3.3 DISK.** Label the disk with the title of the work and the author's full name and address. **ON DISK, TEXT MUST BE SINGLE-SPACED ONLY.** Please identify your editing program.

Original disks are always returned as soon as possible. Other materials will be returned only when adequate return packaging and postage is enclosed. We are not responsible for unreturned submissions. We *will guarantee* the return of original commercial disks mailed to us for verification of an accompanying softkey.

You will be notified of the status of your submission within 4 to 6 weeks after it is received if the article is a softkey accompanied by an original disk. Please submit completed manuscripts directly; do not query first. Previously published material and simultaneous submissions are not accepted.

SUBJECTS

We prefer material on these topics:

1) Original program/article combinations
2) General articles (Apple computing)
3) Softkeys
4) Advanced Playing Techniques (APT's)
5) Hardware modifications
6) DOS modifications
7) Product reviews (hardware and software)
8) Utilities
9) Bit Copy Parameters

WRITING YOUR ARTICLE

Observe the following points of style:

A. Always assume that your reader is a novice and explain all buzzwords and technical jargon. Pay special attention to grammar and punctuation; we require technical competence but also good, readable style.

B. Whenever appropriate, a list of hardware and software requirements should be included at the beginning of the manuscript. When published, this list will be offset from the main text.

C. Include the name and address of the manufacturer and the price when a commercial program is mentioned. This is of particular importance in PRODUCT REVIEWS.

D. When submitting programs, first introduce the purpose of the program and features of special interest.` Include background information describing its use. Tips for advanced uses, program modifications, and utilities can also be included. Avoid long print statements and use TABs instead of spaces.

Remember: A beginner should be able to type the program with ease.

E. A PROGRAM is not accepted for publication without an accompanying article. These articles, as well as articles on **hardware** and **DOS modifications** MUST summarize the action of the main routines and include a fully remarked listing.

F. GENERAL ARTICLES may include advanced tips, tutorials, and explorations of a particular aspect of Apple computing.

G. SOFTKEYS of any length are acceptable and must contain detailed step-by-step procedures. For each softkey, first introduce the locking technique used and then give precise steps to unlock the copy-protected program. Number each step whenever possible. We accept articles which explain locking techniques used in several programs published by the same company.

H. When altering game programs, the changes made are sometimes extensive enough to warrant the title of ADVANCED PLAYING TECHNIQUE (APT). APTs can deal with alterations to a program, deleting annoying sounds, acquiring more points in play and avoiding hazards. Again, provide step-by-step instructions to complete each APT and explain each step's function. APT's of 100 words or more are preferred.

AUTHOR'S RIGHTS

Each article is published under the author's byline. As a rule, all rights, as well as one-time reprint rights are purchased. Purchase of exclusive rights to programs is required; however, alternate arrangements may be made with individual authors depending on the merit of the contribution.

PAYMENTS

COMPUTIST pays upon publication. Rate of payment depends on the amount of editing required and the length of the article. Payment ranges from $20 to $50 per typeset page for an article. We also pay $10 to $20 for short softkeys and APT's. A fully explained softkey accompanied by the commercial disk for verification may earn up to $50 per typeset page.

Please mail your submissions to:

COMPUTIST
Editorial Department
PO Box 110846-T
Tacoma, WA 98411

FREE

your disks from their uncopyable status

Now you can make necessary backups
of your locked-up commercial software
with

The BOOK of Softkeys

shows you
how to softkey
(undo copy-protection on)
commercial software.

Volumes II and III are being compiled now!

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

If you want to make backups
then you want The Book Of Softkeys!

For your copy of The Book Of Softkeys Volume 1, send $20 (Foreign orders add 20%. U.S. funds drawn on U.S. banks. Washington state orders add 7.8% sales tax.) to:
SoftKey Publishing; PO Box 110846-T; Tacoma, WA 98411