For The Serious User Of Apple][Computers

COMPUTIST

Issue No. 36 October 1986

USA $3.75
Canada/Mexico $7.00
All Others $13.25

Softkeys For:

Flight Simulator II
AutoDuel
Critical Reading
Troll's Tale
Robot War
General Manager
Plasmania
Telarium Software

Core:

The Bard's Dressing Room: a character editor

Feature:

The Bus Monitor

(Page 11)

COMPUTIST
PO Box 110846-T
Tacoma, WA 98411

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

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[CTRL]P

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

Requirements: Most of the programs and softkeys which appear in 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. 32. 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. 32 as a back issue or order Program Library Disk No. 32.

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 (Δ). 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.

S.O.S.

(Save On Software)

Title	Publisher	Suggested Retail	Customer Cost	QTY	Total Cost
Recommended Literature:					
☐ Beneath Apple DOS (Book)	Quality Software	$19.95	$16.00	____	_____
☐ Beneath Apple ProDOS (Book)	Quality Software	$19.95	$16.00	____	_____
☐ Disk Edit (Book of Softkeys vol 1)	SoftKey	-	$12.95	____	_____
Recommended Software:					
☐ Global Program Line Editor	Beagle Bros	$49.95	$35.25	____	_____
☐ Super IOB (Issue No. 32 w/disk)	SoftKey		$10.95	____	_____
☐ Magic Window // (specify][or //e)	Artsci	$149.95	$106.00	____	_____
☐ Bag of Tricks II	Quality Software	$49.95	$39.75	____	_____
Miscellaneous Bargains					
☐ Dazzle Draw	Broderbund	$59.95	$47.50	____	_____
☐ F-15 Strike Eagle	Microprose	$34.95	$28.00	____	_____
☐ The Print Shop	Broderbund	$49.95	$39.75	____	_____
☐ Flight Simulator II	Sublogic	$49.95	$44.00	____	_____
☐ Night Mission Pinball	Sublogic	$34.95	$30.75	____	_____
☐ Exodus Ultima III	Origin Systems	$59.95	$47.75	____	_____
☐ Hitchhiker's Guide to the Galaxy	Infocom	$39.95	$31.00	____	_____
☐ Witness	Infocom	$39.95	$31.00	____	_____
☐ Dino Eggs	Microlab	$40.00	$20.00	____	_____
☐ Zork III	Infocom	$44.95	$35.00	____	_____
			Subtotal		_____
			Shipping*		_____
			Total		_____

*Domestic Shipping and Handling: $2.00 per item. Five or more items FREE.

Name _____ ID# _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone _____

VISA _____-_____-_____-_____ Exp. _____

Signature _____ CP36

To order, complete order form and mail to: COMPUTIST PO Box 110937-SOS Tacoma, WA 98411

Offer good while supplies last. Washington residents add 7.8% sales tax. Foreign orders inquire as to appropriate shipping and handling fees. Limited offer, expires December 31, 1986.

FREE COMPUTIST

YES! You Get 2 FREE ISSUES
for every NEW subscriber that you sign up.

We'll extend your current subscription by 2 issues for every new paid subscription you get to fill out the subscriber forms below.

BUT...First Class subscribers MUST recruit First Class subscriptions to receive the 2 extra issues.
This rule (same mailing class) also applies to Foreign rates.
Standard US subscribers can recruit any new subscriptions and extend their standard subscriptions.
US funds drawn on US bank. Send check/money order to: COMPUTIST PO Box 110846-T Tacoma, WA 98411

CURRENT SUBSCRIBER

☐ Add two issues to this subscription.
☐ Renew my subscription. Payment is enclosed.
☐ US: $20 US First Class: $24 ☐ Can/Mex: $34 ☐ Other Foreign: $60

Name _____ ID# _____
Address _____
City _____ State _____ Zip _____
Country _____ Phone _____
VISA _____ Exp. _____
Signature _____ CP36

If you let your subscription expire, you are considered a NEW subscriber and are NOT eligible for the free issues.

NEW SUBSCRIBER

Yes, I want to subscribe to your fine publication.
☐ US: $20 US First Class: $24 ☐ Can/Mex: $34 ☐ Other Foreign: $60

Name _____
Address _____
City _____ State _____ Zip _____
Country _____ Phone _____
VISA _____ Exp. _____
Signature _____ CP36

OR SAVE MORE ON A 12 ISSUE NEW OR RENEWED SUBSCRIPTION

Special prices for a limited time only.
Cannot be used with any other promotional offer.
Prices are for an annual subscription (12 issues)
Yes, sign me up for 12 issues.
☐ US: $32 ☐ US First Class, Canada, Mexico: $45 ☐ Other Foreign: $75

Name _____ ID# _____
Address _____
City _____ State _____ Zip _____
Country _____ Phone _____
VISA _____ Exp. _____
Signature _____ CP36

Attention COMPUTIST Collectors

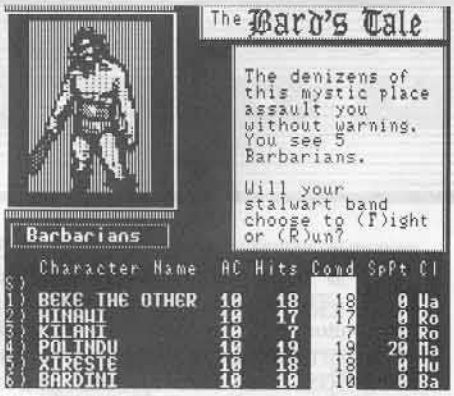
The following back issues are practically gone. Once they have sold-out, they will no longer be available in magazine form to our readers.

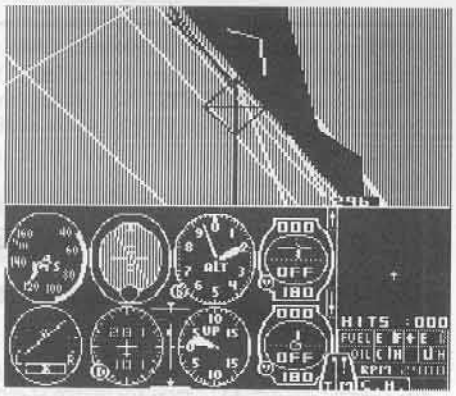
4 *Softkeys* | Ultima II | Witness | Prisoner II | Pest Patrol | Adventure Tips for Ultima II & III | Copy II Plus PARMS Update | *Feature* | Ultima II Character Editor | .....

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 | .....

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 | .....

First Come, first served. Some issues may be slightly damaged. No reservations allowed, prepayment only with [illegible]. To order, phone (206) 474-5750 or use the order form on page 32. For phone orders, please have subscriber ID number and ship-to address ready. Normal back issue rates apply to foreign orders. Washington State residents add 7.8% sales tax.





This month's cover:
Graphics from Mindscape's "Color Me."

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.

Unsolicited manuscripts are assumed to be submitted for publication at our standard rates of payment. SoftKey publishing purchases all and exclusive rights. For more information on submitting manuscripts, see the writers guide on the inside back cover.

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.

COMPUTIST

Issue 36 October 1986

Publisher/Editor: Charles R. Haight **Managing Editor:** Ray Darrah
Technical Editor: Robert Knowles **Circulation:** 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:

features:

core:

APT:

departments:

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.

Opinions expressed are not necessarily those of COMPUTIST or SoftKey Publishing.

Some Philosophy

Everyone who reads and submits articles to COMPUTIST is on the same side of the copy protection issue. We feel we should be able to back up our software. We also should act like a big family when it comes to copy protection. This is why I feel the comments in M. M. McFadden's letter (COMPUTIST No. 31 page 4) about my article on deprotecting F-15 Strike Eagle were uncalled for. Everybody has a different way of deprotecting a piece of software. Also most people don't have access to several copies of a program to see if there are multiple protection schemes. Mr. McFadden's off the cuff comments don't seem to do much good. If you can't save the files back to the disk, who cares about a catalog track. Between my article and all the letters that were written, most people should be able to back up their copy of F-15 Strike Eagle.

So whenever anyone has a different way of backing up a piece of software or finds a flaw in an article, just write in and mention it without a lot of name calling.

Thank you.

Larry Jasonowicz
Marseilles, IL

Mr Jasonowicz: We agree with you that those who wish to backup their software should act as one big family. We apologize for Mr. McFadden's comments.

King's Quest & Black Cauldron

I read your wonderful magazine since the first old issue and I always enjoy a lot of your articles.

Herewith I write you for two reasons: to show you how to unlock the last two hi-res adventure games from Sierra On-Line, (King's Quest II and The Black Cauldron) and give public compliments to the authors of the bit copy programs Echo Plus, Copy II Plus 6.0, Locksmith 6.0 and EDD IV Plus, whose new versions are very powerful.

The new hi-res adventures from Sierra On-Line are very beautiful and amusing to play, although they load the graphic pages very slowly. Worst of all, both have been protected and even a copy with a bit copy program is difficult to make, owing to a nibble count they use on track $0 of the boot side. Yet, the protection routine is the same in both programs so that unlocking one unlocks the other too.

With the Core Disk Searcher program (published in COMPUTIST No. 12), I've searched the boot disks for two hex bytes 8C C0 ($C08C) and, besides the first tracks used by the DOS, I've discovered a routine on track $11 sector $F, which is the nibble count protection. I put a $60 at the beginning of it and voila, the disks worked.

So, resuming: copy with COPYA or other copy program all the sides of the program; insert the boot side of the program you want to unlock and with a sector editor (e.g. The Inspector) read the track $11 sector $F; starting at byte $5 you'll see A9 00 (LDA #00); change the A9 to 60 (RTS) and write back the sector. The disk is now unprotected and easily copyable.

I hope you'll find this unlocking technique interesting.

Thank you for the time spent reading this letter and I am looking forward to reading your next wonderful issue.

Guido Bertoncini
Bergamo, Italy

Wolfenstein APT's

Following each of the APT's in the "Beneath Beyond Castle Wolfenstein" article in COMPUTIST No. 13 has led to much enjoyment with the popular game from MUSE. There is however, one bug that I have found. The Reset patch does dump you into the monitor when Control-Reset is pressed but going into the monitor this way disconnects you from DOS. This isn't apparent until the program tries to access the disk and freezes. The solution is: as soon as you enter the monitor, reconnect the DOS with **A851G** first and then perform any APTs that you want. This prevents much frustration.

One last note: Does anyone out there know how to get the deprotected AppleWriter //e (COMPUTIST No. 18) to work with a fast DOS such as Diversi-DOS?

Jim S Hart
Jacksonville, NC

Eight Cities of Gold

I'd just like to share a couple of updates on some softkeys. Concerning the softkey for Fantavision in COMPUTIST No. 30, my copy had address epilogues of FF FF, so by adding these lines to the controller, it will work just fine on that version!

```
1025 POKE 47405 ,24 : POKE 47406 ,96 : POKE 47497
     ,24 : POKE 47498 ,96
1065 POKE 47405 ,208 : POKE 47406 ,19 : POKE
     47497 ,208 : POKE 47498,183
```

Going back a few issues to the softkey for Seven Cities of Gold in COMPUTIST No. 24, I had no trouble using the method described and ending up with a deprotected copy. But upon playing the game, I had nothing but trouble. Imagine discovering all those lands, amassing all that wealth, only to get home and find myself without anything, including my men and my ships!!! For those of you who had problems as I did, try this method instead.

1) Copy both sides of the disk ignoring errors on tracks 5 & 6 or skip them completely.

2) On the boot side, make these sector edits:

TRACK	SECTOR	BYTE	CHANGE TO
$01	$06	$08	$62
$0E	$07	$75	$EA
$0E	$07	$76	$EA
$0E	$07	$77	$EA

I hope these help anyone who ran into problems. Keep those softkeys coming!!!

M Ferreira
Santa Rosa, CA

input

New Zoom Graphics

COMPUTIST published my softkey for Zoom Graphics in issue 12 (pages 9-10). As published, that softkey will not work on the version of Zoom Graphix with a manufacture date of 10/5/83 (this version is the first to include the Apple DMP on the list of printers). To determine the date of manufacture of a Zoom Grafix disk, see the last column of the original article.

The following modifications to the softkey in issue 12 will copy the 10/5/83 version:

2) substitute:

```
70
CALL -151
B7C0:18
B942:18
B954:29 00
B990:29 00
3D0G
RUN
```

Ignore the terrible noises coming from the disk drive - leave the room if you have to.

3) Disregard the instructions in the original article. Just reboot your favorite DOS, type FP, and go to step 4 of the original article.

14) Use these new BSAVEs:

```
BSAVE GRAFIX.INFO, A$800,L$4D5
BSAVE GRAFIX.OBJ, A$9000,L$9B1
```

Those using multiple drives, especially hard drives, may wish to modify the code in lines 270, 300, and use the space from 300-310 in Grafix Part II in order to request Slot, Drive, and Volume information from the user. The variables SL$, DR$, and VL$ are not used elsewhere in the program, and are available for that purpose.

Michael Decker
Hermitage, TN

Mastering Master Word

When a back-up for Workshark "Master Word" program is made with Locksmith 5.0 Quick Copy it appears to copy perfectly yet it will not run.

Listing the HELLO program discloses line 10 CALL XXXXX which must send the program to limbo because deletion of line 10 after copying results in a runnable copy.

Continue the good work.

Not being an expert, it is suggested you have one of your more expert readers check out the above and verify it. I learned of it second-hand.

A Subscriber
Santa Ana, CA

PFS meets Unidisk

Robert James' sector edits for the ProDOS PFS series on page 5 of COMPUTIST No. 31 works for PFS:GRAPH as well as the other programs. Unidisk owners can manufacture a very "interesting" disk by transferring all of the files of the PFS: series to a single unidisk, as long as they remember that they really don't need multiple copies of ProDOS, and the QUARK program. The result will be a disk that functions like a super-Appleworks, as they can easily exit from one main menu to the next main menu within the PFS series. Apparently, you can make this unidisk collage boot up into whichever of the PFS programs you want to by making sure that you copy the .SYSTEM file and ProDOS first onto the unidisk, and then use the EXIT function to get into the next. If anyone is familiar enough with the ins and outs of ProDOS option menus, they might be able to write a HELLO program that allows direct entry into the program of choice, but this quick & dirty method of booting into one and exiting into the next does work pretty well, too...

Stanley Planton
Chillicothe, OH

Gato again

Here's a softkey for version 1.3 of GATO.

The disk is written with even tracks beginning with the standard D5 AA 96 address header, and odd tracks having D4 AA 96 as a header. The address trailer begins with AF but the remaining digits change from sector to sector.

My first step in deprotecting GATO was to write it to a normal DOS disk. I found I could then catalog the disk from UCSD Pascal. I tried to execute SYSTEM.STARTUP from Pascal, even though I only have the 64K version, and the disk buzzed and whirred for a while until I got a stack overflow error. It also would be interesting for someone who has 128K Pascal to try, however. Also interesting is to run the Pascal Libmap program and scan through the intrinsics in SYSTEM.LIBRARY. ...but on to the softkey.

The following controller for Super IOB completely deprotected this version of GATO. It was necessary to semi-resurrect the Ignore Ending Marks subroutine of the original version of Super IOB as I couldn't figure an elegant way to handle the sector-by-sector changing ending marks with the current subroutine (that is what the pokes in line 1020 take care of).

GATO also has code which checks the disk for its original signature, the teeth of which were pulled through the time-honored art of boot-tracing. Super IOB makes the necessary sector edits.

I would suggest trying the controller as is. If by any chance the publisher is moving the protection checks around on the disk and your copy doesn't work, delete the GOSUB 310 in line 1060, copy the disk, and search for the following byte sequences. Here is where I found them:

```
Tk 0, Sec E bytes: F4 F5 F6 F7 F8 F9
             were: C9 AA F0 5C 38 60
       change to: C9 AA F0 5C 18 60

Tk 0, Sec F bytes: 65 66 67 68 69 6A
             were: 88 10 E7 A8 D0 15
       change to: 88 10 E7 A8 EA EA

Tk 0, Sec F bytes: 70 71 72 73
             were: C9 AF D0 0C
       change to: C9 AF EA EA

Tk 0, Sec F bytes: 79 7A 7B 7C
             were: C9 08 B0 03
       change to: C9 08 EA EA

Tk 1, Sec E bytes: CF D0 D1 D2 D3 D4
             were: F0 02 38 24 18 68
       change to: F0 02 18 24 18 68
```

```
1000 REM GATO 1.3 CONTROLLER
1010 TK = 0 :ST = 0 :LT = 35 :CD = WR :FAST = 0
1020 POKE 47497 ,24 : POKE 47498 ,96 :T1 = TK :
     GOSUB 490 : RESTORE
1030 IF TK / 2 = INT (TK / 2 ) THEN READ A1 ,A2 ,A3
1040 GOSUB 190
1050 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1050
1060 GOSUB 310 : GOSUB 230 : GOSUB 490 :TK = T1
     :ST = 0
1070 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1070
1080 ST = 0 :TK = TK + 1
1090 IF TK < LT THEN 1020
1095 POKE 47497 ,208 : POKE 47498 ,183
1100 HOME : PRINT : PRINT "DONE^ WITH^ COPY" :
     END
5000 REM DATA FOR GATO VERS 1.3
5010 DATA 212 ,170 ,150
```

input

```
5020 DATA 212 ,170 ,150
5030 DATA 8^ CHANGES
5040 DATA 1 ,14 ,209 ,24
5050 DATA 0 ,14 ,248 ,24
5060 DATA 0 ,15 ,105 ,234 ,0 ,15 ,106 ,234
5070 DATA 0 ,15 ,114 ,234 ,0 ,15 ,115 ,234
5080 DATA 0 ,15 ,123 ,234 ,0 ,15 ,124 ,234
```

Ann Onymous
San Luis, CA

More Bard's Tale

I would like to pass on the following information. First, I would like to say that the softkey for King's Quest (COMPUTIST No. 30, pg. 7) did not work [for me]. I found the "20 00 FF" four times at different locations. Changing these all to "EA EA EA" did not work for my version. There does seem to be some important code at $1600, although I cannot find how the program gets there or returns.

As for the Bard's Tale (COMPUTIST No. 30, pg. 11) the following sector edits work:

```
trk $01, sct $0C, bytes $00-$02
  from:4C 69 05 to:18 60 DD
trk $01, sct $0F, bytes $00-$02
  from:4C 69 A0 to:18 60 DD
```

As for Karate Champ (COMPUTIST No. 31, pg. 9) the following sector edits will work. They eliminate the call to the bit insertion routine and the jump to the code that clears memory and boots the disk.

```
trk $00, sct $03, bytes $BE
  from:20 00 BF 90 03 4C 93 BE
   to: EA EA EA EA EA EA EA EA
```

Brian A Troha
Stoughton, WI

Enhancing the Ultima IV Editor

The Ultima IV Character Editor by Danny Pollak in COMPUTIST No. 30 was a godsend to my characters, who were having a rough time at doing anything. But the main thing that the Editor lacked was the ability to change the virtues at will. I took out a sector editor and began changing bytes here and there and finally found where they were being kept track of. The virtues are kept as numbers ranging from 00 (being a partial Avatar) to 99 (nearly becoming a partial Avatar.) Checking where the bytes laid in memory after being loaded and adding on a little part to the Editor, finally made it out to what it was supposed to be, a complete editor.

```
345 PRINT "0)VIRTUES"
363 IF A$ = "0" THEN GOTO 2660
2660 CA = 2304 : FOR X = 0 TO 7 :VI(X) = FN B1(X
     ) : NEXT
2670 HOME : HTAB 16 : PRINT "VIRTUES" : PRINT
2680 FOR X = 7 TO 0 STEP - 1 : PRINT R$(X) : NEXT
2681 VTAB 3
2682 FOR X = 0 TO 7 : HTAB 15 : PRINT CHR$ (48 *
     (VI(X) < 10) )VI(X) : NEXT
2690 X = 0
2700 VTAB 3 + X : HTAB 15 :A1$ = "0" :A2$ = "9" :MAX
     = 2 : GOSUB 2390 : IF A$ = CH$ AND X = 0 THEN
     2700
2710 IF A$ = CH$ THEN X = X - 1 : GOTO 2700
2720 IF B$ = "" THEN B$ = STR$ (VI(X) )
2730 VI(X) = VAL (B$) : HTAB 15 : PRINT CHR$ (48
     * (VI(X) < 10) )VI(X) :X = X + 1 : IF X <
     8 THEN 2700
2740 GOSUB 2500 : IF A$ = "N" THEN 2690
2750 FOR X = 0 TO 7 : POKE (CA + X) , INT (VI(X)
     / 10) * 16 + (VI(X) - INT (VI(X) / 10
     ) * 10) : NEXT : GOTO 270
```

Tim Scott
Fargo, ND

bugs

COMPUTIST No. 31

Softkey for Time Zone:

The Super IOB controller will not function correctly as printed. Insert a RESTORE command at the beginning of line 1020 to fix the situation.

COMPUTIST No. 35

Softkey for The Perfect Score:

The procedure as printed works only for the first eleven sides of the program. To copy side two of disk F, do the following:

1) Load COPYA as in step one of the article and make the following patches in addition to those printed in COMPUTIST No. 35.

302:12 N 35F:12

2) After copying the disk, make the following sector edits:

Track $00, Sector $05, bytes $39 - $3B
From: BD 8C C0
To: 4C 81 02

Track $00, Sector $05, bytes $81 - $85
From: D0 10 88 10 F4
To: EA EA EA EA EA

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.

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
Cranston Manor Sierra On-Line
Snoggle Broderbund
ABM Muse
Mychess II Datamost
Story Tree Scholastic
Agent U.S.A. Scholastic
Handicapping System Sports Judge
Echo Plus Agranat Systmes
Great Cross Country Road Race Activision
Odin Odesta
Mabel's Mansion Datamost
Brain Bank The Obsevatory
Under Fire Avalon Hill
Crimson Crown Penguin
Crypt of Media Sir Tech
EDD IV Utilico Microware
The Works First Star Software
Cross Clues Science Research
Peeping Tom Microlab
Jigsaw Microfun
Miner 2049er II Microfun

readers' softkey & copy exchange

Steve McLendon's softkey for...

Critical Reading

Josten's Learning Systems, Inc.
800 E. Business Center Dr.
Mount Prospect, IL 60056

Requirements:
Apple][Plus, //e
Means of entering the Monitor
DeMuffin Plus
Any of the Borg-Warner Critical Reading Series disks
Blank formatted disk
Sector editor
Disk searcher (optional)

Sorry, Super IOB, but I am not able to make you deprotect this one. Only about half of each track on any of these Borg-Warner disks has valid data on it and the other half is meant to throw off the bit copiers and just about anything else as well.

To give an example, from disk "C", here is the sector map for tracks 8-A:

Sector	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Tk 8	x	x	x	x	x	x	x	x	x	x	x	d	d	d	d	d
Tk 9	x	d	d	d	d	d	d	d	d	d	d	d	d	d	d	x
Tk A	d	d	x	d	d	d	d	d	d	d	x	d	d	d	d	d

An "x" indicates a sector with invalid data which even BW's own RWTS cannot read; "d" indicates that sector contains valid data. Imagine trying to write a Super IOB controller to handle this type of scheme. Well, I did, but then I realized I would probably have to verify the sector maps on **every** disk in the series. If even one sector was different, the controller would have to be modified.

So DeMuffin Plus, which has been all but forgotten by many folks, is the one and only tool to use here. We will have to use the BW RWTS, but we would have had to do that even with Super IOB.

1) Load DeMuffin Plus into a safe area of memory.

BLOAD DEMUFFIN PLUS,A$6000

2) Boot the Borg-Warner disk and, just as the serial number is displayed at the bottom, reset into the monitor.

3) Look at the code from $1500-0F and record these 16 hex numbers.

4) Move DeMuffin Plus down to its normal location and run it

803<6000.8103M
803G

5) Copy all files onto your blank, initialized disk.

6) Now boot normal DOS, get into the monitor (**CALL -151**), and type in the code at $1500 which you recorded in step 3. For disk "C", these bytes should be:

1500:AC D5 AD BE B7 B6 BC F2
1508:F3 DA AD DA AD E6 9D D5

7) Save this little piece of code to disk.

BSAVE BTCD,A$1500,L$10

8) Type **NEW** and enter the following BASIC program.

```
10 D$=CHR$(4)
20 PRINT D$'BLOAD BTCD"
30 PRINT D$'RUN F[CTRL-S]3TUTOR"

SAVE HELLO
```

9) Now with your sector editor make the following mods.

Track	Sector	Byte	From	To
3	8	$AD	$8C	$B1
3	8	$AE	$34	$3A
3	8	$AF	$34	$B2

These three bytes should be found on the indicated sector. However, if you are using anything but virgin DOS 3.3 they could be anywhere, in which case you will have to do a disk search to find their location. Write these changes back to disk.

You should now have a COPYAable Borg-Warner Critical Reading Series disk. This technique will work on all disks in the entire series. David Ward asked for help (Input, COMPUTIST No. 16), and here it is. I am appalled that Borg-Warner charges an educational institution close to $1000 for a set of these disks and refuses to provide backups. Mr. Ward, here is your backup, compliments of COMPUTIST.

Daniel J. Elliot's softkey for...

Troll's Tale

Sierra On-Line
36575 Mudge Ranch Rd.
Course Gold, CA 93614

Requirements:
Apple][or better
Super IOB v1.5
1 blank disk side

Come along and let me tell you a tale, a Troll's Tale. This is the introduction to Troll's Tale, a cute, first adventure for children ages 6 through 10 years of age. Sierra On-Line has made creative use of The Graphics Magician from Penguin Software in this graphic text adventure. As usual, they have also made creative use of copy protection on this release as well.

The Protection

The entire protection scheme for this disk consists of altered address prologue and epilogue marks on different tracks. The data prologues and epilogues however are standard. For tracks $00 - $02, the address prologue is the standard D5 AA 96 but the address epilogue has been changed from DE AA TO ED AA. For tracks $03 - $22, the address prologue is DB AA 96, while the address epilogue is the standard DE AA. I find the sector editor of Copy][Plus 5.1 ideal for determining these changes. Now it is only necessary to write a controller which will poke the proper bytes in at the proper time during the read/write cycle.

The Procedure

All that is required to copy Troll's Tale is to install the controller at the end of this article into Super IOB and RUN the resulting program.

A faster DOS such as Diversi DOS or Pronto DOS is als a nice addition to this program, but it is still necessary to copy tracks $00 - $02 from the original disk because for some reason, the hello program will mess up if only tracks $3 - $22 are copied, then a fast DOS added. If the program has it's original DOS or standard DOS 3.3, it is possible to Reset into Applesoft anytime and CATALOG the disk.

Also, for reasons unknown to me, if the DOS is Diversi DOS, attempting to reset into Applesoft at any point past the title page will drop into the monitor and lock up the keyboard. Before the title page, their Reset works fine with Diversi DOS.

Enjoy the tale.

controller

```
1000 REM TROLLS TALE
1010 TK = 0 :ST = 0 :LT = 35 :CD = WR
1020 T1 = TK : GOSUB 490
1025 POKE 47505 ,237 - 15 * (TK > 2 ) : POKE
     47445 ,213 + 6 * (TK > 2 )
1030 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1030
1040 IF BF THEN 1060
1050 ST = 0 :TK = TK + 1 : IF TK < LT THEN 1025
1060 GOSUB 230 : GOSUB 490 :TK = T1 :ST = 0
1070 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1070
1080 ST = 0 :TK = TK + 1 : IF BF = 0 AND TK < LT THEN
     1070
1090 IF TK < LT THEN 1020
```

```
1100 HOME : PRINT "DONE^ WITH^ COPY" : END
```

controller checksums

1000	- $356B	1050	- $79AD
1010	- $3266	1060	- $DC82
1020	- $C11A	1070	- $D487
1025	- $8831	1080	- $2EDA
1030	- $9E30	1090	- $8140
1040	- $1CA1	1100	- $A73B

Darry Distreou's softkey for...

Robot War

Muse
347 N. Charles St.
Baltimore, MD 21201

Requirements:
Apple][or better
A sector editor
FID from the system master
Super IOB v1.5
4 blank disk sides

Robot War is an educational strategy game with its own language and compiler. After programming your robot, you put it into a battlefield where it must compete in mortal combat with other similarly programmed robots.

As I expected, my Robot War disk was protected. A little snooping revealed that it uses a modified DOS based on DOS 3.2. Through the use of the swap controller and some DOS modifications, we can have a standard DOS 3.3 version of Robot War.

The Softkey

1) Boot the original Robot War disk and when the main menu comes up on the screen, choose option 6 to exit to Applesoft BASIC.

2) Now we will move the entire Robot War DOS to a safe location.

CALL -151
2600<9600.BFFFM

3) You must now find the volume number of your Robot War disk (mine was 001).

CATALOG

Write down the volume number of the disk.

4) Boot a 48K slave disk (preferably with no hello program) and format a disk we will call disk A.

INIT DISKA

5) Save the Robot War DOS and Robot War RWTS as two separate files on disk A.

BSAVE RWTS.ROBOTWAR
,A$4800,L$800
BSAVE ROBOTWAR.DOS
,A$2600,L$2200

6) Patch DOS and format a disk that will be called disk B with the volume number you determined in step 3.

POKE -19523,12
INIT HELLO,V1

7) Install the Robot War controller at the end of this article into Super IOB and copy the original Robot War disk to a new disk labeled disk C.

8) Now, transfer all the files from disk C to disk B by using FID.

9) After we have transferred all the files, put disk A into the drive and load the file ROBOTWAR.DOS.

BLOAD ROBOTWAR.DOS

10) Type in the following bytes and then press Reset. This has the effect of disabling DOS.

CALL -151
3F2:03 E0 45

11) Now move Robot War's DOS back to its original place and activate it.

CALL -151
9600<2600.47FFM
9D7EG

12) Put disk B into the drive execute the boot file.

RUN HELLO

13) When the main menu comes up, choose option 5 to initialize disk D.

14) When the process is done, exit the Robot War and get out a sector editor to copy track 0, sector 0 of disk B to track 0, sector 0 of disk D.

15) Now, install the CopyDOS controller at the end of this article into Super IOB and use it to copy tracks 0 through 2 from disk D to disk B.

You now have a COPYAable version of Robot War on disk B.

RobotWar controller

```
1000 REM ROBOT WAR
1010 TK = 0 :ST = 0 :LT = 35 :CD = WR :DOS = 13
1020 T1 = TK : GOSUB 490 : GOSUB 360 : ONERR GOTO
     550
1030 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1030
1040 IF BF THEN 1060
1050 ST = 0 :TK = TK + 1 : IF TK < LT THEN 1030
1060 GOSUB 490 :TK = T1 :ST = 0 : GOSUB 360
1070 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1070
1080 ST = 0 :TK = TK + 1 : IF BF = 0 AND TK < LT THEN
     1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "CONTROLLER^ DONE" : END
10010 IF PEEK (6400 ) < > 162 THEN PRINT CHR$ (4
     ) "BLOAD^ RWTS.ROBOTWAR,A$1900"
```

controller checksums

1000	- $356B	1060	- $90D6
1010	- $23F9	1070	- $98D3
1020	- $3D3E	1080	- $7A3E
1030	- $2B3F	1090	- $5D64
1040	- $D354	1100	- $68C4
1050	- $3735	10010	- $67B5

CopyDOS controller

```
1000 REM COPYDOS
1010 TK = 0 :LT = 3 :ST = 15 :LS = 15 :CD = WR :FAST
     = 1
1020 GOSUB 490 : GOSUB 610
1030 GOSUB 490 : GOSUB 610 : IF PEEK (TRK ) = LT
     THEN 1050
1040 TK = PEEK (TRK ) :ST = PEEK (SCT ) : GOTO 1020
1050 HOME : PRINT "DOS^ COPIED" : END
```

controller checksums

1000	- $356B	1030	- $5F3F
1010	- $FF63	1040	- $321D
1020	- $1371	1050	- $6239

Kevin Sartorelli's softkey for...

General Manager

Sierra On-Line

Requirements:
General Manager 2.0Y
Blank disks
COPYA

This versatile data base uses a nibble count routine on track 0 as its protection. The nibble count routine is encoded and a checksum of it is generated to further confuse the issue. The following method will remove both the nibble count and the memory check.

1) Copy all disks with COPYA.

2) Insert the copy of the Master Program disk.

3) Go into the monitor with **CALL -151**.

4) Load in the file containing the encoded nibble count routine.

```
BLOAD SORT INTERFACE OBJ
  ,A$7000
```

5) Disable the nibble count.

```
7054:BE BF
```

6) Save the patched file back to disk.

```
UNLOCK SORT INTERFACE OBJ
BSAVE SORT INTERFACE OBJ
  ,A$7000,L$100
```

7) Load in the file containing the memory check.

```
BLOAD GENERAL MANAGER
```

8) Disable the memory check.

```
641C:EA
```

9) Save the patched file.

```
UNLOCK GENERAL MANAGER
BSAVE GENERAL MANAGER
```

Your copy of The General Manager is now ready to use.

Kevin Sartorelli's softkey for...

Plasmania

Sirius Software

Requirements:
A DOS 3.3 slave disk with no HELLO
A RAM card to run the game

This game by Sirius is a bit of fun to play but like most games it is protected. As it appeared to be a single load game that only required the disk when the game had ended, I felt it could be made into binary files.

To do this I did a boot code trace and found that what was read in between games was the code to do the talking that accompanies the title page, and the title page. Below is the method I used to break Plasmania down to three files. The first file is the main game, the second the 'talk' code, and the third a small file to load the other two. The game when cracked like this requires a RAM card to work as the 'talk' code is stored on the RAM card until needed and then moved from there instead of being read in from the disk. The boot code trace has to be done twice as the game takes up most of memory and some of this is overwritten when a slave disk is booted.

1) Go into the monitor

```
CALL -151
```

2) Move the first stage boot from ROM to RAM.

```
6600<C600.C6FFM
```

3) Make a patch to load in the game and enter the monitor after loading.

```
2F0:A9 59 8D F5 04 A9 FF 8D
2F8:F6 04 4C 00 04
66F8:A9 F0 8D C9 08 A9 02 8D
6700:CA 08 4C 01 08
```

4) Start the boot.

```
6600G
```

5) Move the memory that would be destroyed by the upcoming boot.

```
9100<800.900M
```

6) Boot a slave disk with no HELLO and return to the monitor.

```
6[CTRL]P
CALL -151
```

7) Restore the moved code and save the main game to the disk.

```
800<9100.91FFM
BSAVE PLASMA1,A$800,L$8800
```

8) Repeat steps 2, 3, and 4 to reload the program.

9) Move the talk code away from DOS's area.

```
2000<9000.BFFFM
```

10) Boot the slave disk again and return to the monitor.

```
6[CTRL]P
CALL -151
```

11) Patch the talk code so it will work, and add code to move itself out of the RAM card at run time.

```
4FF1:00 BF 60
4F00:AD 83 C0 AD 83 C0 A9 00
4F08:85 04 85 06 A9 E0 85 05
4F10:A9 40 85 07 A0 00 B1 04
4F18:91 06 C8 D0 F9 E6 07 E6
4F20:05 D0 F3 AD 82 C0 60
```

12) Save the talk code to the disk.

```
BSAVE PLASMA0,A$2000,L$3000
```

13) The following is code to load in the main program, load the talk code into the RAM card, and enter the game at $6000.

```
300:AD 50 C0 AD 57 C0 AD 54
308:C0 AD 52 C0 AD 81 C0 AD
310:81 C0 A0 00 B9 76 03 F0
318:06 20 ED FD C8 D0 F5 AD
320:82 C0 A0 00 B9 8D 03 F0
328:06 20 ED FD C8 D0 F5 EE
330:F4 03 AD 83 C0 AD 83 C0
338:A9 90 85 07 A9 D0 85 05
340:A0 00 84 04 84 06 B1 04
348:91 06 C8 D0 F9 E6 07 E6
350:05 D0 F3 A9 E0 85 07 A9
358:40 85 05 A0 00 84 04 84
360:06 B1 04 91 06 C8 D0 F9
368:E6 07 E6 05 A5 07 D0 F1
370:AD 82 C0 4C 00 60 8D 84
378:C2 CC CF C1 C4 D0 CC C1
380:D3 CD C1 B0 AC C1 A4 C4
388:B0 B0 B0 8D 00 8D 84 C2
390:CC CF C1 C4 D0 CC C1 D3
398:CD C1 B1 8D 00
```

14) Save the new loader.

```
BSAVE PLASMANIA,A$300,L$9D
```

Now to run Plasmania you type BRUN PLASMANIA and away it goes.

Larry Rando's softkey for...

Telarium Software

Telarium Corp.
1 Kendall Square
Cambridge, MA 02139

Requirements:
Whole disk copier that can ignore errors
Perry Mason
Farenheit 451
Rendezvous with Rama
Nine Princes in Amber

Telarium's protection schemes are basically the same (at least in the fact that they usually reside in a Binary file called IO). Changing a standard nibble count is all that it takes to defeat these schemes.

Perry Mason& Nine Princes in Amber

1) Copy all four sides with any whole disk copier that can ignore errors.

2) Boot DOS 3.3 and load IO from disk 1.

```
PR#6
BLOAD IO
```

3) Enter the monitor and defeat this file's nibble count.

```
CALL -151
1CC1:A9 00 EA
```

4) Save the modified file.

```
BSAVE IO,A$A00,L$1512
```

That's all!

readers' softkey & copy exchange

Rendezvous with Rama

1) Copy all four sides with your whole disk copier.

2) Boot DOS 3.3 and load the offending file from disk 1.

PR#6
BLOAD IO

3) Enter the monitor and defeat the nibble count.

CALL -151
1BF5:20 29 1C

4) Save this defeated file.

BSAVE IO,A$A00,L$1512

Fahrenheit 451

1) Copy all four sides with your whole disk copier.

2) Boot DOS 3.3 and load the protection file.

PR #6
BLOAD IO

3) Enter the monitor and correct this file.

CALL -151
1C24:EA EA EA

4) Save this version of IO.

BSAVE IO,A$800,L$1516

I hope these procedures help you in your quest for deprotection.

Jeff Lucia's APT for...

Championship Lode Runner

Requirements:
A sector editor
One blank disk
A good bit copier

When playing Championship Lode Runner, have you ever wished you could skip to any level, have any amount of players, see the special password for the certificate, revive games that have been deleted or modify the high scores? I know that I have. This is why I have developed the following APT for all of the above.

1) First, Copy tracks 3-8 onto your blank disk.

2) Have a little fun by playing Championship Lode Runner for a while, then save the game with any name you like (write the name down so you remember it).

3) Run your sector editor and read track $0C sector $0D

4) Here's the hard part. In the text portion find the name of your saved game. Now go to the first letter of the name and then go forward eight bytes.

5) This byte will be the real level number. The next byte will be the real level number minus 1. The third byte is how many men you have. The other five bytes will be your score (in a special order).

6) Now that you know what each byte is, modify them in hex, to the desired values. If you want to revive a deleted game you must look for the name of that game. There will be an inverse "@" for the first letter of that game. Change it to a normal letter. Then change the amount of men left.

7) Once you are done write the sector back to the disk.

8) If you want to change the high scores use your sector editor and read tracks $0C sector $0F then find the name of the high score you desire to modify and move forward eight bytes. The high scores are stored the same way as games.

An Example

Here is a little example of what I was saying in step 4-5. Let us say that you saved a game named "FOOP" (Good name) now we use Copy II+ 4.3's (or any version) sector editor. Here is what track $0C sector $0D will look like the following:

```
00- C6 CF CF D0 A0 A0 A0 A0  FOOP
08- 01 00 05 00 00 00 00 00  A@E@@@@@
```

Now let's look at this. Look at the byte eight bytes forward of the letter "F" in the word "FOOP". Notice the hex value "01" this is your real level. Also notice that next byte is a "00" and that the third byte is an "05". This means that "FOOP" has 5 men and is on level 1 with no score. Now you want to go to level 50 with 255 men. So, change the first byte to a $32 the second byte to a $31 and the third byte to an $FF. You MUST change the second byte to the level minus one (1) otherwise the program knows you're trying to cheat it and starts you at level one.

Here is also one quick example of undeleting a deleted game. The name of my game was called, "Level 42."

```
10- 00 C5 D6 C5 CC A0 B4 B2  @EVEL 42
18- 26 25 BE 01 59 47 00 00  &%>AYG@@
```

Just change the first byte of the name to any letter or number and you will have a game with a high score of 1,594,470.

I hope you have a lot of fun with this!

Daniel J. Elliot's softkey for...

Kidwriter v1.0

Requirements:
Apple][, //e or //c
Super IOB v1.5
1 blank disk side

Kidwriter is a word processor for children, ages 6-10, which allows them to create their own story boards. This is a very neat little program for developing a child's interest in the computer. While I do not plan on using the program myself, a backup would be very convenient for any program used by small children. Unfortunately, my luck held out and the version of Kidwriter softkeyed by Mike Stafford in COMPUTIST No. 20 had a different protection scheme than mine. This left me no alternative but to develop a new softkey.

The Protection

During the boot, an Applesoft prompt appears at the bottom of the screen indicating a somewhat normal DOS. Next, using my nibble editor, I examined the tracks and sectors for altered address and data prologues and/or epilogues. The data field was normal but the address field was another matter.

First of all, the address headers alternated between the usual D5 AA 96 and the not so normal D4 AA 96. This is much like the protection used on several Penguin releases. The address field trailers proved to be AF A0. I quickly made a Super IOB controller and tried to copy the disk.

The controller got some "Drive Errors" so I examined the sectors more closely and noticed that the address field trailers changed on different tracks. I therefore revised the controller to ignore the data field trailers and presto!, a deprotected Kidwriter.

Step by Step

1) Install the controller at the end of this article into Super IOB and copy the disk.

You now have a COPYAable Kidwriter. If you also have a kid, you'll need it.

controller

```
1000 REM KIDWRITER
1010 TK = 0 :ST = 0 :LT = 35 :CD = WR
1020 T1 = TK : GOSUB 490 : ONERR GOTO 550
1022 POKE 47405 ,24 : POKE 47406 ,96 : POKE 47497
     ,24 : POKE 47498 ,96
1025 POKE 47445 ,212 + (TK / 2 = INT (TK / 2 ) )
```

readers' softkey & copy exchange

```
1030 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1030
1040 IF BF THEN 1060
1050 ST = 0 :TK = TK + 1 : IF TK < LT THEN 1025
1060 GOSUB 490 :TK = T1 :ST = 0 : POKE 47445 ,213
1070 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST <
     DOS THEN 1070
1080 ST = 0 :TK = TK + 1 : IF BF = 0 AND TK < LT THEN
     1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "DONE^ WITH^ COPY" : END
```

controller checksums

```
1000  - $356B     1050  - $CD87
1010  - $3266     1060  - $CB56
1020  - $5528     1070  - $C353
1022  - $5E97     1080  - $FA65
1025  - $BCB3     1090  - $6A3C
1030  - $AAB2     1100  - $9BD8
1040  - $FC6B
```

Glen Tatum's softkey for...

Color Me

Mindscape

Requirements:
Super IOB or COPYA
A disk scanner
A sector editor

Color Me is a new disk from Mindscape, it is a double high-res color book-type program. My kids love to use it, and it uses the color capabilities of the new Imagewriter II color printer. I don't know if anyone else has had trouble copying the disk or not, but my copies always just kept rebooting.

Using a sector editor with search capabilities (I use Copy II+) I tried searching the copy for the hex commands 4C 00 C6 (JMP $C600,or reboot disk). Looking around in the same area I saw a JMP 1706, if I booted the copy disk and then reset into the monitor and tried a 1706G, the disk light came on and it started reading more data but then stopped. Obviously, it needs to be in some loop to continue. If we go back with a sector editor and reverse the two commands so it loops at 1706 instead of C600 then the program loads and runs fine. So, here is a step by step for Color Me:

1a) If you're using Super IOB, then use the controller at the end of this article to copy Color me and go to step 2.

1b) If you are using COPYA, then enter the monitor and tell DOS to ignore the ending marks.

```
CALL -151
B988:18 60
B925:18 60
3D0G
RUN COPYA
```

2) Search your disk for the sequence 4C 00 C6, mine was at byte 93 of Track $0 Sector $8. A bit before this sequence, you should see a 4C 06 17 (mine was at 8C).

3) Get out your sector editor and make the following changes to these sequences.

Byte	was	now
8C	4C	4C
8D	06	00
8E	17	C6
93	4C	4C
94	00	06
95	C6	17

Write the sector back out to the disk, and you have it finished. The same Super IOB controller or modified COPYA can be used to copy all of the picture disks as well.

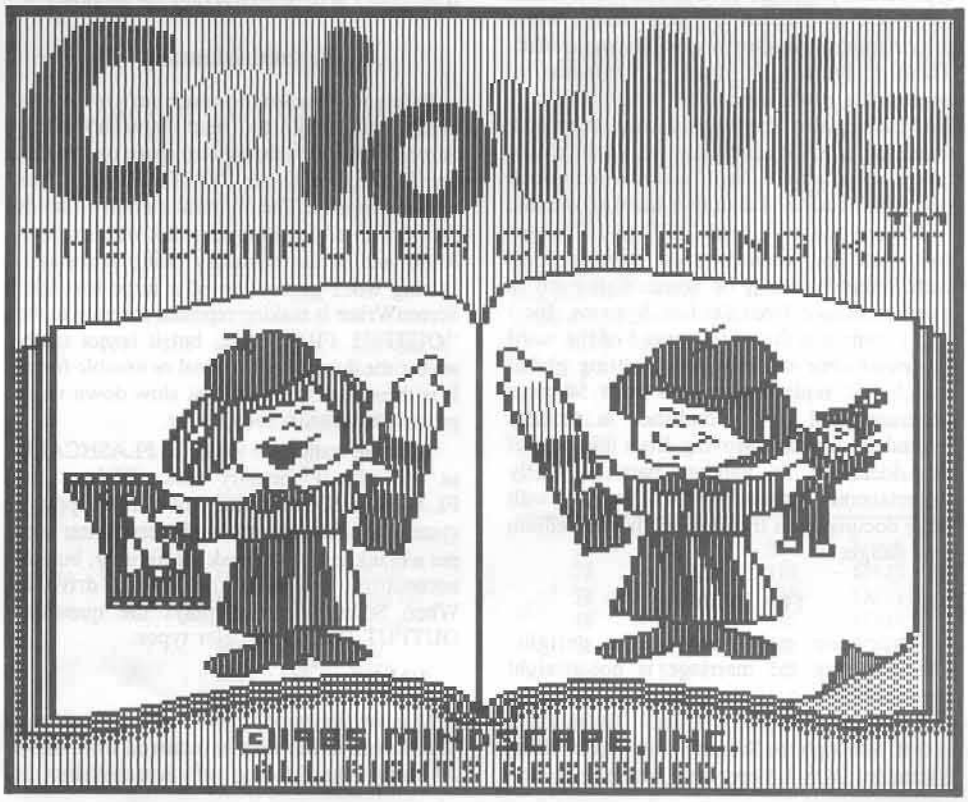
controller

```
1000 REM COLOR ME
1010 TK = 0 :LT = 35 :ST = 15 :LS = 15 :CD = WR :FAST
     = 1
1015 POKE 47496 ,24 : POKE 47497 ,96 : POKE 47397
     ,24 : POKE 47398 ,96
1020 GOSUB 490 : GOSUB 610
1030 GOSUB 490 : GOSUB 610 : IF PEEK (TRK ) = LT
     THEN 1050
1040 TK = PEEK (TRK ) :ST = PEEK (SCT ) : GOTO 1020
1050 HOME : PRINT "COPYDONE" : END
```

controller checksums

```
1000  - $356B     1030  - $76FF
1010  - $2544     1040  - $BA80
1015  - $62D1     1050  - $2FBC
1020  - $A16C
```

To stop an EXEC file, Reset into the monitor and type:
AAB3:00
Reconnect DOS and type:
CLOSE
to close the file.



Screenwriter meets Flashcard

by Herbert Alfred Mayer

Most word processing programs become disk intensive when text files of more than a few typewritten pages are edited. The result is that the typist is plagued by long delays while information is transferred to and from the disk. These delays can be virtually eliminated by appropriate application of a "RAM disk". A RAM disk, also known as a solid state disk emulator, is RAM configured to imitate a disk drive. This article shall describe one such marriage of a RAM disk to a word processor.

The use of the FLASHCARD (a 147K byte solid state disk emulator from Synetix, Inc.) with ScreenWriter][or ScreenWriter //e (a word processor from On-line Systems, Inc.) vastly improves the editing speed of the word processor. For instance, a text-string global search and replace operation on a 50 page document will be accomplished in seconds instead of minutes. Moving from one part of the document to another part is nearly instantaneous. You will find that working with large documents is transformed from a tedium to a delight.

The Problem

While the end result is a delight, consummating the marriage is not straight forward. On-line "does not support any of the disk emulators for ScreenWriter." The problem is that, although the ScreenWriter disk contains a standard DOS 3.3 operating system including a standard RWTS routine, ScreenWriter uses its own RWTS routine for text file access. Fortunately, ScreenWriter will operate properly when connected to a standard DOS 3.3 or the FLASHCARD alternate RWTS routine. This article will describe how to make a FLASHCARD version of either ScreenWriter][version 2.0 or ScreenWriter //e version 2.2.

Virtual Memory

Before we proceed, however, let's review the preferred way to use ScreenWriter. ScreenWriter uses the unused space on the disk that is assigned to the "OUTPUT FILE" as virtual memory. The "virtual memory" holds the portion of the document that will not fit in RAM and is not currently being processed. During word processing of a large text file, ScreenWriter is making repeated accesses to the "OUTPUT FILE" disk, but it is not really saving the document in a final or useable form. It is these disk accesses that slow down word processing with ScreenWriter.

What we want to do is use the FLASHCARD as the virtual memory disk. Without the FLASHCARD, assuming a two drive Apple][system, the best way to use ScreenWriter is to put a blank initialized disk (preferably, but not necessarily, with DOS deleted) in drive 2. When ScreenWriter displays the question: OUTPUT FILE?, the user types:

VM,D2

(VM is the file name I give to virtual memory.) The ScreenWriter disk is removed from drive #1 after the loading of ScreenWriter is complete. The INPUT FILE will be from a file on a document disk subsequently placed in drive #1. Text is saved to the document disk using the command:

sNAME,d1

Now let's add a FLASHCARD to the system. We will use the FLASHCARD in place of the blank initialized disk in drive 2. When ScreenWriter displays the question: OUTPUT FILE?, the user will type

VM,S5,D1

(assuming the FLASHCARD is in slot 5). Text will be saved to the document disk in drive 1 using the command:

sNAME,s6,d1

(The D1 or d1 suffix will not be necessary if drive 2 is not accessed.)

FAST SPOOL Printer spooling requires the use of some memory device to buffer the data going from the computer to the printer. ScreenWriter permits using blank disk space for this buffer memory. I use spooling mainly to gain, from within ScreenWriter, boldfacing and underlining capabilities with my EPSON printer. It also permits editing one document while printing another. The latter is best implemented with a 294K (2 drive) version of the FLASHCARD. In general, it is advantageous to use the FLASHCARD for spooling, as it will speed spooling and save wear and tear on a disk drive. The required modification will only be described for the EPSON SPOOLER, as it would be too redundant to describe the modification for all

seven spooler drivers provided on the ScreenWriter disk. The same type of modification, however, can be made for any of the other spoolers. Set up the spooler as per the instructions in the ScreenWriter manual and then make the modifications described below.

Modifications

A FLASHCARD version of ScreenWriter may be prepared as follows:

1) Notch a blank disk so that both sides may be used. (Alternatively, use two blank disks.) This will be our ScreenWriter/Flashcard disk.

2) Use the COPYA utility on your DOS 3.3 SYSTEM MASTER to copy the ScreenWriter disk to the front side of the ScreenWriter/Flashcard disk. If this is ScreenWriter][version 2.0, delete the TUTORIAL file to make room for the SSD programs we must add to the disk.

3) Initialize the back side of the ScreenWriter/Flashcard disk. Be sure not to copy any files to the back side. If you have a utility that will permit deleting DOS from the back side, use it. If you wish, you may use this back side once with ScreenWriter as the virtual memory disk in order to register the VM file on it. Then when ScreenWriter displays the question: OUTPUT FILE?, the user needs only type:

1,S5,D1

4) Listing #1 presents a modified version of the AUTOCOPY program supplied by Synetix on their DOS 3.3 DRIVERS disk. Enter the program as shown and, using the name: SSD AUTOSTART, save it to the front side of the ScreenWriter/Flashcard disk.

5) Use the FID utility on your DOS 3.3 SYSTEMS MASTER disk to transfer the SSD DRIVER/ALT and COPY programs supplied by Synetix on their DOS 3.3 DRIVERS disk to the front of your ScreenWriter/Flashcard disk.

6) The remaining tasks will require a good sector editor such as BAG OF TRICKS from Quality Software. Make the disk modifications shown in Table #1.

Customizing

When you customize your new ScreenWriter/Flashcard disk, be sure to make the following slot and drive assignments:

Master Disk-	Slot=6, Drive=1
Text Disk-	Slot=6, Drive=1
Spooler Disk-	Slot=5, Drive=1

The last item assumes you wish to set up spooling and that the FLASHCARD is in Slot 5. If you have a 294K byte (2 drive) FLASHCARD, use Drive 2 for the spooler so that you can edit and print simultaneously.

Operation

Your ScreenWriter/Flashcard disk operates as follows:

Upon booting the front side of the disk, the SSD AUTOSTART program is loaded and run. The SSD AUTOSTART program, BRUNs the SSD DRIVER/ALT program, which loads the SSD RWTS routine into the INIT area of DOS and installs a jump from the DOS RWTS routine to the SSD RWTS routine. The SSD AUTOSTART program then BLOADs the COPY program into memory and requests the user to reverse the disk in drive 1. The back side of the ScreenWriter/Flashcard disk is then copied into the flashcard. This initializes the flashcard. The user is then again requested to reverse the disk in drive 1.

The SSD AUTOSTART program then BRUNs the START program, which is the ScreenWriter program that initiates the loading of the various ScreenWriter files. These ScreenWriter files have been modified with jumps installed from their special RWTS routines to the SSD RWTS routine. The only ScreenWriter feature that has been lost is the ability to initialize a data disk; data disks must be initialized beforehand with standard (unmodified) DOS booted.

Table 1

Track	Sector	Byte	From	To	Notes
For either ScreenWriter version:					
$00	$0D	$42	$34	$06	RUN command
$01	$09	$75	"START"	"SSD AUTOSTART"	
For ScreenWriter][version 2.0-					
$03	$0B	$94	$20	$EA	RAM
			$00	$EA	Editor:
			$0F	$EA	Enable
$13	$04	$4D	$20	$EA	NON-RAM
			$00	$EA	Editor:
			$60	$EA	Enable
$12	$0F	$0E	$A0	$A0	RAM
			$2C	$2C	Editor:
			$84	$A9	Connect
			$48	$74	SSD
			$A9	$4C	RWTS
			$74	$00	
			$85	$BD	
			$49	$00	
$05	$09	$E9	$A0	$A0	NON-RAM
			$07	$07	Editor:
			$84	$A9	Connect
			$48	$69	SSD
			$A9	$4C	RWTS
			$69	$00	
			$85	$BD	
			$49	$00	
$17	$0B	$EC	$A0	$A0	EPSON
			$01	$01	Spooler:
			$84	$A9	Connect
			$48	$91	SSD
			$A9	$4C	RWTS
			$91	$00	
			$85	$BD	
			$49	$00	
For ScreenWriter //e version 2.2-					
$0F	$07	$90	$20	$EA	RAM
			$00	$EA	Editor:
			$7F	$EA	Enable
$0E	$03	$49	$20	$EA	NON-RAM
			$00	$EA	Editor:
			$6E	$EA	Enable
$17	$03	$68	$A2	$4C	RAM
			$00	$00	Editor:
			$D0	$BD	Connect
			$A8	$00	SSD RWTS
$1A	$08	$4E	$A2	$4C	NON-RAM
			$00	$00	Editor:
			$D0	$BD	Connect
			$A8	$00	SSD RWTS
$08	$08	$08	$A2	$4C	//e
			$00	$00	Editor:
			$D0	$BD	Connect
			$A8	$00	SSD RWTS
$13	$08	$2F	$A2	$4C	EPSON
			$00	$00	Spooler:
			$D0	$BD	Connect
			$A8	$00	SSD RWTS

Program 1

```
10 TEXT : HOME : PRINT "SSD^ AUTOSTART"
20 PRINT CHR$ (4 ) "BRUN^ SSD^ DRIVER/ALT"
30 PRINT CHR$ (4 ) "BLOAD^ COPY"
40 A = PEEK (43634 ) + 256 * PEEK (43635 )
50 REM OS = ORIGINAL SLOT, OD = ORIGINAL DRIVE,
     CS = COPY SLOT, CD = COPY DRIVE, FMT= 0DON'T
     FORMAT, FMT= 1DO FORMAT
60 OS = 6 :OD = 122CS = 5 :CD = 1 :FMT = 0
70 PRINT "ABOUT^ TO^ COPY^ SLOT^ " OS "^ DRIVE^
     " OD : PRINT "^^^^^^^^^^^^ TO^ SLOT^
     " CS "^ DRIVE^ " CD : PRINT
80 PRINT "PLEASE^ REVERSE^ SCREENWRITER^
     DISK!" : PRINT : PRINT "AND^ THEN" : PRINT
     : INPUT "PRESS^ [RETURN]:^ " ;ZZ$
90 GOSUB 130
100 HOME : PRINT "SSD^ INITIALIZED" : PRINT :
     PRINT "PLEASE^ REVERSE^ SCREENWRITER^
     DISK!" : PRINT : PRINT "AND^ THEN" : PRINT
     : INPUT "PRESS^ [RETURN]:^ " ;ZZ$
110 PRINT CHR$ (4 ) "BRUN^ START"
120 END
130 REM COPY DISKS
140 B = PEEK (A + 3 ) + 256 * PEEK (A + 4 ) : POKE
     B ,OS : POKE B + 1 ,OD : POKE B + 2 ,CS : POKE
     B + 3 ,CD : POKE B + 4 ,255 : POKE B + 5 ,255
     * (FMT < > 0 )
150 CALL A
160 RETURN
```

checksums

```
10  - $E7C3     90  - $F0D5
20  - $56EE     100 - $D925
30  - $6444     110 - $A271
40  - $9F67     120 - $C544
50  - $7989     130 - $C6C5
60  - $ABF0     140 - $5A0E
70  - $A00B     150 - $2A12
80  - $CFC0     160 - $3BEE
```

Enhancing your Apple with the...

Bus

Note: COMPUTIST magazine or SoftKey Publishing will not be held responsible for any damages incurred while following this procedure.

by Clay Harrell and Sidney Fernstock

Some of you COMPUTIST readers may have been alive during the dark ages when all computers had "front panels" filled with switches, dials, and blinking lights (now relegated to B-grade science-fiction movies). From these marvelous control panels you could examine any memory location, change it, step through the program, find an error, and correct the code without having to exit the program you were running. Then came those infernal high-level languages, lower-cost computers, and (ugh) monitor programs. I'm proud, finally, to announce a major step backwards in computing - the Apple Bus Monitor.

Basically, this device "rides the bus" in the Apple and reveals where the CPU is and whats it's doing. In normal operation, it gives you an "average" reading of the value on the Apple's address bus (usually the program counter), and the contents of memory at that location. On many computers, this information would be severely deficient for any serious debugging purposes, but since all the I/O on the 6502 is through memory locations, you can use this device in an amazing number of informative and entertaining ways.

In addition, there is a slow-down feature which allows you to watch the CPU at very slow rates, or even single-step through a program to debug it. **A few cautions:** the Bus Monitor is absolutely useless for debugging BASIC programs, and requires a working knowledge of the 6502, the Apple, and of Assembly language to justify the effort required to build one. Be advised, too, that this project is strictly for the hardware builders and those intrepid souls who love the challenge of something new. Further, the Bus Monitor will not allow some of the more sophisticated functions of a good front panel such as alter, trap, or break at a specific location or value, and it doesn't work 100% correctly on Apple][s manufactured before 1978. Finally, the Bus Monitor won't work when the Apple is under the control of a plug-in co-processor card such as a Z-80 Softcard, "the Mill" 6809 card, or one of the fancy new 68000 or 8088 cards.

The Bus Monitor is built up on a "kludge" card that plugs into a peripheral slot connector in the Apple. If you like to use wire-wrap construction, stick to "two-level" sockets or resign yourself to losing two slots to this card. When I built mine, I tried for a long time to figure some way of mounting the LED displays on the card to avoid cabling problems, but was unsuccessful. The result is that a cable must run from your Apple to a box which houses the controls and displays. This is a minor problem if you have an Apple //e with its "helpful" teeny-weeny openings in the rear panel. Or if you ever watch channel 2 in your house (the RFI problem is much worse with an exposed cable).

Alternatively, you can run the entire peripheral slot bus out to a separate box and wire up the circuitry and displays on a single board. You can use a homemade plug and cable for your external bus, or a commercial device like "Extend-a-slot", but the cable length for reliable data will be severely restricted with this approach. I ended up using one of those expensive Vector Electronics plug board cards (Jameco Electronics #4609, 415-592-8097 $24.95). Using the second finger edge and a card connector, I ran the bus to an external box with all my circuitry and LEDs. This provided to be a wise choice as all the circuitry was external and easy to debug and repair, and I could still use the rest of the Vector card for building another peripheral, hence not sacrificing a slot.

Another potential problem is trying to use this device in a stuffed-full Apple][or][Plus with the power supply running near its limits. In this case, you'll have to hook up another +5 volt power supply to the display and connect the ground of the extra power supply to the ground of the Apple's power supply. I found using the already whimpy Apple power supply too much for my Bus Monitor (even on my //e), so I used an external power supply (Jameco Electronics #PS72559, $14.95) with at least 2 (preferably 3) amps of +5 volts. The problem is that the LEDs specified in the schematic draw 200-300 millimps each from the +5 volt supply. If you can find lower current LEDs, by all means use them, but be sure that they are **Fully-Decoded Hexadecimal** display LEDs.

The Control Panel

A suggested front panel layout for the display box is shown below. In addition to the 6 LED displays (four for address, two for data), controls on the box include toggle switches for Normal/Slow speed, Slow/Medium/Fast speed control (when in slow mode), Slow/Single Step, and push-button switches for Step and NMI (if you have to ask what NMI does, you don't need it). A single variable control allows fine adjustment of the Speed in the Slow mode.

The Schematic

A slightly abbreviated schematic is shown on the next page. The two LED digits which display the data bus connect to the eight outputs of the 75LS377 as shown: the less significant digit (LED5) connects to the latched output from D0 to D3, and LED6 goes to D4-D7. The four Address LEDs can go directly onto the bus at peripheral slot pins 2-17, unless the cable is significantly over two feet in length. For long cables, it may be necessary to connect LSTTL buffers such as the 74LS07 in series with each of the address lines. The connection scheme for the Address LEDs is:

Pin 2	LED4 D1	Pin 6	LED3 D1
Pin 3	LED4 D2	Pin 7	LED3 D2
Pin 4	LED4 D4	Pin 8	LED3 D4
Pin 5	LED4 D8	Pin 9	LED3 D8
Pin 10	LED2 D1	Pin 14	LED1 D1
Pin 11	LED2 D2	Pin 15	LED1 D2
Pin 12	LED2 D4	Pin 16	LED1 D4
Pin 13	LED2 D8	Pin 17	LED1 D8

Monitor

For the address LEDs, all four latches (pin 5 of the LED) are connected to the output of the 74LS00 as shown, and the Blanking inputs (pin 4) all go to +5 volts. Note that the latch input for the Data LEDs goes to +5 volts.

The cable can be twisted pair (the best for impedance matching) or shielded flat-ribbon (better for RFI, but much harder to find). Regular flat ribbon cable can be used, but you've been warned of the consequences (in fact, it's not that bad, as I use flat ribbon cable without any problems, but the potential is there...). In any case, it must consist of at least 30 conductors:

Address	16 lines
Data	8 lines
control	4 lines
power	2 lines

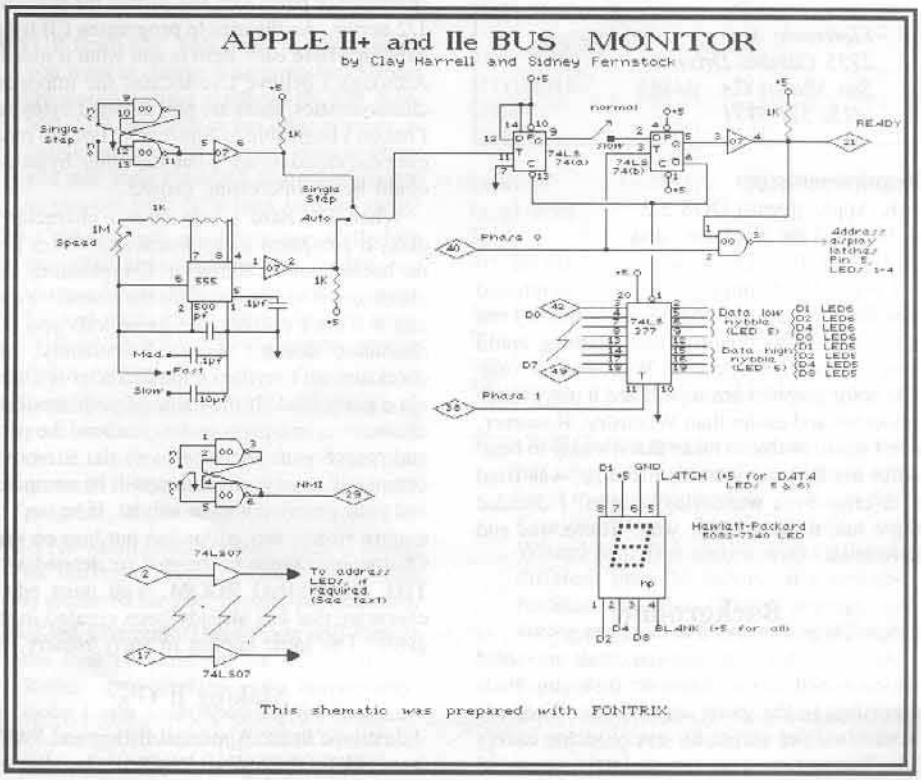
The control lines consist of clock phases 0 and 1, the READY line and the NMI line. The power lines consist of a +5 and a ground. If you are powering the LEDs from the Apple, you should use at least 5 ribbon cable lines for each.

For additional uses for the Bus Monitor refer to the article by Jeffrey Mazur in the column "Hardtalk" in the June 1982 Softalk. It's pretty obvious that you can entertain yourself endlessly by watching your favorite game draw its shapes on the hi-res screen in slow-motion, watching a BASIC program scroll up at one letter per second, or getting the last bug out of your assembly language "magnum opus", but there are a great number of applications in which the Bus Monitor is worth its weight in gold. Probably the most frustrating experience in programming is when your program jumps to oblivion or ties itself up in an endless loop, and the only way to recover is to hit Reset or worse yet, power down to regain control. With the Bus Monitor, you can generally tell when the program is in an endless loop by the stable pattern that appears on the Address and Data displays. You may not always be able to tell how the program got there, but knowing where the loop lives in your program is usually a tremendous help.

Similarly, the nature of the "crash" that occurs is often revealed by the contents of the address and data bus: FFFF is usually a good indication that you tried to go to a nonexistant location or yanked the wrong hardware line and strangled the CPU.

Of course, for deprotection the Bus Monitor is invaluable. For example, finding a nibble count or where a protected DOS is running in memory is a breeze by merely viewing the address bus during the disk access. For example, Penguin's Crimson Crown is copyable with COPYA, but due to a nibble count, the copy will not run. By booting the copy with the Bus Monitor, you can see exactly where the problem occurs when the disk hangs for a moment during the nibble count. The reason the Bus Monitor is of particular importance in this application is that the nibble count is EOR'ed and hidden. Without the Bus Monitor finding a routine that could be anywhere between $00 and $FFFF in memory (including a second bank of $D000-DFFF) could be a very time consuming chore. If the routine is EOR'ed before and after being executed (as in Crimson Crown), it could be impossible! The Bus Monitor could save you hours in boot code tracing and other frivolous activities to find those nibble counts and other protection code!

Many other uses will become obvious as you work with the Bus Monitor. You'll probably wonder why anyone would ever build a computer without one!



THE DRESSING

by Joe Montano

Electronic Arts
2755 Campus Drive
San Mateo, CA 94403
(415) 571-7171

Requirements:
64K Apple][with DOS 3.3
A Bard's Tale Character disk

I was a bit skeptical when I first purchased The Bard's Tale. Having been a Wizardry nut for years, I was doubtful that anything could match it, much less better it! Well, I was wrong. The color graphics are superb and it plays much smoother and easier than Wizardry. However, don't construe that to mean that it's easy to beat! After my favorite wizard, Frobozz, was fried to a crisp by a wandering dragon, I decided there had to be a better way. There was and here it is.

Background

All of the characters that you create are stored on track $00 of the character disk you made according to the game instructions. They are stored two per sector for every sector except $0. That means you can maintain up to 30 characters and/or parties on each disk. The remainder of the disk is game data and the map of Skara Brae.

In each character sector, bytes $00 and $80 begin the name of a character or party and all character or party data follow that for precisely 1/2 sector. As the article progresses I'll try to explain where each item is and what it means. Although I believe I've located the important characteristics, there are still assorted bytes that I haven't been able to figure out. For the more curious, discovering what the other bytes are could be an interesting project.

When The Bard's Tale saves a character to disk, it computes a checksum and writes it to the last byte of the character. On reloading, that checksum is verified against the character data and if it isn't correct, the game tells you that character doesn't exist. Fortunately, the checksum isn't verified if the character is loaded via a party load. If the name of your modified character is on a party roster, just load the party and resave your character with the R(emove) command. A new checksum will be computed and your problem will be solved. If he isn't on a party roster, this editor can put him on one. Characters cannot be created or deleted with THE DRESSING ROOM. You must edit a character that has already been created in the game. The same applies to party rosters.

Making it Go

Just type in the Applesoft listing and SAVE it as "THE DRESSING ROOM" (I realize it's long, but keep the faith). As soon as you RUN it you will get a reminder on loading modified characters into the game and will be advised to place your character disk in the drive of your choice. Just press the number of the drive your disk is in and it will then read the names from the disk and present you with a Master Roster of all your characters. The names will be listed in pairs (two per sector, remember?) and you will be able to edit both, if you wish, before writing them back to the disk. Once a character is selected for edit, each menu displayed will have all command keys listed at the bottom. There is no need for written documentation to operate The Dressing Room.

While debugging the The Dressing Room, it would be a good idea to make line 280 into a REM line until finished because that is the line that writes your characters back to the disk. Each time you return to the main character display, the data is read from the buffer just like it will be written to the disk. So when that looks right, it probably is.

One final note. Since The Dressing Room uses the RWTS directly, it will only work on a DOS that has a standard DOS 3.3 RWTS and uses DOS 3.3 page $03 vectors. If in doubt, boot with your DOS 3.3 System Master. If your character disk is write protected when you try to write your character back it will pretend to write, but it really won't. If you're not sure, read your character back into the buffer and check it.

BARD'S ROOM

What Makes it Go

Here is a basic overview of the main routines and what they do or change. For purposes of this description I will reference only the first character in the sector. For locations of the second, just add $80.

100-120 Set-up: Calls routines for initialization, compiling and displaying the master roster.

130-240 Main menu: This routine calls 'display character' and lists your command keys. Here is also where you can switch, and edit your second character.

250-410 Working subroutines: These do all the little things that have to be done often, like centering titles, drawing borders, etc. Also included is that touchy line 280 that writes to the disk.

420-630 Name check: When the disk sector is read into the buffer, this routine looks for character names. If it finds none, it tells you so and returns to the Master Roster for another selection. The names found are read into variables NX$ for one, NY$ for the other, depending on the variable BUF. The name is stored in bytes $00-$0F, with trailing bytes filled with $FF. The first byte of a party name is always $AA.

640-660 Get experience and gold: Experience is stored in bytes $14-1F in the form of 1 decimal digit per byte (e.g. 326 exp. points would be $03 $02 $06 in bytes $1D-$1F, respectively). Gold is stored in bytes $24-$2F in the same manner. The variables are EP and GL, naturally.

670-790 Display character: This is the routine lets you look at the character you have so you can change him to the character you want.

800-1110 Team edit: This is where you go when you want to change a party roster. Names are stored starting at bytes $10, $20, $30, $40, $50, and $60 in the same manner as individual character sectors. Here is where you add your modified character's name so you can load them back into the game.

1120-1360 Compile master roster: Called by the initialization routine, this reads every sector of track $00 into the buffer (except $0) and reads all the names into the array N$() which is used by the display master roster routine.

1370-1480 Display master roster: This is where you make your selection of which characters to edit and then read them into the buffer. This is also where you exit the program, if you want to do it right.

1490-1710 Main modify menu: This is where the fun begins. A list of 12 changable items is displayed for your perusal and selection.

1720-2760 Changes! These lines will change the first 10 items on the modify menu. Rather than explain each individually, table 1 is a chart showing the locations concerned. Even though Spell levels and Equipment aren't changed in these lines, I added them to make the chart more complete.

Table 1

Item		Bytes
Name		$00-$0F
Attributes		$10-$13
Class		$38
Race		$39
Level	Twice	$20-$21 & $22-$23
Experience		$14-$1F
Gold		$24-$2F
Hit points	Twice	$30-$31 & $32-$33
Spell points	Twice	$34-$35 & $36-$37
Status		$3E
Spell levels		$40-$43
Equipment		$50-$5F

2770-3020 Change spell levels: Each magic user has one byte to record his highest spell level in each magic class. Sorcerer-$40, Conjurer-$41, Magician-$42, and Wizard-$43. This routine works a little bit different than the others. It's probably because all these change routines got boring and I had to do something different.

3030-3630 Edit equipment: It is here, in Garth's Equipment Warehouse, that you may obtain any weapon, armor, magical item, or anything else available in The Bard's Tale. All carried items are stored in bytes

$60-$6F in groups of two bytes each. For instance, byte $60 tells whether the item in byte $61 is unequipped, equipped, or unusable (0, 1 and 2, respectively). There are 127 different items available numbered $01 thru $7F. The initialization reads all equipment into array I$(). The menu allows you to equip, unequip, or drop an item, or it will send you to another menu to add an item. There, all items are listed out for you and you may take your choice. If you are already carrying 8 items, you won't be allowed to the add menu, though. You must drop something first.

3640-4000 Initialization and Introduction: A short routine is poked into memory page $03 for disk access, DOS is told where the buffer is, and all data are read into their respective variable arrays. Set HIMEM. Why 29000? It seemed as good a place as any and it doesn't bomb my programming utilities. Displays first screen with reminder and makes drive selection.

4010-4230 Data: Enough said.

Make it Go Farther

If you read the program listing close in the area of the changes, you'll notice that very few limits are placed upon the editor. This is because I'm not really sure what the limits of the game are. If a character doesn't work properly in the game you should have a pretty good idea about which characteristics you went to the extreme on. It's a simple matter to re-edit the character until it does work right.

Possible modifications I might suggest would be adding a character printout function, creating and deleting characters, copying characters to another sector and modifying them. The problem you face in creating is figuring out the math involved in converting the five attributes into four bytes. If you do, let me know how because I can't get it.

I have no adventure tips or APT's to share because The Bard's Dressing Room is all the APT you should ever need. We don't need to take ALL the fun out of the game, do we?

The Dressing Room

```
10 REM ************************
20 REM *                      *
30 REM *     THE BARD'S       *
40 REM *   DRESSING ROOM      *
50 REM *        BY            *
60 REM *    JOE MONTANO       *
70 REM *                      *
80 REM ************************
90 REM
100 GOSUB 3650 : REM  INITALIZATION
110 GOSUB 1130 : REM COMPILE MASTER ROSTER
120 GOSUB 1380 : REM DISPLAY MASTER ROSTER
130 REM MAIN MENU
140 REM IF TEAM NAME GOTO TEAM EDIT
150 GOSUB 430 : IF ASC (Y$ ) = 170 THEN 810
160 X = FRE (0 )
170 GOSUB 370
180 VTAB 21 : HTAB 3 : PRINT "M)ODIFY" ; TAB( 21
    ); "W)RITE^ TO^ DISK" : HTAB 3 : PRINT
    "R)OSTER^ (NO^ CHNG)" ; TAB( 21 ); "E)DIT^
    2ND^ CHAR. ^ " ; : GET AN$
190 AN = ASC (AN$ )
200 IF AN = 77 THEN 1510
210 IF AN = 82 THEN 120
220 IF AN = 87 THEN 280
230 IF AN = 69 THEN GOSUB 310 : GOTO 150
240 GOTO 180
250 REM END  IT  ALL
260 POKE 34 ,0 :A$ = "DON'T^ FORGET^ TO^ LOCK^
    THE^ DOOR" : HOME : VTAB 12 : GOSUB 410 :
    VTAB 22 : END
270 REM WRITE  BUFFER  TO  DISK
280 POKE 47092 ,2 : CALL 768 : POKE 47092 ,1
290 GOTO 180
300 REM SWAP  NAMES  FROM  BUFFER
310 IF BUF = B1 THEN BUF = B2 : GOTO 330
320 BUF = B1
330 RETURN
340 REM SINGLE  LINE  BORDER
350 VTAB 20 : INVERSE : PRINT AA$ : PRINT AB$; :
    HTAB 39 : PRINT AB$ : PRINT AA$ : NORMAL :
    RETURN
360 REM DOUBLE  LINE  BORDER
370 VTAB 20 : INVERSE : PRINT AA$ : PRINT AB$; :
    HTAB 39 : PRINT AB$ : PRINT AB$; : HTAB 39
    : PRINT AB$ : PRINT AA$ : NORMAL : RETURN
380 REM TRIPLE  LINE  BORDER
390 VTAB 19 : INVERSE : PRINT AA$ : PRINT AB$; :
    HTAB 39 : PRINT AB$ : PRINT AB$; : HTAB 39
    : PRINT AB$ : PRINT AB$; : HTAB 39 : PRINT
    AB$ : PRINT AA$ : NORMAL : RETURN
400 REM CENTER  HEADINGS  AND  OTHER  MESSAGES
410 HTAB 21 - LEN (A$ ) / 2 : PRINT A$ : RETURN
420 REM CHECK  BUFFER  FOR  NAMES  AND  READ  INTO
    VARIABLES  NX$  AND  NY$
430 HOME
440 NX$ = ""
450 FOR L = 0 TO 15
460 N1 = PEEK (BUF + L )
470 IF N1 = 255 THEN 520
480 IF N1 = 0 THEN NC = 1 : GOSUB 310 : IF PEEK (BUF
    ) < > 0 THEN 430
490 IF N1 = 0 AND NC = 1 THEN A$ = "THERE^ ARE^ NO^
    CHARACTERS^ THERE!" : VTAB 12 : GOSUB 410
    : FOR T = 1 TO 1500 : NEXT : POP :NC = 0 : GOTO
    120
500 N1$ = CHR$ (N1 )
510 NX$ = NX$ + N1$
520 NEXT L
530 GOSUB 310 :NY$ = ""
540 FOR L = 0 TO 15
550 N1 = PEEK (BUF + L )
560 IF N1 = 255 THEN 600
570 IF N1 = 0 THEN NY$ = "------" :L = 15 : GOTO 600
580 N1$ = CHR$ (N1 )
590 NY$ = NY$ + N1$
600 NEXT L
610 GOSUB 310
620 REM CHECK  IF  NAME  IS  TEAM  NAME
630 Y$ = LEFT$ (NX$ ,1 ) : IF ASC (Y$ ) = 170 THEN
    RETURN
640 REM READ EXPERIENCE AND GOLD INTO VARIABLES
    EP AND GL
650 EP$ = "" : FOR X = 20 TO 31 :EP$ = EP$ + STR$
    ( PEEK (BUF + X ) ) : NEXT :EP = VAL (EP$ )
660 GL$ = "" : FOR X = 36 TO 47 :GL$ = GL$ + STR$
    ( PEEK (BUF + X ) ) : NEXT :GL = VAL (GL$ )
670 REM DISPLAY  CHARACTER
680 PRINT : INVERSE : PRINT "^ " NX$ "^ " ; :
    NORMAL : PRINT TAB( 26 ); "2ND^ CHARACTER:"
690 INVERSE : PRINT "^ LVL^ " PEEK (BUF + 32 ) *
    256 + PEEK (BUF + 33 ) "^ " R$( PEEK (BUF +
    57 ) ); : NORMAL
700 INVERSE : PRINT "^ " C$( PEEK (BUF + 56 ) );
    "^ " ; : NORMAL : PRINT TAB( 28 ); LEFT$ (NY$
    ,10 )
710 PRINT
720 PRINT "SPELL^ LEVELS:" ; TAB( 21 ); "HIT^
    PTS:^ " PEEK (BUF + 48 ) * 256 + PEEK (BUF
    + 49 ) "/" PEEK (BUF + 50 ) * 256 + PEEK (BUF
    + 51 )
730 PRINT TAB( 21 ); "SPL^ PTS:^ " PEEK (BUF + 52
    ) * 256 + PEEK (BUF + 53 ) "/" PEEK (BUF
    + 54 ) * 256 + PEEK (BUF + 55 )
740 PRINT "SORCERER:^ " PEEK (BUF + 64 ) ; TAB( 21
    ) "EXP:^ " EP
750 PRINT "CONJURER:^ " PEEK (BUF + 65 ) ; TAB( 21
    ) "GOLD:^ " GL
760 PRINT "MAGICIAN:^ " PEEK (BUF + 66 ) ; TAB( 21
    ); "STATUS:^ " ST$( PEEK (BUF + 62 ) )
770 PRINT "^ ^ WIZARD:^ " PEEK (BUF + 67 ) ; TAB(
    21 ); "AC:^ " 10 - PEEK (BUF + 63 )
780 FL = 1 : GOSUB 3070 :FL = 0
790 RETURN
800 REM TEAM  EDIT
810 HOME :A$ = "TEAM^ EDITING" : GOSUB 410
820 PRINT
830 PRINT "TEAM^ NAME:^ " NX$; TAB( 26 ); "2ND^
    CHARACTER"
840 PRINT TAB( 28 ); LEFT$ (NY$ ,10 )
850 FOR NM = 1 TO 6
860 NM$(NM ) = ""
870 FOR L = 0 TO 15
880 N1 = PEEK (BUF + NM * 16 + L )
890 IF N1 = 255 THEN 920
900 N1$ = CHR$ (N1 )
910 NM$(NM ) = NM$(NM ) + N1$
920 NEXT L
930 NEXT NM
940 FOR X = 1 TO 6
950 PRINT X ")^ " NM$(X )
960 NEXT
970 X = FRE (0 ) : GOSUB 390
980 VTAB 20 : HTAB 3 : PRINT "W)RITE^ TO^ DISK"
990 HTAB 3 : PRINT "E)DIT^ 2ND^ CHARACTER" : HTAB
    3 : PRINT "CHANGE^ WHICH^ (0^ EXITS):^ " ;
    : GET CC$ : PRINT CC$;
1000 CC = VAL (CC$ )
1010 CA = ASC (CC$ ) : IF CA = 69 THEN GOSUB 310
    : GOTO 150
1020 IF CA = 87 THEN GOSUB 280
1030 IF CC = 0 OR CC > 6 THEN 120
1040 PRINT : VTAB 19 : CALL - 958
1050 GOSUB 370
1060 VTAB 21
1070 HTAB 3 : PRINT "CHANGE:^ " ;NM$(CC )
1080 HTAB 3 : INPUT "^ ^ ^ ^ TO:^ " ;N2$
1090 FF = 255 : FOR X = 0 TO 15 : POKE BUF + CC *
    16 + X ,FF : NEXT
1100 FOR X = 1 TO LEN (N2$ ) : POKE BUF + CC * 16
    + X - 1 , ASC ( MID$ (N2$ ,X ,1 ) ) + 128 : NEXT
1110 GOTO 810
1120 REM READ ALL NAMES FROM DISK TO VARIABLE N$(
    )
1130 HOME
1140 A$ = "COMPILING^ MASTER^ ROSTER" : VTAB 12
    : FLASH : GOSUB 410 : NORMAL
1150 FOR X = 1 TO 15 : POKE 47084 ,0 : POKE 47085
    ,SEC(X ) : CALL 768 :B1 = BUF
1160 N$(X ) = ""
1170 FOR L = 0 TO 15
1180 N1 = PEEK (B1 + L )
1190 IF N1 = 255 THEN 1230
1200 IF N1 = 0 THEN N$(X ) = "------" :L = 15 : GOTO
    1230
1210 N1$ = CHR$ (N1 )
1220 N$(X ) = N$(X ) + N1$
1230 NEXT L
```

```
1240 NEXT X
1250 FOR X = 1 TO 15 : POKE 47084 ,0 : POKE 47085
     ,SEC(X ) : CALL 768 :X2 = X + 15 :B2 = BUF +
     128
1260 N$(X2 ) = ""
1270 FOR L = 0 TO 15
1280 N1 = PEEK (B2 + L )
1290 IF N1 = 255 THEN 1340
1300 IF N1 = 0 THEN N$(X2 ) = "------" :L = 15 :
     GOTO 1340
1310 N1$ = CHR$ (N1 )
1320 N$(X2 ) = N$(X2 ) + N1$
1330 IF X = 1 THEN POKE 47096 , VAL (Z$ )
1340 NEXT L
1350 NEXT X
1360 RETURN
1370 REM DISPLAY MASTER ROSTER
1380 HOME :A$ = "MASTER^ ROSTER" : GOSUB 410
1390 PRINT
1400 FOR X = 1 TO 15
1410 PRINT CHR$ (X + 64 ) ")^ " ;N$(X ); TAB( 21
     ); "^ /^ " ;N$(X + 15 ) : NEXT
1420 GOSUB 350
1430 VTAB 21 : HTAB 3 : PRINT "YOUR^ CHOICE^
     (<ESC>^ TO^ END):^ " ; : GET CH$
1440 IF ASC (CH$ ) = 27 THEN 260
1450 IF ASC (CH$ ) => 65 AND ASC (CH$ ) =< 79 THEN
     PRINT CH$ :CH = 16 - ( ASC (CH$ ) - 64 ) : GOTO
     1480
1460 PRINT : GOTO 1430
1470 REM READ SELECTED SECTOR FROM DISK
1480 POKE 47084 ,0 : POKE 47085 ,CH : CALL 768 :
     RETURN
1490 REM BEGIN MODIFY ROUTINES
1500 REM MODIFY MENU
1510 HOME :A$ = "MODIFY" : GOSUB 410
1520 A$ = NX$ : GOSUB 410
1530 PRINT : PRINT "^ 1)^ NAME"
1540 PRINT "^ 2)^ RACE"
1550 PRINT "^ 3)^ CLASS"
1560 PRINT "^ 4)^ LEVEL"
1570 PRINT "^ 5)^ EXPERIENCE"
1580 PRINT "^ 6)^ GOLD"
1590 PRINT "^ 7)^ HIT^ POINTS"
1600 PRINT "^ 8)^ SPELL^ POINTS"
1610 PRINT "^ 9)^ STATUS"
1620 PRINT "10)^ MAKE^ ATTRIBUTES^ ALL^ 18"
1630 PRINT "11)^ SPELL^ LEVEL"
1640 PRINT "12)^ EQUIPMENT"
1650 GOSUB 390
1660 VTAB 21 : HTAB 3 : INPUT "ENTER^ CHOICE^
     (<RET>^ EXITS):^ " ;CC$
1670 IF CC$ = "" THEN 150
1680 CC = VAL (CC$ )
1690 IF CC < 1 OR CC > 12 THEN 1510
1700 ON CC GOSUB 1730 ,1830 ,1950 ,2080 ,2180
     ,2280 ,2380 ,2490 ,2600 ,2720 ,2780 ,3050
1710 GOTO 1510
1720 REM EDIT NAME
1730 VTAB 3 :A$ = "NAME^ CHANGE" : GOSUB 410
1740 VTAB 19 : CALL - 958 : GOSUB 370
1750 VTAB 21 : HTAB 3 : PRINT "CURRENT^ NAME:^ "
     NX$
1760 HTAB 3 : INPUT "^ ^ ^ ^ NEW^ NAME:^ " ;NN$
1770 IF NN$ = "" THEN 1810
1780 FF = 255 : FOR X = 0 TO 15 : POKE BUF + X ,FF
     : NEXT
1790 FOR X = 1 TO LEN (NN$ ) : POKE BUF + X - 1 , ASC
     ( MID$ (NN$ ,X ,1 ) ) + 128 : NEXT
1800 NX$ = NN$
1810 RETURN
1820 REM EDIT RACE
1830 HOME :A$ = "CHANGE^ RACE" : GOSUB 410
1840 PRINT : PRINT "CURRENT^ RACE:^ " R$( PEEK
     (BUF + 57 ) )
1850 PRINT
1860 FOR X = 0 TO 6 : PRINT X + 1 ")^ " ;R$(X ) : NEXT
1870 GOSUB 390
1880 VTAB 21 : HTAB 3 : PRINT "ENTER^ CHOICE:^ "
     ; : GET CC$ : PRINT CC$
1890 IF CC$ = CHR$ (13 ) THEN 1930
1900 CC = VAL (CC$ )
1910 IF CC < 1 OR CC > 7 THEN 1880
1920 POKE BUF + 57 ,CC - 1
1930 RETURN
1940 REM EDIT CLASS
1950 HOME :A$ = "CHANGE^ CLASS" : GOSUB 410
1960 PRINT
1970 PRINT "CURRENT^ CLASS:^ " C$( PEEK (BUF +
     56 ) )
1980 PRINT
1990 FOR X = 0 TO 9 : PRINT SPC( X < 9 ) ;X + 1 ")^
     " ;C$(X ) : NEXT
2000 GOSUB 390
2010 VTAB 21 : HTAB 3 : INPUT "ENTER^ CHOICE:^ "
     ;CC$
2020 IF CC$ = "" THEN 2060
2030 CC = VAL (CC$ )
2040 IF CC < 1 OR CC > 10 THEN 2010
2050 POKE BUF + 56 ,CC - 1
2060 RETURN
2070 REM EDIT LEVEL
2080 VTAB 3 :A$ = "CHANGE^ LEVEL" : GOSUB 410
2090 VTAB 19 : CALL - 958 : GOSUB 370 : VTAB 21 :
     HTAB 3 : PRINT "CURRENT^ LEVEL:^ " PEEK
     (BUF + 32 ) * 256 + PEEK (BUF + 33 )
2100 HTAB 3 : INPUT "^ ^ ^ ^ NEW^ LEVEL:^ " ;CC$
2110 IF CC$ = "" THEN 2160
2120 CC = VAL (CC$ )
2130 T = INT (CC / 256 )
2140 POKE BUF + 32 ,T : POKE BUF + 34 ,T
2150 POKE BUF + 33 ,CC - (T * 256 ) : POKE BUF
     + 35 ,CC - (T * 256 )
2160 RETURN
2170 REM EDIT EXPERIENCE
2180 VTAB 3 :A$ = "CHANGE^ EXPERIENCE" : GOSUB
     410
2190 VTAB 19 : CALL - 958 : GOSUB 370
2200 VTAB 21 : HTAB 3 : PRINT "CURRENT^
     EXPERIENCE:^ " ;EP
2210 HTAB 3 : INPUT "^ ^ ^ ^ NEW^ EXPERIENCE:^ "
     ;CC$
2220 IF CC$ = "" THEN 2260
2230 CC = VAL (CC$ ) :CC$ = STR$ (CC ) :CC$ =
     "000000000000" + CC$ :CC$ = RIGHT$ (CC$ ,12
     )
2240 FOR X = 1 TO 12 : POKE BUF + 19 + X , VAL ( MID$
     (CC$ ,X ,1 ) ) : NEXT
2250 EP = CC
2260 RETURN
2270 REM EDIT GOLD
2280 VTAB 3 :A$ = "CHANGE^ GOLD" : GOSUB 410
2290 VTAB 19 : CALL - 958 : GOSUB 370
2300 VTAB 21 : HTAB 3 : PRINT "CURRENT^ GOLD:^ "
     GL
2310 HTAB 3 : INPUT "^ ^ ^ ^ NEW^ GOLD:^ " ;CC$
2320 IF CC$ = "" THEN 2360
2330 CC = VAL (CC$ ) :CC$ = STR$ (CC ) :CC$ =
     "000000000000" + CC$ :CC$ = RIGHT$ (CC$ ,12
     )
2340 FOR X = 1 TO 12 : POKE BUF + 35 + X , VAL ( MID$
     (CC$ ,X ,1 ) ) : NEXT
2350 GL = CC
2360 RETURN
2370 REM EDIT HIT POINTS
2380 VTAB 3 :A$ = "CHANGE^ HIT^ POINTS" : GOSUB
     410
2390 VTAB 19 : CALL - 958 : GOSUB 370
2400 VTAB 21 : HTAB 3 : PRINT "CURRENT^ HIT^
     POINTS:^ " PEEK (BUF + 48 ) * 256 + PEEK
     (BUF + 49 ) "/" PEEK (BUF + 50 ) * 256 + PEEK
     (BUF + 51 )
2410 HTAB 3 : INPUT "^ ^ ^ ^ NEW^ HIT^ POINTS:^
     " ;CC$
2420 IF CC$ = "" THEN 2470
2430 CC = VAL (CC$ )
2440 T = INT (CC / 256 )
2450 POKE BUF + 48 ,T : POKE BUF + 50 ,T
2460 POKE BUF + 49 ,CC - (T * 256 ) : POKE BUF
     + 51 ,CC - (T * 256 )
2470 RETURN
2480 REM EDIT SPELL POINTS
2490 VTAB 3 :A$ = "CHANGE^ SPELL^ POINTS" : GOSUB
     410
2500 VTAB 19 : CALL - 958 : GOSUB 370
2510 VTAB 21 : HTAB 3 : PRINT "CURRENT^ SPELL^
     POINTS:^ " PEEK (BUF + 52 ) * 256 + PEEK
     (BUF + 53 ) "/" PEEK (BUF + 54 ) * 256 + PEEK
     (BUF + 55 )
2520 HTAB 3 : INPUT "^ ^ ^ ^ NEW^ SPELL^ POINTS:^
     " ;CC$
2530 IF CC$ = "" THEN 2580
2540 CC = VAL (CC$ )
2550 T = INT (CC / 256 )
2560 POKE BUF + 52 ,T : POKE BUF + 54 ,T
2570 POKE BUF + 53 ,CC - (T * 256 ) : POKE BUF
     + 55 ,CC - (T * 256 )
2580 RETURN
2590 REM EDIT STATUS
2600 HOME :A$ = "CHANGE^ STATUS" : GOSUB 410
2610 PRINT : PRINT "CURRENT^ STATUS:^ " ST$(
     PEEK (BUF + 62 ) )
2620 PRINT
2630 FOR X = 0 TO 7 : PRINT X + 1 ")^ " ;ST$(X ) :
     NEXT
2640 GOSUB 390
2650 VTAB 21 : HTAB 3 : PRINT "ENTER^ CHOICE:^ "
     ; : GET CC$ : PRINT CC$
2660 IF CC$ = CHR$ (13 ) THEN 2700
2670 CC = VAL (CC$ )
2680 IF CC < 1 OR CC > 7 THEN 2650
2690 POKE BUF + 62 ,CC - 1
2700 RETURN
2710 REM CHANGE ATTRIBUTES TO 18
2720 VTAB 3 :A$ = "CHANGE^ ALL^ ATTRIBUTES^ TO^
     18" : GOSUB 410
2730 POKE BUF + 16 ,148 : POKE BUF + 17 ,146 : POKE
     BUF + 18 ,148 : POKE BUF + 19 ,128
2740 VTAB 19 : CALL - 958 : GOSUB 350 : VTAB 21 :
     HTAB 18 : PRINT "DONE"
2750 FOR T = 1 TO 1500 : NEXT T
2760 RETURN
2770 REM EDIT SPELL LEVELS
2780 HOME :A$ = "CHANGE^ SPELL^ LEVELS" : GOSUB
     410
2790 PRINT : PRINT "CURRENT^ LEVELS:"
2800 PRINT
2810 V = 7 :H = 11 :H1 = 13
2820 B3 = 64
2830 X$ = "<--" :BL$ = "^ ^ ^ "
2840 VTAB 7
2850 PRINT "SORCERER:"
2860 PRINT : PRINT "CONJURER:"
2870 PRINT : PRINT "MAGICIAN:"
2880 PRINT : PRINT "^ ^ WIZARD:"
2890 VTAB V : HTAB H : PRINT PEEK (BUF + B3 )
2900 V = V + 2 :B3 = B3 + 1
2910 IF B3 < 68 GOTO 2890
2920 B3 = 64 :V = 7 : VTAB V : HTAB H1 : PRINT X$
2930 GOSUB 370
```

```
2940 VTAB 21 : HTAB 3 : PRINT "<RET>^ TO^ MOVE^
     POINTER^ ^ ^ L)EAVE"
2950 HTAB 3 : PRINT "<--^ AND^ -->^ TO^ CHANGE^
     POINTS^ ^ " ; : GET CC$
2960 CC = ASC (CC$ )
2970 IF CC = 13 THEN VTAB V : HTAB H1 : PRINT BL$
     :V = V + 2 :B3 = B3 + 1 : IF V > 13 THEN V = 7
     :B3 = 64
2980 IF CC = 13 THEN VTAB V : HTAB H1 : PRINT X$
2990 IF CC = 76 THEN RETURN
3000 IF CC = 8 THEN IF PEEK (BUF + B3 ) > 0 THEN POKE
     BUF + B3 , PEEK (BUF + B3 ) - 1 : VTAB V : HTAB
     H : PRINT PEEK (BUF + B3 )
3010 IF CC = 21 THEN IF PEEK (BUF + B3 ) < 7 THEN
     POKE BUF + B3 , PEEK (BUF + B3 ) + 1 : VTAB V
     : HTAB H : PRINT PEEK (BUF + B3 )
3020 PRINT : GOTO 2940
3030 REM EDIT EQUIPMENT
3040 REM EQUIPMENT MENU
3050 HOME :A$ = "GARTH'S^ EQUIPMENT^ WAREHOUSE"
     : GOSUB 410
3060 A$ = NX$ : GOSUB 410
3070 PRINT
3080 PRINT "CURRENT^ ITEMS^ ( * =EQUIPPED) :"
3090 PRINT :B3 = 80 :N = 1
3100 FOR X = 0 TO 6 STEP 2
3110 E$ = "^ " :E1$ = "^ "
3120 IF PEEK (BUF + B3 + X ) = 1 THEN E$ = " * "
3130 IF PEEK (BUF + B3 + X + 8 ) = 1 THEN E1$ = " * "
3140 PRINT N ")" E$I$( PEEK (BUF + B3 + X + 1 ) ) :
     TAB( 21 ) :N + 4 ")" E1$I$( PEEK (BUF + B3 +
     X + 9 ) )
3150 N = N + 1
3160 NEXT
3170 IF FL = 1 THEN RETURN
3180 X = FRE (0 ) : GOSUB 370
3190 VTAB 21 : HTAB 3 : PRINT "E)QUIP^ ^ ^ ^ A)DD^
     ITEM^ ^ ^ L)EAVE"
3200 HTAB 3 : PRINT "U)NEQUIP^ ^ D)ROP^ ITEM^ ^
     " ; : POKE - 16368 ,0 : GET CC$ : PRINT CC$
3210 CC = ASC (CC$ )
3220 IF CC = 76 THEN RETURN
3230 REM EQUIP ITEM
3240 IF CC = 69 THEN VTAB 19 : CALL - 958 : GOSUB
     390 : VTAB 21 : HTAB 3 : PRINT "WHICH^ ITEM:^
     " ; : GET CW$ : IF VAL (CW$ ) > 0 AND VAL (CW$
     ) < 9 THEN POKE BUF + B3 + VAL (CW$ ) * 2 -
     2 ,1
3250 REM UNEQUIP ITEM
3260 IF CC = 85 THEN VTAB 19 : CALL - 958 : GOSUB
     390 : VTAB 21 : HTAB 3 : PRINT "WHICH^ ITEM:^
     " ; : GET CW$ : IF VAL (CW$ ) > 0 AND VAL (CW$
     ) < 9 THEN POKE BUF + B3 + VAL (CW$ ) * 2 -
     2 ,0
3270 IF CC = 68 THEN GOSUB 3310
3280 IF CC = 65 THEN GOSUB 3370
3290 GOTO 3050
3300 REM DROP ITEM
3310 VTAB 19 : CALL - 958 : GOSUB 390 : VTAB 21 :
     HTAB 3
3320 PRINT "WHICH^ ITEM:^ " ; : GET CW$ :CW = VAL
     (CW$ ) : IF CW < 0 OR CW > 8 THEN 3340
3330 FOR X = CW TO 7 : POKE BUF + B3 + X * 2 - 2
     , PEEK (BUF + B3 + X * 2 ) : POKE BUF + B3
     + X * 2 - 1 , PEEK (BUF + B3 + X * 2 + 1
     ) : NEXT : POKE BUF + B3 + 15 ,0 : POKE BUF +
     B3 + 16 ,0
3340 RETURN
3350 REM ADD ITEM
3360 REM CHECK # OF ITEMS
3370 NI = 9
3380 FOR X = 1 TO 8
3390 Y = PEEK (BUF + B3 + X * 2 - 1 )
3400 IF Y = 0 THEN NI = X :X = 8
3410 NEXT
3420 REM LIST AVAILABLE ITEMS
3430 FI = 1 :LA = 15
3440 IF NI = 9 THEN RETURN
3450 HOME : PRINT
3460 FOR X = FI TO LA
3470 PRINT SPC( X < 100 ) ; SPC( X < 10 ) ;X ")^ "
     LEFT$ (I$(X ) ,14 ) ; TAB( 21 ) :X + 15 ")^ "
     : LEFT$ (I$(X + 15 ) ,14 )
3480 NEXT
3490 GOSUB 370
3500 VTAB 21 : HTAB 3 : PRINT "F)ORWARD" ; TAB(
     21 ) ; "A)DD"
3510 HTAB 3 : PRINT "B)ACKWARD" ; TAB( 21 ) ;
     "L)EAVE^ ^ " ; : GET CC$ : PRINT CC$
3520 CC = ASC (CC$ )
3530 IF CC = 70 AND FI < 113 THEN FI = FI + 30 :LA
     = FI + 14 : GOTO 3450
3540 IF CC = 66 AND FI > 1 THEN FI = FI - 30 :LA =
     FI + 14 : GOTO 3450
3550 IF CC = 76 THEN RETURN
3560 IF CC < > 65 THEN PRINT : GOTO 3490
3570 PRINT : VTAB 19 : CALL - 958 : GOSUB 390 :
     VTAB 21 : HTAB 3
3580 IF CC$ = "" THEN 3450
3590 INPUT "WHICH^ ITEM:^ " ;CC$
3600 CC = VAL (CC$ )
3610 IF CC < 1 OR CC > 127 THEN 3570
3620 POKE BUF + B3 + NI * 2 - 1 ,CC
3630 NI = NI + 1 : GOTO 3440
3640 REM INITIALIZATION
3650 REM SET HIMEM BELOW BUFFER
3660 HIMEM: 29000
3670 DIM N$(30 ) ,SEC(16 ) ,I$(200 )
3680 HOME : INVERSE :A$ = "^ THE^ BARD'S^
     DRESSING^ ROOM^ " : GOSUB 410 :A$ = "^ ^ ^ ^
     ^ ^ BY^ JOE^ MONTANO^ ^ ^ ^ ^ ^ " : GOSUB 410
     : NORMAL : POKE 34 ,2
3690 AA$ = "" : FOR X = 1 TO 39 :AA$ = AA$ + "^ " :
     NEXT X
3700 AB$ = "^ "
3710 REM LOAD SHORT ROUTINE FOR DISK ACCESS
3720 FOR LOC = 768 TO 773 : READ NUM : POKE LOC
     ,NUM : NEXT LOC : POKE 47083 ,0 : POKE 47091
     ,0 : POKE 47092 ,1
3730 DATA 32 ,227 ,3 ,76 ,217 ,3
3740 REM TELL DOS WHERE BUFFER IS AND READ DATA
     INTO VARIABLES
3750 BUF = 29000
3760 POKE 47088 ,BUF - INT (BUF / 256 ) * 256 :
     POKE 47089 , INT (BUF / 256 )
3770 FOR X = 0 TO 6 : READ R$(X ) : NEXT : FOR X =
     0 TO 9 : READ C$(X ) : NEXT : FOR X = 1 TO 127
     : READ I$(X ) : NEXT
3780 FOR X = 0 TO 7 : READ ST$(X ) : NEXT
3790 REM READ SECTOR LOCATIONS INTO SEC( )
3800 FOR X = 15 TO 1 STEP - 1 :SEC(16 - X ) = X : NEXT
3810 REM INTRODUCTION
3820 PRINT : PRINT "THIS^ EDITOR^ WILL^ EDIT^
     CHARACTERS^ ON^ THE"
3830 PRINT "BARD'S^ TALE^ CHARACTER^ DISK. ^ ^
     HOWEVER."
3840 PRINT "YOU^ MUST^ LOAD^ MODIFIED^
     CHARACTERS^ BACK"
3850 PRINT "INTO^ THE^ GAME^ WITH^ A^ PARTY^
     (TEAM)^ NAME."
3860 PRINT : PRINT "EXAMPLE. ^ ^ ^ > * ATEAM"
3870 PRINT : PRINT "IF^ YOU^ TRY^ TO^ LOAD^ A^
     MODIFIED^ CHARACTER"
3880 PRINT "INDIVIDUALLY^ THE^ NAME^ WILL^ NOT^
     BE"
3890 PRINT "RECOGNIZED. ^ ^ ONCE^ LOADED^ AND^
     RESAVED."
3900 PRINT "THE^ PROBLEM^ NO^ LONGER^ EXISTS. ^
     AND^ IT"
3910 PRINT "WILL^ THEREAFTER^ LOAD^ NORMALLY. ^
     ^ YOU^ CAN"
3920 PRINT "MODIFY^ A^ TEAM^ ROSTER^ WITH^ THE^
     NAMES^ OF"
3930 PRINT "THE^ MODIFIED^ CHARACTERS."
3940 PRINT : PRINT "TO^ START, ^ PLACE^
     CHARACTER^ DISK^ INTO^ THE"
3950 PRINT "DRIVE^ OF^ YOUR^ CHOICE^ AND^
     PRESS^ THE"
3960 PRINT "DRIVE^ NUMBER^ (1^ OR^ 2)."
3970 PRINT : PRINT ">" : : GET Z$ : PRINT Z$
3980 IF Z$ < > "1" AND Z$ < > "2" THEN HOME : GOTO
     3820
3990 POKE 47082 , VAL (Z$ )
4000 RETURN
4010 REM DATA
4020 DATA HUMAN ,ELF ,DWARF ,HOBBIT ,HALF-ELF
     ,HALF-ORC ,GNOME
4030 DATA WARRIOR ,WIZARD ,SORCERER ,CONJURER
     ,MAGICIAN ,ROGUE ,BARD ,PALADIN ,HUNTER
     ,MONK
4040 DATA TORCH ,LAMP ,BROADSWORD ,SHORT^ SWORD
     ,DAGGER ,WAR^ AXE ,HALBARD ,MACE ,STAFF
     ,BUCKLER ,TOWER^ SHIELD
4050 DATA LEATHER^ ARMOR ,CHAIN^ MAIL ,SCALE^
     ARMOR ,PLATE^ ARMOR ,ROBES ,HELM
     ,LEATHER^ GLOVES
4060 DATA GAUNTLETS ,MANDOLIN ,HARP ,FLUTE
     ,MTHR^ SWORD ,MTHR^ SHIELD ,MTHR^ CHAIN
     ,MTHR^ SCALE
4070 DATA SAMURAI^ FIGURINE ,BRACERS^ [6]
     ,BARDSWORD ,FIRE^ HORN ,LIGHT^ WAND
     ,MTHR^ DAGGER
4080 DATA MTHR^ HELM ,MTHR^ GLOVES ,MTHR^ AXE
     ,MTHR^ MACE ,MTHR^ PLATE ,OGRE^ FIGURINE
     ,LAK'S^ LYRE
4090 DATA SHIELD^ RING ,DORK^ RING ,FIN'S^
     FLUTE ,KAEL'S^ AXE ,BLOOD^ AXE ,DAYBLADE
     ,SHIELD^ STAFF
4100 DATA ELF^ CLOAK ,HAWKBLADE ,ADMT^ SWORD
     ,ADMT^ SHIELD ,ADMT^ DAGGER ,ADMT^ HELM
     ,ADMT^ GLOVES
4110 DATA ADMT^ MACE ,BROOM ,PUREBLADE
     ,EXORWAND ,ALI'S^ CARPET ,MAGIC^ MOUTH
     ,LUCKSHIELD
4120 DATA GIANT^ FIGURINE ,ADMT^ CHAIN ,ADMT^
     SCALE ,ADMT^ PLATE ,BRACERS^ [4] ,ARC^
     SHIELD
4130 DATA PURE^ SHIELD ,MAGE^ STAFF ,WAR^ STAFF
     ,THIEF'S^ DAGGER ,SOUL^ MACE ,WITHER^
     STAFF
4140 DATA SORCERSTAFF ,SWORD^ OF^ PAK ,HEAL^
     HARP ,GALT'S^ FLUTE ,FROST^ HORN ,DMND^
     SWORD
4150 DATA DMND^ SHIELD ,DMND^ DAGGER ,DMND^
     HELM ,GOLEM^ FIGURINE ,TITAN^ FIGURINE
     ,CONJURSTAFF ,ARC'S^ HAMMER
4160 DATA STAFF^ OF^ LOR ,POWERSTAFF
     ,MOURNBLADE ,DRAGON^ SHIELD ,DMND^ PLATE
     ,WARGLOVES
4170 DATA LOREHELM ,DRAGONWAND ,KIEL'S^
     COMPASS ,SPEEDBOOTS ,FLAME^ HORN ,TRUTH^
     DRUM ,SPIRITDRUM
4180 DATA PIPES^ OF^ PAN ,RING^ OF^ POWER
     ,DEATH^ RING ,YBARRASHIELD ,SPECTRE^ MACE
     ,DAG^ STONE
4190 DATA ARC'S^ EYE ,OGREWAND ,SPIRITHELM
     ,DRAGON^ FIGURINE ,MAGE^ FIGURINE ,TROLL^
     RING
4200 DATA TROLL^ STAFF ,ONYX^ KEY ,CRYSTAL^
     SWORD ,STONE^ BLADE ,TRAVEL^ HELM ,DEATH^
     DAGGER
```

```
4210 DATA MONGO^ FIGURINE ,LICH^ FIGURINE ,EYE
     ,MASTER^ KEY ,WIZWAND ,SILVER^ SQUARE
     ,SILVER^ CIRCLE
4220 DATA SILVER^ TRIANGLE ,THOR^ FIGURINE
     ,OLD^ MAN^ FIGURINE ,SPECTRE^ SNARE
4230 DATA OK ,POISONED ,OLD ,DEAD ,STONE
     ,PARALYSED ,POSSESSED ,INSANE
```

checksums

```
10   - $BADD
20   - $9B13
30   - $4D3B
40   - $AD92
50   - $C899
60   - $FF65
70   - $A3BF
80   - $A900
90   - $924D
100  - $E176
110  - $9B6D
120  - $59F4
130  - $5649
140  - $5F1B
150  - $5678
160  - $DCD0
170  - $C932
180  - $C9D4
190  - $BAC2
200  - $E4A0
210  - $7F28
220  - $9ECF
230  - $D922
240  - $351F
250  - $918F
260  - $E9B9
270  - $9DFC
280  - $69DA
290  - $D91D
300  - $A111
310  - $2FD1
320  - $D561
330  - $D6AB
340  - $5782
350  - $89E1
360  - $3D53
370  - $DE1C
380  - $DC80
390  - $0074
400  - $5458
410  - $2D39
420  - $9529
430  - $08D2
440  - $4749
450  - $60B8
460  - $97A1
470  - $99D3
480  - $5337
490  - $7AF8
500  - $983C
510  - $38A2
520  - $1D6B
530  - $8B6B
540  - $153A
550  - $10AC
560  - $4D87
570  - $4B5B
580  - $F834
590  - $6D63
600  - $4C46
610  - $BCE7
620  - $9CB7
630  - $5DA1
640  - $7AF0
650  - $CE02
660  - $9C0B
670  - $0852
680  - $10C8
690  - $3576
700  - $DEFB
710  - $4060
720  - $B28A
730  - $E904
740  - $86B9
750  - $4C64
760  - $DABE
770  - $2F87
780  - $DD34
790  - $E2BD
800  - $6C9A
810  - $3B69
820  - $78A9
830  - $DCD0
840  - $1DEE
850  - $B15F
860  - $9706
870  - $0A9A
880  - $BB25
890  - $CC77
900  - $BEB6
910  - $4BA8
920  - $2F93
930  - $30DE
940  - $B4E4
950  - $A464
960  - $515E
970  - $CE1C
980  - $2AF1
990  - $B79C
1000 - $DA53
1010 - $ADDB
1020 - $B6CD
1030 - $5C47
1040 - $33A7
1050 - $844D
1060 - $ABE9
1070 - $5429
1080 - $09CD
1090 - $50E1
1100 - $640E
1110 - $146A
1120 - $5ABD
1130 - $ACE6
1140 - $61EC
1150 - $58EE
1160 - $D48C
1170 - $81A2
1180 - $C0BB
1190 - $A63C
1200 - $4A27
1210 - $7B4C
1220 - $B867
1230 - $72AD
1240 - $E34D
1250 - $269B
1260 - $342A
1270 - $23D0
1280 - $046E
1290 - $9394
1300 - $B706
1310 - $949C
1320 - $7780
1330 - $8247
1340 - $0817
1350 - $87B5
1360 - $A024
1370 - $662D
1380 - $AB40
1390 - $12B4
2130 - $CE53
2140 - $FE27
2150 - $BEA0
2160 - $B58E
2170 - $5065
2180 - $F171
2190 - $025B
2200 - $1F2B
2210 - $FAB1
2220 - $6118
2230 - $1319
2240 - $4D9C
2250 - $A570
2260 - $6D43
2270 - $F95D
2280 - $C1ED
2290 - $0CC7
2300 - $EE06
2310 - $FE35
2320 - $3332
2330 - $021B
2340 - $7860
2350 - $CF8F
2360 - $8CD0
2370 - $5C69
2380 - $E4D3
2390 - $7AFB
2400 - $B385
2410 - $6761
2420 - $2F6C
2430 - $33B6
2440 - $68F2
2450 - $45FF
2460 - $CB38
2470 - $BFD9
2480 - $2119
2490 - $44A5
2500 - $938D
2510 - $258B
2520 - $2CA9
2530 - $A389
2540 - $3A5D
2550 - $4344
2560 - $CA96
2570 - $0B6D
2580 - $718D
2590 - $EA4F
2600 - $511E
2610 - $CED8
2620 - $93F7
2630 - $0DB8
2640 - $C4ED
2650 - $A7E4
2660 - $7228
2670 - $B89C
2680 - $C155
2690 - $5CEA
2700 - $32F4
2710 - $4FA6
2720 - $C79B
2730 - $36CF
2740 - $86E0
2750 - $BA4F
2760 - $236A
2770 - $3165
2780 - $B446
2790 - $4D87
2800 - $9838
2810 - $58D7
2820 - $D497
2830 - $46AC
2840 - $ADE6
2850 - $E5EA
2860 - $90AD
2870 - $F620
2880 - $4565
2890 - $ACB9
2900 - $D513
2910 - $198E
2920 - $387F
2930 - $C088
2940 - $8A3C
2950 - $7266
2960 - $87D5
2970 - $9475
2980 - $212D
2990 - $678F
3000 - $1A30
3010 - $DF2A
3020 - $95BC
3030 - $1DAC
3040 - $D553
3050 - $C73A
3060 - $9300
3070 - $C98A
3080 - $7E5B
3090 - $A9B3
3100 - $F632
3110 - $4310
3120 - $526D
3130 - $D28D
3140 - $8D67
3150 - $2A88
3160 - $0443
3170 - $476F
3180 - $B9E6
3190 - $D5BF
3200 - $CBC0
3210 - $644A
3220 - $C69B
3230 - $134E
3240 - $2D84
3250 - $9879
3260 - $6A87
3270 - $AC05
3280 - $3BD7
3290 - $FBF5
3300 - $C592
3310 - $ABA4
3320 - $EA7A
3330 - $9FDE
3340 - $C314
3350 - $AA14
3360 - $3AAE
3370 - $C57C
3380 - $53D6
3390 - $8579
3400 - $A361
3410 - $2B2C
3420 - $1A9D
3430 - $D977
3440 - $AE14
3450 - $BAF8
3460 - $7D37
3470 - $A2DC
3480 - $42A2
3490 - $0EAA
3500 - $FF37
3510 - $5EAA
1400 - $9DE1
1410 - $702B
1420 - $842B
1430 - $90BD
1440 - $21C1
1450 - $48C4
1460 - $BFF1
1470 - $8F5E
1480 - $ABE1
1490 - $CFD4
1500 - $3651
1510 - $DA6B
1520 - $A3D3
1530 - $6D72
1540 - $330D
1550 - $59BA
1560 - $9FA2
1570 - $0B20
1580 - $7A8D
1590 - $DDBB
1600 - $D484
1610 - $DB46
1620 - $F91B
1630 - $C470
1640 - $B628
1650 - $A257
1660 - $76BD
1670 - $13F7
1680 - $6588
1690 - $774D
1700 - $C61A
1710 - $3916
1720 - $DB02
1730 - $82EB
1740 - $50C6
1750 - $3CBE
1760 - $A200
1770 - $375A
1780 - $88A5
1790 - $BF10
1800 - $1288
1810 - $CC38
1820 - $7764
1830 - $0C18
1840 - $7EC4
1850 - $7BBD
1860 - $2A6D
1870 - $BF1A
1880 - $3E5B
1890 - $DDA6
1900 - $C22B
1910 - $B332
1920 - $B9C3
1930 - $93D5
1940 - $F6C9
1950 - $29EF
1960 - $0C8D
1970 - $E569
1980 - $A25F
1990 - $FC95
2000 - $0690
2010 - $6FF1
2020 - $532C
2030 - $B87C
2040 - $9B46
2050 - $11B3
2060 - $627D
2070 - $67E6
2080 - $357F
2090 - $03BC
2100 - $F2E7
2110 - $8193
2120 - $4AC8
3520 - $A4FF
3530 - $A570
3540 - $E0B3
3550 - $7050
3560 - $8F3B
3570 - $730E
3580 - $A812
3590 - $826B
3600 - $953D
3610 - $EE15
3620 - $7E95
3630 - $62AB
3640 - $5E07
3650 - $4A5A
3660 - $3102
3670 - $94AB
3680 - $9E8B
3690 - $8D62
3700 - $8DB9
3710 - $64DF
3720 - $E33E
3730 - $1F26
3740 - $9F8E
3750 - $6375
3760 - $5CF9
3770 - $8331
3780 - $4BCE
3790 - $3FD7
3800 - $3D1D
3810 - $A091
3820 - $D68D
3830 - $59D1
3840 - $97D3
3850 - $3CDC
3860 - $65A7
3870 - $C6F6
3880 - $4EC8
3890 - $121E
3900 - $EE2B
3910 - $276D
3920 - $469B
3930 - $1DB0
3940 - $3D4E
3950 - $000A
3960 - $729A
3970 - $478E
3980 - $18D9
3990 - $4F8E
4000 - $E628
4010 - $BBF9
4020 - $B652
4030 - $A14E
4040 - $317C
4050 - $3D2A
4060 - $2CBF
4070 - $2907
4080 - $8993
4090 - $67E2
4100 - $E8CE
4110 - $AE7E
4120 - $E50C
4130 - $5E54
4140 - $673B
4150 - $6E19
4160 - $C4CD
4170 - $790E
4180 - $646C
4190 - $6CE1
4200 - $F8F5
4210 - $650D
4220 - $420E
4230 - $77CB
```

Mousepaint
for non-Apples

by Keven D. Miller

Requirements:
64K Apple][or compatible computer
1 Disk drive
Apple Mousepaint diskette

Mousepaint, by Apple Computer Inc., is a scaled down version of the MacIntosh's Macpaint written for the Apple][computer. Using a mouse for computer control, you can create colorful hi-res pictures very rapidly with several different drawing options. The Mousepaint package includes a disk containing Mousepaint and a demo program with the ProDOS operating system, a mouse with a hardware card to place into any slot, and a manual describing installation of the mouse and operation of Mousepaint. The manual also includes two sections describing programmable access to the mouse through BASIC and assembly language. There are several other files on the disk besides Mousepaint and the demo, many of which are used by these programs. However, no definition is given for them and there are no supplied utilities for making menu bars and using the mouse. The demo program is written in BASIC so it could be used as an example, but it includes very little documentation.

With my Franklin Ace 100, I found that I had 2 obstacles to overcome. First, I had to get ProDOS to boot, and second, I had to get Mousepaint to run. Both of these programs access certain ROM addresses to detect which series of the Apple][it is running on.

ProDOS

Before the ProDOS patches were published in COMPUTIST No. 9, Page 18, I had managed to get ProDOS booting via. boot-code-tracing. The locations I found to patch are as follows:

Trk	Sect	Bytes	Old-values	New-values
$00	$01	$55-57	AE C0 FB	EA A2 EA
$01	$09	$60-61	A9 00 00	A5 0C
$01	$0C	$B4-B6	AE B3 FB	EA A2 EA
$01	$0C	$C7-C9	AE 1E FB	EA A2 AD

You can make these patches using a sector editor; personally I prefer DiskEdit. Because ProDOS is a system file, like any other file, these track / sector locations could possibly be different. But I believe this is unlikely to happen. The patch indicated above lets ProDOS think it is running on an Apple][with auto-boot ROM. For those of you who do not have access to various Apple computers, here are 3 F8-ROM locations that I have found to be used for computer identification.

Adr	][	][+	//e	//c	Ace 100	Ace 1000
FB1E:	AD	AD	AD	4C	AD	AD
FBB3:	38	EA	06	06	C1	EA
FBC0:	60	EA	EA	00	00	EA

Mousepaint

From my inspection of Mousepaint, I only found references to $FBC0. All of these are in the form of load register (LDA, LDX, LDY). The patch is to change these references to load register with the desired value from the table above.

Two files need to be patched: MP.INIT and MP.OBJ. First, we need to disable the auto-start BASIC program STARTUP.

Startup

Boot up the Mousepaint disk. As soon as you see the BASIC prompt, type [-C] to stop the program. By adding the following BASIC line, the STARTUP program will exit to BASIC when booting from the disk.

45 END

To continue the STARTUP program you can type "RUN 50".

Mp.Init

Looking at the extended directory listing via. the "CATALOG" command, MP.INIT shows a BLOAD address at $230 and a length of 157 bytes. After BLOADing MP.INIT at $230, and making the patches, I found that BSAVEing from this area altered some of the code before saving it to disk. However, MP.INIT is loaded by another program which specifies the load address, so we can BLOAD and BSAVE it anywhere.

PREFIX MP
CALL -151
BLOAD MP.INIT,A$2030
2057L

Here is what we will do to this code:

```
2057- 0A        ASL
2058- 8D 04 03  STA  $0304
205B- D0 01     BNE  $205E
205D- 60        RTS
205E- A0 00     LDY  #$00
2060- AD C0 FB  LDA  $FBC0  Change here to
2063- F0 02     BEQ  $2067  NOP LDA #$EA
2065- A0 02     LDY  #$02   ($EA $A9 $EA)
2067- 84 82     STY  $82
2069- A0 19     LDY  #$19
206B- 20 B0 02  JSR  $02B0
206E- A4 82     LDY  $82
2070- A9 00     LDA  #$00
2072- 8D 78 04  STA  $0478
2075- 8D 78 05  STA  $0578
2078- B9 C5 02  LDA  $02C5,Y
207B- 8D F8 04  STA  $04F8
```

Type the following:

2060:EA A9 EA
BSAVE MP.INIT,A$2030,L157

Mp.Obj

MP.OBJ BLOADs at $4000 with a length of 19968 bytes. Five patches are needed here. Type the following:

BLOAD MP.OBJ,A$4000
6154LL

Here is what we are going to do to this code:

```
6154- A5 45     LDA $45
6156- 48        PHA
6157- 8A        TXA
6158- 48        PHA
6159- 98        TYA
615A- 48        PHA
615B- AC C0 FB  LDY $FBC0 Change: EA A0 EA
615E- D0 01     BNE $6161 (NOP LDY #$EA)
6160- 58        CLI
6161- A2 0C     LDX #$0C
6163- B5 80     LDA $80,X
6165- 9D 68 5F  STA $5F68,X
6168- CA        DEX
6169- 10 F8     BPL $6163
616B- A0 13     LDY #$13
616D- 20 8A 5F  JSR $5F8A
6170- B0 60     BCS $61D2
6172- A0 14     LDY #$14
6174- 20 8A 5F  JSR $5F8A
6177- AE 03 03  LDX $0303
617A- BD B8 04  LDA $04B8,X
617D- AC C0 FB  LDY $FBC0 Change: EA A0 EA
6180- F0 01     BEQ $6183
6182- 4A        LSR
6183- 8D 19 5F  STA $5F19
6186- BD B8 03  LDA $03B8,X
6189- AC C0 FB  LDY $FBC0 Change: EA A0 EA
618C- F0 01     BEQ $618F
618E- 6A        ROR
618F- 8D 18 5F  STA $5F18
6192- 18        CLC
6193- BD 38 05  LDA $0538,X
6196- AC C0 FB  LDY $FBC0 Change: EA A0 EA
6199- F0 01     BEQ $619C
619B- 4A        LSR
619C- BD 38 04  LDA $0438,X
619F- AC C0 FB  LDY $FBC0 Change: EA A0 EA
61A2- F0 01     BEQ $61A5
61A4- 6A        ROR
61A5- 8D 1A 5F  STA $5F1A
```

Type the following:

615B:EA A0 EA
617D:EA A0 EA
6189:EA A0 EA
6196:EA A0 EA
619F:EA A0 EA
BSAVE MP.OBJ,A$4000,L19968
[-C]

Following the above procedure should give you a working ProDOS environment as well as an operational mouse with Mousepaint.

softkey for...

Flight Simulator II

v1.05

by Eric Sunshine

SubLOGIC Corp.
713 Edgebrook Dr.
Champaign, IL 61820

Editors Note: *Due to space limitations, only the text and hexdump portions of this article are presented here. All source code that would normally accompany this article will appear in COMPUTIST No. 37.*

Requirements:
at least 48K and Applesoft
A blank disk
A blank initialized work disk
A sector editor that can write specific memory pages to specific sectors (such as **The Inspector** or **SREAD / SWRITE** from COMPUTIST No. 24)

Deprotecting A2-FS2 was quite challenging, but more importantly, it was fun! I hope that you have some fun too; or at least feel satisfied knowing that you've conquered a beast that has been on the Most Wanted List longer than most of us care to remember.

Since the protection scheme used on Flight Simulator II is so involved, I assume that changes, if any, are minimal for versions released after 1.05. For the more enhanced versions, though, some modification to the softkey may be necessary. *(ed. note: this is probably true for earlier versions too.)*

The Boot Process

The firmware located on the disk controller card must be able to read Track 0, Sector 0 from

any bootable disk. The sector is loaded into memory locations $800 through $8FF. Then, in a loop, sequential sectors are read in until the number of sectors loaded equals the number in memory location $800. Since this location contains a "01" on the A2-FS2 disk as with DOS 3.3, only one sector is loaded. Once the firmware is finished loading sectors it begins execution of the code at $801 via a machine language "JuMP" instruction.

Here, the boot process of A2-FS2 begins to differ drastically from a normal DOS 3.3 disk. First, the code at $801 clears hi-res page 2 ($4000-$5FFF), and then reveals it. Once this is accomplished, it loads memory locations $1D00 through $1FFF with the second stage of the A2-FS2 boot. This code is stored on Track 0, and is encoded in a manner similar to the "4 and 4" encoding technique used in the "address" fields of normal DOS 3.3 diskettes. Execution then proceeds to location $1D00, once again via a "JuMP" instruction.

The code at $1D00 begins by filling memory locations $2000 through $25FF with more data from track 0. This data, also stored with a modified "4 and 4" encoding technique, is the heart of the A2-FS2 DOS. Contained here ($2000-$25FF) are all of the reading and writing subroutines necessary for proper operation of the flight simulator.

Notice that the A2-FS2 disk operating system occupies the same area of memory as hi-res page 1. In the 48K system, it is constantly overwritten during game play. In order to retain disk access, a short routine at $1F06 recalibrates the drive arm (brings it back to track 0) and reads in DOS each time it is needed. In the 64K computer, on the other hand, a nearly exact copy of the disk operating system is loaded into the language card, along with other features exclusive to this size system.

Once DOS is loaded, the code at $1D00 uses it to read the "main" part of the boot into memory locations $A7E0 through $B2DF. Control is then passed to memory location $A7E2. This "main" section of the boot carries out such functions as loading the A2-FS2 logo, and determining the amount of memory in the system. The only part of this code ($A7E0 - $B2DF) that needs modification is the part which loads the language card. Other than that, it need not be dealt with.

The Protection

The protection on the A2-FS2 disk involves checksum tests, a track-by-track nibble count, and constantly changing prologue (or "header") marks. In addition, Flight Simulator was originally written to the disk at a slower than normal speed. This technique, coupled with its special track format allows 38 tracks of data to be stored on only 35.

The track with the least protection is track 0; the format is "4 and 4" and there is no nibble count. The format on the rest of the tracks is quite involved, although it does resemble the "6 and 2" encoding technique used in the data fields of normal DOS 3.3 diskettes.

Unlike DOS 3.3, an A2-FS2 track is not divided into sectors and has only one "gap". But, it does have a prologue to identify the start of data. When information is needed from the A2-FS2 disk, an entire track is read in. Once "postnibbled", this data is partitioned into 4 separate areas of 1024 bytes each.

The "6 and 2" encoding scheme modifies data in such a way as to make it suitable for storage on disk. (see *Beneath Apple DOS*). A sector of data (256 bytes) is broken up into two parts; a block of 256 bytes making up the primary data, and one of 86, making up the secondary data. In all, a total of 342 bytes are required to write one sector of information to a disk.

An A2-FS2 track consists of 4 parts. The first 4096 bytes make up 16 blocks of primary data (256 X 16 = 4096). The next 1376 bytes make up 16 blocks of secondary data (86 X 16 = 1376). The third part of the track, 673 bytes, is what I call the "language card data." The final 384 bytes are used for the nibble count.

If you have been keeping up with the math, you may have noticed that the first 5472 bytes of data would fit very nicely into the 16 sectors of a DOS 3.3 track. And, of course, since we intend to remove the copy-protection, the final 384 bytes can be discarded. The real problem lies in finding a place to put the 673 bytes of "language card data."

What Needs To Be Done

A lot of work is involved in softkeying Flight Simulator II, so make sure that you read the instructions well. I have compiled a list of those things which need to be done in order to successfully deprotect A2-FS2 (although not necessarily in this order).

a) Both the first stage boot ($1D00-$1FFF) and the A2-FS2 DOS must be captured, along with the "language card data."

b) The data on the disk must be converted to standard DOS 3.3 format using the first two pieces of code from step 1.

c) The routine at $1F06 which reads in the A2-FS2 DOS must be rewritten so as to maintain 48K compatibility.

d) Both the DOS at $2000-$25FF and the DOS in the language card must be rewritten; not only the read routines, but also, those that write, since the "Save Mode Library" function must be preserved.

e) The routine which loads the language card with data (from the third part of an A2-FS2 track) must be rewritten.

f) A home for the "language card data" must be found. (This is difficult, since the majority of the A2-FS2 disk is used.)

The Softkey

1) Start by making sure you have a 48K slave disk with a small or DELETEd HELLO program. Make one if necessary.

2) Type in each of the hexdumps accompanying this article and BSAVE them with the appropriate parameters. Make sure you have typed them correctly. This saves you trouble later and makes the softkey easier to follow.

COPY performs a Super IOB-style translation of the disk from A2-FS2's DOS to DOS 3.3.

BSAVE COPY,A$1000,L$C7

NEW DOS replaces FS2's "load a track" routine.

BSAVE NEW DOS,A$23D0,L$E7

LC DOS is the language card version of NEW DOS. Just type in the shaded portions on top of NEW DOS (after saving it).

BSAVE LC DOS,A$23D0,L$E7

WRITE replaces FS2's "write a track".

BSAVE WRITE,A$21E3,L$F8

You guessed it. LC WRITE is the language card version of WRITE.

BSAVE LC WRITE,A$21E3,L$F8

BOOT1 is the new boot sector (track 0, sector 0) for FS2.

BSAVE BOOT1,A$800,L$84

LC LOADER puts the new FS2 DOS into the language card, if available.

BSAVE LC LOADER,A$20AF,L$58

3) INITialize a blank disk to transfer A2-FS2 onto.

4) Make certain that your A2-FS2 disk is write-protected, and then insert it in the drive. Next, move the Boot ROM down to RAM (assuming you don't have a //c) and tell it to jump into the Monitor after it has loaded track 0, sector 0 (Boot 1).

```
CALL-151
1600<C600.C6FFM
16F8:8D E8 C0 4C 59 FF
1600G
```

5) Now modify Boot 1 so that, after loading Boot 2 ($1D00-$1FFF), it enters the Monitor, rather than continuing execution at $1D00.

```
7FF:A2 60
882:8D E8 C0 4C 59 FF
C0E9
7FFG
```

6) Modify Boot 2, making it drop into the Monitor after loading the A2-FS2 DOS ($2000-$25FF), and the "main" part of the boot at $A7E0-$B2DF.

```
1D2D:8D E8 C0 4C 59 FF
1D03G
```

7) Now fool the main boot into thinking that there is a language card, even if there isn't one, and tell it to load this data into memory starting at $4000, instead of into the language card at $D000. Note that the "language card data" is on every track, just following the 16 blocks of secondary data. Therefore, tracks $1 through $22 must be accessed in order to capture all of the code.

```
ACB4:40
ACCC:E8 C0
C0E9
ACAEG
```

8) At this point, all of the data and code that cannot be easily transferred to the backup is in memory. Boot 1 at $800-8FF can be discarded since we will write our own. As for the main boot, this will be transferred with the rest of the data when we copy the disk as a whole. The remaining code will be saved for later modification. But first, replace the bytes that were altered when we inserted our breakpoints. Then boot up your work disk and save the modules.

```
1D2D:4C E2 A7 AD 01 1E
C600G
BSAVE BOOT2 (1D00-1FFF),A$1D00
  ,L$300
BSAVE DOS (2000-25FF),A$2000
  ,L$600
BSAVE LANGUAGE CARD,A$4000
  ,L$2CA0
```

9) ***Editors Note:*** *Hopefully you have two drives, or else you will be swapping disks about 70 times (2 disks * 35 tracks) in this step.*

The program "COPY" that you put on your disk earlier was designed to copy tracks $1 through $22 of the flight simulator onto a standard DOS 3.3 disk (except, of course, for the "language card data" and the nibble count). Provisions have been made to allow the use of one or two drives. When the program first starts up, a question mark along with a flashing cursor will appear in the lower left hand corner of the screen. Here, you must type in either a "1" or a "2", depending on whether you have one or two drives. If you type a "1", the program will prompt you to put in the SOURCE disk by printing an "S" in the same corner of the screen. Likewise, it will print a "T" when it expects you to insert the TARGET disk. The TARGET disk, of course, will be the blank disk which was initialized in step 1. Once the correct disk is in place, any keypress will continue the copy process. NOTE: Before pressing the "2" key, be sure that the flight simulator disk is in drive 1, and the blank disk is in drive 2, since there is no prompting when using two drives.

The program "COPY" uses both the A2-FS2 DOS and the Boot 2 file, so load these files from your work disk before you start the copier.

```
BLOAD COPY
BLOAD BOOT2 (1D00-1FFF)
BLOAD DOS (2000-25FF)
CALL-151
1000G
```

10) When complete, place your work disk in the drive and boot it.

```
C600G
```

At this point, you have all of the data necessary to make a backup, so put your A2-FS2 disk away for safe-keeping.

11) It is now time to rewrite the disk access routines. We'll start by fixing the code that loads the A2-FS2 DOS from track 0. Whenever a disk function is requested, a subroutine at $1EC4 is called. This subroutine checks to see if DOS is loaded and goes to $1F06 if it is not. Here the drive arm is recalibrated and DOS is read in. That is what we need to rewrite.

How do we make it read from a normal DOS 3.3 disk? We can't use a normal RWTS and we can't write our own (at least a complete one), since there is not enough room in memory. One way to make space would be to remove some of the data used by FS2 to calculate checksums and find correct prologues ($1E00-$1E54 and $1E95-$1EAC).

Instead, we can use a routine that is always in memory and does exactly what we want: it recalibrates the drive arm and reads sectors from track 0. Yes, of course, the boot ROM on the disk controller card does just that. But wait a minute. If we call this routine we will merely succeed in rebooting the disk. This can be easily circumvented, though, by creating a dual purpose Boot 1 in track 0, sector 0. (Refer to the assembly listing of DOS LOADER with this article.) All we have to do is tell this special Boot 1 that it should reload DOS rather than boot the disk.

Of course, when we call the boot ROM it will destroy some valuable memory, so we will have to move this vulnerable data out of the way before we start the load. The A2-FS2 DOS itself is loaded into memory from $2000 to $25FF. When it reads in a track, the data is stored in memory from $2600 to $3F7F, so our routine can move our sensitive memory there as follows:

FROM:	TO:
$0000	$3600
$0100	$3700
$0300	$3800
$0800	$3900

To tell our new Boot 1 that we want it to load DOS, we must give it a signal. We will do this by storing a "$49" into memory location $0, and a "$23" into location $1, although any numbers could have been used. The program "DOS LOADER" is written to do all that.

The listing "BOOT 1" is a special piece of code, since it must be able to determine whether to load DOS or actually boot the disk. In addition, it must emulate the original Boot 1 found on the flight simulator by clearing hi-res page 2 and revealing it.

Next we have to rewrite the A2-FS2 DOS ($2000-$25FF). We will start with the read routines. Any good read routine should be able to select the correct track for reading, read that track, and postnibble the data. A buffer should also be supplied. The A2-FS2 DOS already has a track-seek routine, and since it will function regardless of the disk format, we will not have to write one. As for the buffer, we can use the same memory as the original DOS does ($2600-$35FF). We will have to write our own postnibble routine, but since the encoding values are the same, we may use the "6 and 2 read translate table" already present. So, as you can see, all we really have to do is write a routine to read the disk.

By the time the actual read routine is called, the drive arm is at the correct track, the "read translate table" is set up, the buffer is available, and the drive is running. Therefore, we merely have to read 16 sectors (1 track) of data into the buffer and return control to the calling routine. In this way, we accomplish the same function as the code which will be replaced, the flight simulator being none the wiser.

The routine called "NEW DOS" (see the appropriate listing) may be divided into 3 parts. The first part counts the sectors as they are read. The second reads the address field of a sector and verifies that it is the correct one, and the last part reads the actual data field and postnibbles the data.

The routine which we need to replace in the A2-FS2 DOS ($2000-$25FF) starts at memory location $23D0. This same routine in the "language card DOS" starts at $D7B5.

Since the DOS in the language card is nearly identical, all we need to do is change the starting address (.OR) in line 1 of the listing, and reassemble the file (using "LC DOS" for the name). Make the file originate at $D7B5 (i.e. .OR $D7B5).

Flight Simulator][writes to a disk when given the command to save the "User Mode Library". Therefore, we must rewrite the disk write routines, in both the DOS at $2000 and the DOS in the language card. By the time the write routine is called, the drive arm is at the proper location, the disk is spinning, and the buffer is ready. We then, must prenibble the sectors one at a time, and in a loop, write all 16 to the disk. In order to write each sector to its proper location, the "new" write routine calls the "Read Address Field" section of NEW DOS, or LC DOS, as the case may be.

As for the write routine in the language card, lines 1 and 2 of the source listing must be changed. The origin becomes $D5C8 (i.e. .OR $D5C8), and the Read Address location becomes $D7E2 (i.e. ADDRESS .EQ $D7E2). This becomes the file "LC WRITE".

The subroutine at $AC9B in the "main" boot determines the presence of a language card. If one is found, a routine at $20AF in the A2-FS2 DOS is called. This routine strips the "language card data" off the end of the track in memory, and moves it to the language card. It is called a total of 34 times, one time per track.

The routine "LC LOADER" accesses a table to guide it through the loading of the language card. The data is read in backwards (i.e. Sector $F is loaded, then $E, etc.). Since the routine needs to be called only once, a slight modification to the code at $AC9B will be made (later).

Anyway...

12) We're ready to start modifying the copy of FS2. At this point, it is necessary to have a sector editor which is capable of writing selected memory to the disk, such as The Inspector. SREAD/SWRITE from COMPUTIST No. 24 can perform the desired function. Put your sector editor into memory now.

First, we will put Boot 1 onto the disk. Clear page $800, and load the code from the work

disk. This is a good time to write protect your work disk.

```
CALL-151
800:0 N 801<800.8FEM
BLOAD BOOT1,A$800
```

Next, insert the disk onto which you copied A2-FS2 and enter your sector editor, or use SREAD/SWRITE. Boot 1 must be written to Track 0, Sector 0, so do so now.

write page	to track	sector
$0800	$00	$00

Example using SWRITE:

```
SWRITE T$0,S$0,A$800
```

13) The DOS at $2000 requires new read and write routines plus one to load the language card, so load the original DOS and then load the new segments on top the old. Also, a small patch must be made to prevent the flight simulator from initializing the disk before it saves the "User Mode Library". A little "cleanup" to make the code list nicely can be done at this time.

```
BLOAD DOS (2000-25FF),A$2000
BLOAD NEW DOS,A$23D0
BLOAD WRITE,A$21E3
BLOAD LC LOADER,A$20AF
CALL-151
2107:0 N 2108<2107.210EM
2190:0 N 2191<2190.21E1M
22DB:0 N 22DC<22DB.2379M
```

(the next instruction prevents INIT from occurring)

```
238A:18 60 0 N 238D<238C.23CEM
24B7:0 N 24B8<24B7.2576M
```

Save this code ($2000-$25FF) to Track 0, sectors 1 through 6. A hint for SRWITE users: A BASIC program to write the sectors sequentially in a loop will save a lot of typing. Don't forget to translate the hexadecimal numbers where necessary.

write page	to track	sector
$2000	$00	$01
$2100	$00	$02
$2200	$00	$03
$2300	$00	$04
$2400	$00	$05
$2500	$00	$06

14) The "DOS LOADER" in Boot 2 must be replaced. Load BOOT2 and then load the new routine on top of the old. A little cleanup is in order here too, so the last command shown clears some unused data.

```
BLOAD BOOT2 (1D00-1FFF),A$1D00
BLOAD DOS LOADER,A$1F06
CALL-151
1F5A:0 N 1F5B<1F5A.1F7BM
```

Write this code ($1D00-$1FFF) to Track 0, Sectors 7 through 9.

write page	to track	sector
$1D00	$00	$07
$1E00	$00	$08
$1F00	$00	$09

15) Now comes the hard part. We must search the disk for a place to store the "language card data". Looking through this block of code reveals that not all of it needs to be kept. In general, for A2-FS2 version 1.05, the information from $D3D0 to $F3FF and F600 to $F9FF turn out to be valid data. The rest may be discarded ($D000-$D3CF, $E400-$E5FF, and $FA00-$FCA0). The disk itself seems to contain a number of unused sectors. Tracks $21 and $22, plus Sectors $B through $F of Track $20 are free, so we may place our "language card data" here (a perfect fit).

Of course, we must load the new read and write routines, and while we're at it, clean up some memory for a nicer looking disassembly. In addition, a patch must be made to prevent disk initialization during a save of the "User Mode Library". It is also necessary to tell the program not to reload the language card upon reset.

```
BLOAD LANGUAGE CARD,A$4000
BLOAD LC DOS,A$47B5
BLOAD LC WRITE,A$45C8
CALL-151
```

This patch prevents reloading of the language card:

```
4494:18 60 0 N 4497<4496.44F3M
4575:0 N 4576<4575.45C6M
46C0:0 N 46C1<46C0.475EM
```

The following line prevents INIT from occuring:

```
476F:18 60 0 N 4772<4771.47B3M
489C:0 N 489D<489C.495BM
```

Insert the disk onto which you copied the simulator, and save the code to it. The block of memory, $4300 through $63FF will be saved sequentially starting on Track $20, Sector $B and will end on Track $22, Sector $B. The rest of the memory $F600-$F9FF will be saved to Track $22, from Sector $C to Sector $F.

mem pages	track	Sector ->	Sector
$4300-47FF	$20	$0B	$0F
$4800-57FF	$21	$00	$0F
$5800-63FF	$22	$00	$0B
$6600-69FF	$22	$0C	$0F

Example: write page $4300 to track $20, sector $B; write page $4400 to sector $C, and so on until page $4700 is written to sector $F.

16) One sector edit has to be made. Since the new routine which loads the language card needs to be called only once, we have to make a small modification to the code at $AC9B. This code lies in Sector $C, on Track $9. Change the bytes $E2 through $E8 to "$EA" (the NOP instruction).

Track	Sector	Bytes	Change To (NOP)
$09	$0C	$E2-$E8	EA EA EA

You now have a completely deprotected (COPYAable) copy of A2-FS2. Write protect the disk, so as not to accidentally destroy it.

Some Notes

A lot of time and effort went into retaining as many of the original program features as possible. Three come to mind. First, care has been taken to insure that the "reset handler" functions properly. Second, both 48K and 64K modes have been preserved. And third, the "Save Mode Library" option has been retained. The only change in the operation of the program becomes apparent when saving the "User Mode Library". The A2-FS2 manual states that the disk on which the library is saved does not need to be initialized. When using the softkeyed version, however, the disk MUST be initialized beforehand since the simulator no longer performs this function.

User Mode Libraries already saved to disk with the original Flight Simulator DOS may be converted to DOS 3.3 format using the technique outlined in step 9 of this softkey. Since newer versions of Flight Simulator][perform more functions, they use up more disk space. In addition, more data may be stored in the language card. Finding space to put the language card data could become a big problem. There are numerous solutions, of course. A disk with 36, or even 40 tracks could be created, but then would not be "truly" COPYAable. If the language card data grows too large, it could be moved to the back side of the disk, thus facilitating a boot which would require flipping the disk. A more sensible(?) approach, would be to "scrunch" the hi-res picture which presently uses 2 whole tracks. It could then be loaded and unscrunched during the boot process. The newly acquisitioned disk space, of course, could then be used to hold "language card data."

Hexdumps

COPY

```
1000: 20 2F FB 20 40 FB 20 58   $6B0D
1008: FC 2C 57 C0 A9 BF 20 ED   $30FA
1010: FD 20 0C FD C9 B3 B0 F9   $D1EF
1018: C9 B1 90 F5 29 03 8D EA   $2A04
1020: B7 4A 85 FF AE E9 B7 8E   $8FE4
1028: 08 1E 9D 8A C0 9D 89 C0   $5ECE
1030: A9 60 20 95 BE A9 00 20   $2DCB
1038: 5A BE 20 F9 1E 20 C0 10   $CE50
1040: A0 02 8C F4 B7 A0 00 8C   $EFF4
1048: F0 B7 8C 0A 1E 8C 01 1E   $1518

1050: 20 58 FC A5 FF D0 05 A9   $609E
1058: D3 20 B7 10 BD 8A C0 20   $6620
1060: AC 1F 20 74 20 20 C0 10   $E577
1068: AD 0B 1E 4A 8D EC B7 20   $D402
1070: DA FD A5 FF D0 0B 20 48   $9EC3
1078: F9 A9 D4 20 B7 10 EE 78   $65B7
1080: 04 A9 35 A2 01 A0 0F 8D   $8089
1088: F1 B7 8E F8 B7 8C ED B7   $8216
1090: 20 AB 10 CE F1 B7 AC ED   $6B20
1098: B7 88 10 F1 AC 01 1E C8   $4DBD

10A0: C8 C8 C8 C0 88 90 A6 20   $F6DF
10A8: 2F FB 60 A0 00 8C EB B7   $D46F
```

```
10B0: A0 E8 A9 B7 4C B5 B7 20   $AF7E
10B8: ED FD 20 0C FD 4C 58 FC   $3AC0
10C0: AE E9 B7 9D 88 C0 60      $2B5E
```

DOS LOADER

```
1F06: A0 00                     $B033
1F08: B9 00 00 99 00 36 B9 00   $D7E8
1F10: 01 99 00 37 B9 00 03 99   $8F72
1F18: 00 38 B9 00 08 99 00 39   $09E1
1F20: 88 D0 E5 A9 49 85 00 A9   $D01D
1F28: 23 85 01 AD 08 1E 4A 4A   $4C93
1F30: 4A 4A 09 C0 8D 39 1F 20   $DDFF
1F38: 00 C6 A0 00 B9 00 36 99   $17E3
1F40: 00 00 B9 00 37 99 00 01   $321D
1F48: B9 00 38 99 00 03 B9 00   $FC6E

1F50: 39 99 00 08 88 D0 E5 A9   $FCFC
1F58: 00 60                     $14B4
```

NEW DOS

```
23D0: A9 00 A2 35 A0 0F 85 A9   $6892
23D8: 86 AA 84 AB AD 0B 1E AE   $F085
23E0: 08 1E 4A 85 AC BD 8E C0   $5058
23E8: 20 FD 23 B0 0F 20 43 24   $A277
23F0: B0 F6 AE 08 1E C6 AA C6   $A988
23F8: AB 10 ED 18 60 BD 8C C0   $0B5C
2400: 10 FB C9 D5 D0 F7 BD 8C   $98CA
2408: C0 10 FB C9 AA D0 F3 BD   $D95C
2410: 8C C0 10 FB C9 96 D0 EA   $AEDA
2418: A0 03 85 B8 BD 8C C0 10   $B9D7

2420: FB 2A 85 B7 BD 8C C0 10   $090E
2428: FB 25 B7 88 D0 EC 85 B7   $E348
2430: A5 B8 C5 AC D0 0B A4 AB   $39D8
2438: B9 A7 24 C5 B7 D0 BE 18   $F309
2440: 60 38 60 BD 8C C0 10 FB   $0796
2448: C9 D5 D0 F7 BD 8C C0 10   $2753
2450: FB C9 AA D0 F3 BD 8C C0   $D0B3
2458: 10 FB C9 AD D0 EA A9 00   $90FF
2460: A0 56 84 B7 BC 8C C0 10   $E7B9
2468: FB 59 00 3F A4 B7 88 99   $0A0C

2470: 00 36 D0 EE 84 B7 BC 8C   $5173
2478: C0 10 FB 59 00 3F A4 B7   $B30E
2480: 91 A9 C8 D0 EF BC 8C C0   $13FE
2488: 10 FB D9 00 3F D0 B2 A0   $724B
2490: 00 A2 56 CA 30 FB B1 A9   $6503
2498: 5E 00 36 2A 5E 00 36 2A   $35EF
24A0: 91 A9 C8 D0 EE 18 60 00   $03DC
24A8: 0D 0B 09 07 05 03 01 0E   $7244
24B0: 0C 0A 08 06 04 02 0F      $3CD5
```

LC DOS

```
23D0: A9 00 A2 35 A0 0F 85 A9   $6892
23D8: 86 AA 84 AB AD 0B 1E AE   $F085
23E0: 08 1E 4A 85 AC BD 8E C0   $5058
23E8: 20 E2 D7 B0 0F 20 28 D8   $36CB
23F0: B0 F6 AE 08 1E C6 AA C6   $7DF4
23F8: AB 10 ED 18 60 BD 8C C0   $9FE0
2400: 10 FB C9 D5 D0 F7 BD 8C   $4CB6
2408: C0 10 FB C9 AA D0 F3 BD   $4DE0
2410: 8C C0 10 FB C9 96 D0 EA   $7AA6
2418: A0 03 85 B8 BD 8C C0 10   $2D6B

2420: FB 2A 85 B7 BD 8C C0 10   $DD72
2428: FB 25 B7 88 D0 EC 85 B7   $77F4
2430: A5 B8 C5 AC D0 0B A4 AB   $EDA4
2438: B9 8C D8 C5 B7 D0 BE 18   $A8AE
2440: 60 38 60 BD 8C C0 10 FB   $AC01
2448: C9 D5 D0 F7 BD 8C C0 10   $7CF4
2450: FB C9 AA D0 F3 BD 8C C0   $7B24
2458: 10 FB C9 AD D0 EA A9 00   $CB58
2460: A0 56 84 B7 BC 8C C0 10   $4C2E
2468: FB 59 00 3F A4 B7 88 99   $51AB

2470: 00 36 D0 EE 84 B7 BC 8C   $FAE4
2478: C0 10 FB 59 00 3F A4 B7   $E8A9
2480: 91 A9 C8 D0 EF BC 8C C0   $B869
2488: 10 FB D9 00 3F D0 B2 A0   $29EC
2490: 00 A2 56 CA 30 FB B1 A9   $CE94
2498: 5E 00 36 2A 5E 00 36 2A   $6E48
24A0: 91 A9 C8 D0 EE 18 60 00   $A84B
24A8: 0D 0B 09 07 05 03 01 0E   $29E3
24B0: 0C 0A 08 06 04 02 0F      $AB50
```

WRITE

```
21E3: AD 0B 1E A2 00            $4E0D
21E8: A0 0F 4A 85 AC 86 A8 84   $3D95
21F0: AB 20 0F 22 AE 08 1E 20   $37D8
21F8: FD 23 B0 12 A0 03 BD 8C   $979B
2200: C0 10 FB 88 10 F8 20 40   $5874
2208: 22 C6 AB 10 E4 18 60 A5   $C10E
2210: AB 18 69 26 85 A9 A2 00   $96D0
2218: A0 02 88 B1 A8 4A 3E 00   $0816
2220: 37 4A 3E 00 37 99 00 36   $5074
2228: E8 E0 56 90 ED A2 00 98   $7976

2230: D0 E8 A2 55 BD 00 37 29   $D35C
2238: 3F 9D 00 37 CA 10 F5 60   $8FCB
2240: 38 AE 08 1E 86 AA BD 8D   $A775
2248: C0 BD 8E C0 10 03 4C D0   $5A14
2250: 22 AD 00 37 85 A9 A9 FF   $174A
2258: 9D 8F C0 1D 8C C0 48 68   $93DC
2260: EA A0 04 48 68 20 D2 22   $A7CF
2268: 88 D0 F8 A9 D5 20 D1 22   $1066
2270: A9 AA 20 D1 22 A9 AD 20   $1DC1
2278: D1 22 98 A0 56 D0 03 B9   $00C6

2280: 00 37 59 FF 36 AA BD 55   $0A97
2288: 1E A6 AA 9D 8D C0 BD 8C   $87BE
2290: C0 88 D0 EB A5 A9 EA 59   $8F26
2298: 00 36 AA BD 55 1E AE 08   $4225
22A0: 1E 9D 8D C0 BD 8C C0 B9   $EF37
22A8: 00 36 C8 D0 EA AA BD 55   $0352
22B0: 1E A6 AA 20 D4 22 A9 DE   $1BC8
22B8: 20 D1 22 A9 AA 20 D1 22   $6F74
22C0: A9 EB 20 D1 22 A9 FF 20   $8EA3
22C8: D1 22 BD 8E C0 BD 8C C0   $7FC9

22D0: 60 18 48 68 9D 8D C0 1D   $25DE
22D8: 8C C0 60                  $BEA3
```

LC WRITE

```
21E3: AD 0B 1E A2 00            $4E0D
21F8: A0 0F 4A 85 AC 86 A8 84   $3D95
21F0: AB 20 F4 D5 AE 08 1E 20   $7BB8
21F8: E2 D7 B0 12 A0 03 BD 8C   $A470
2200: C0 10 FB 88 10 F8 20 25   $7E5D
2208: D6 C6 AB 10 E4 18 60 A5   $DDF7
2210: AB 18 69 26 85 A9 A2 00   $CAF9
2218: A0 02 88 B1 A8 4A 3E 00   $14EF
2220: 37 4A 3E 00 37 99 00 36   $0C5D
2228: E8 E0 56 90 ED A2 00 98   $658F

2230: D0 E8 A2 55 BD 00 37 29   $8F75
2238: 3F 9D 00 37 CA 10 F5 60   $9332
2240: 38 AE 08 1E 86 AA BD 8D   $FB5C
2248: C0 BD 8E C0 10 03 4C B5   $23DF
2250: D6 AD 00 37 85 A9 A9 FF   $6471
2258: 9D 8F C0 1D 8C C0 48 68   $9017
2260: EA A0 04 48 68 20 B7 D6   $D8EB
2268: 88 D0 F8 A9 D5 20 B6 D6   $569F
2270: A9 AA 20 B6 D6 A9 AD 20   $1AFF
2278: B6 D6 98 A0 56 D0 03 B9   $F493

2280: 00 37 59 FF 36 AA BD 55   $3ED2
2288: 1E A6 AA 9D 8D C0 BD 8C   $73EB
2290: C0 88 D0 EB A5 A9 EA 59   $BB63
2298: 00 36 AA BD 55 1E AE 08   $B670
22A0: 1E 9D 8D C0 BD 8C C0 B9   $DB72
22A8: 00 36 C8 D0 EA AA BD 55   $F707
22B0: 1E A6 AA 20 B9 D6 A9 DE   $79AA
22B8: 20 B6 D6 A9 AA 20 B6 D6   $2FFE
22C0: A9 EB 20 B6 D6 A9 FF 20   $EF1E
22C8: B6 D6 BD 8E C0 BD 8C C0   $8DEF

22D0: 60 18 48 68 9D 8D C0 1D   $7718
22D8: 8C C0 60                  $F035
```

LC LOADER

```
20AF: A2                        $DC7E
20B0: 00 8E 03 1E BD FE 20 F0   $8733
20B8: 44 8D 04 1E BD 01 21 8D   $2CD0
20C0: 01 1E 20 10 21 AE 03 1E   $15B4
20C8: BD 03 21 18 69 26 85 A6   $1AB3
20D0: BD 05 21 85 A8 A0 00 84   $F637
20D8: A5 84 A7 B1 A5 91 A7 88   $224B
20E0: D0 F9 DE 05 21 DE 03 21   $B3FC
20E8: 10 08 A9 0F 9D 03 21 CE   $0992
20F0: 01 1E CE 04 1E D0 CB AE   $94F4

20F8: 03 1E E8 D0 B4 60 04 21   $584E
2100: 00 21 21 0F 0B F9 F3      $BE15
```

BOOT1

```
0800: 01 A9 60 8D 01 08 A6 2B   $A41B
0808: 8A 4A 4A 4A 4A 09 C0 8D   $C76D
0810: 5C 08 A9 6A 45 00 45 01   $6EE5
0818: A8 F0 1E 2C 82 C0 20 2F   $6302
0820: FB 20 58 FC A0 40 84 E6   $928A
0828: 20 F2 F3 2C 50 C0 2C 52   $D14F
0830: C0 2C 55 C0 2C 57 C0 A0   $BA4D
0838: 01 B9 7C 08 85 51 B9 7E   $6AC3
0840: 08 85 50 B9 80 08 8D 64   $02EC
0848: 08 B9 82 08 8D 69 08 A4   $0DED

0850: 51 B9 6C 08 85 3D A5 50   $E516
0858: 85 27 20 5C 00 C6 50 C6   $9C6C
0860: 51 A5 51 C9 06 D0 E8 A6   $8FBF
0868: 2B 4C 00 1D 00 0D 0B 09   $5CD5
0870: 07 05 03 01 0E 0C 0A 08   $1A83
0878: 06 04 02 0F 06 09 25 1F   $BE34
0880: 00 06 60 4C               $A25D
```

AutoDuel

by Charles Taylor

Origin Systems, Inc
340 Harvey Road
Manchester, NH 03103

Requirements:
Super IOB v1.5
Six blank disk sides
Sector Editor
Apple][with 64K

Autoduel is another fine arcade-adventure-fantasy game from Chuckles and Lord British. Unlike the Ultima series, your arcade skills will get somewhat of a workout on these games. Moebius will be especially tough, because of unusual key commands for fighting and movement. A joystick is not even an option on this one.

Most of the credit for this softkey goes to Mr. Roetman and his fine softkey for Ultima IV (see COMPUTIST No. 28). After correcting a typo in the controller, I copied this program with it, made a few sector edits and played the game. Unfortunately, I screwed up the 2-disk drive option in the process. This should not be too much of a hardship as almost the entire game is played with Side B after Side A is booted. As with most multi-disk games, only the boot sides are copy protected.

The Procedure

1) Boot your system master and tell DOS that it is to BRUN the greeting program.

PR#6
POKE 40514,52

2) Put in a blank disk and initialize it with AutoDuel's boot filename.

INIT B□A T

3) Install the controller at the end of this article into Super IOB and use it to copy side A of AutoDuel to the disk you formatted in step 2.

4) Make the following sector edits to the disk created in step 3.

```
Track  Sector  Bytes A5 - BE

From 8D 5D BD A9 9B 8D 2C BF D0 0C A9 B5
To   EA EA EA A9 9B EA EA EA D0 0C A9 AD

From 85 4E 8D 5D BD A9 D5 8D 2C BF A9 E8
To   85 4E EA EA EA A9 9B EA EA EA A9 B7

From A0 B7
To   A0 E8
```

Write the sectors back out and you're done!

controller

```
1000 REM AUTODUEL/ULTIMA 4
1010 TK = 3 :LT = 4 :ST = 15 :LS = 15 :CD = WR
1020 POKE 47405 ,24 : POKE 47406 ,96 : POKE 47497
     ,24 : POKE 47498 ,96
1030 POKE 47829 ,3 :T1 = TK : GOSUB 490 : GOSUB 210
1040 GOSUB 190 : GOSUB 610
1050 TK = TK + 1 :LT = LT + 1 : IF PEEK (BUF ) < MB
     AND TK < 35 THEN 1040
1060 POKE 47405 ,208 : POKE 47406 ,19 : POKE
     47497 ,208 : POKE 47498 ,183 : POKE 47829
     ,213 : GOSUB 230
1070 TK = T1 :LT = 35 : GOSUB 490 : GOSUB 610 : IF
     PEEK (TRK ) = LT THEN 1090
1080 TK = PEEK (TRK ) :ST = PEEK (SCT ) :LT = TK
     + 1 : GOTO 1020
1090 HOME : PRINT "COPYDONE.^ DOS^ NOT^
     COPIED." : END
5000 DATA 213 ,170 ,181
5010 DATA 215 ,170 ,151
5020 DATA 213 ,170 ,150
5030 DATA 213 ,170 ,151
5040 DATA 215 ,170 ,150
5050 DATA 215 ,170 ,151
5060 DATA 221 ,170 ,158
5070 DATA 221 ,170 ,159
5080 DATA 213 ,170 ,181
5090 DATA 223 ,170 ,158
5100 DATA 223 ,170 ,159
5110 DATA 221 ,170 ,158
5120 DATA 221 ,170 ,159
5130 DATA 223 ,170 ,158
5140 DATA 223 ,170 ,159
5150 DATA 213 ,170 ,150
5160 DATA 213 ,170 ,181
5170 DATA 213 ,170 ,151
5180 DATA 215 ,170 ,150
5190 DATA 215 ,170 ,151
5200 DATA 213 ,170 ,150
5210 DATA 213 ,170 ,151
5220 DATA 215 ,170 ,150
5230 DATA 215 ,170 ,151
5240 DATA 213 ,170 ,181
5250 DATA 221 ,170 ,158
5260 DATA 221 ,170 ,159
5270 DATA 223 ,170 ,158
5280 DATA 223 ,170 ,159
5290 DATA 221 ,170 ,158
5300 DATA 221 ,170 ,159
5310 DATA 223 ,170 ,158
5320 DATA 213 ,170 ,181
5330 DATA 223 ,170 ,159
5340 DATA 245 ,170 ,182
5350 DATA 245 ,170 ,183
5360 DATA 247 ,170 ,182
```

controller checksums

```
1000 - $356B     5140 - $59D2
1010 - $3189     5150 - $8F2A
1020 - $C562     5160 - $AFA5
1030 - $545E     5170 - $8041
1040 - $DDB4     5180 - $0796
1050 - $A5C8     5190 - $1269
1060 - $044B     5200 - $95A5
1070 - $B732     5210 - $8E5D
1080 - $045C     5220 - $1DBE
1090 - $0EB7     5230 - $4CFD
5000 - $47E9     5240 - $EAF8
5010 - $C5B9     5250 - $3B74
5020 - $8750     5260 - $F464
5030 - $0BA3     5270 - $35E1
5040 - $0D61     5280 - $E4C4
5050 - $8BAD     5290 - $598C
5060 - $D0F6     5300 - $7A80
5070 - $397D     5310 - $67A1
5080 - $CC35     5320 - $2132
5090 - $96F7     5330 - $6F01
5100 - $4E87     5340 - $0EF1
5110 - $58CC     5350 - $9339
5120 - $A176     5360 - $B6AC
5130 - $0A8D
```

```
4000: 20 E2 F3 8D 52 C0 A9 00   $5991
4008: 85 FF 85 FD A9 00 85 E4   $B390
4010: A6 FF 20 9D 40 EE EF 40   $9927
4018: AE EF 40 E0 05 90 05 A2   $F7AB
4020: 00 8E EF 40 BD EA 40 85   $A141
4028: E4 A6 FF AD F4 40 9D C2   $B3D3
4030: 40 AD F6 40 9D CC 40 AD   $5A9C
4038: F5 40 9D D6 40 AD F7 40   $DAC8
4040: 9D E0 40 E6 FF A5 FF C9   $54D8
4048: 0A 90 04 A9 00 85 FF 20   $2ABF

4050: 9D 40 A2 03 BD F4 40 18   $B44A
4058: 7D F0 40 9D F4 40 DD F8   $5DCC
4060: 40 90 0C BD F0 40 49 FF   $3310
4068: 69 00 9D F0 40 90 E5 CA   $5978
4070: 10 E2 AD 00 C0 30 03 4C   $7E7D
4078: 0C 40 C9 9B D0 09 8D 10   $70E2
4080: C0 8D 51 C0 4C 58 FC 29   $1A27
4088: 0F A6 FD 9D F0 40 E8 E0   $24AA
4090: 04 D0 02 A2 00 86 FD 8D   $04B1
4098: 10 C0 4C 0C 40 BD D6 40   $94A1

40A0: 48 A0 00 BD C2 40 0A 90   $2D85
40A8: 01 C8 86 FE AA 68 20 57   $E6B2
40B0: F4 A6 FE BC E0 40 BD CC   $B792
40B8: 40 A2 00 0A 90 01 E8 4C   $087E
40C0: 3A F5 00 00 00 00 00 00   $54E6
40C8: 00 00 00 00 00 00 00 00   $1406
40D0: 00 00 00 00 00 00 00 00   $54E6
40D8: 00 00 00 00 00 00 00 00   $1406
40E0: 00 00 00 00 00 00 00 00   $54E6
40E8: 00 00 7F 55 2A D5 AA 00   $7D65

40F0: 03 05 07 09 00 00 00 00   $A18B
40F8: 8C C0 8C C0               $CD93
```

BSAVE LINES, A$4000, L$FB

Coming soon...

(to a magazine near you)

COMPUTIST No. 37:

Softkeys for: Under Fire, Pegasus][, Take 1 (revisited), Magic Slate, Alter Ego, Rendezvous, Quicken, Story Tree, Assembly Language Tutor, Avalon Hill games, Dark Crystal.

Features: Playing Karateka on a //c, Track Finder, Sylk to Dif.

Core: Breaking In: tips for beginners, The DOS Alterer.

Review: Copy][Plus, 6.0.

COMPUTIST No. 38:

Softkeys for: Cyclod, Alternate Reality, Boulder Dash I & II, Hard Hat Mack (Revisited), The Other Side, F-15 Strike Eagle, Championship Lode Runner, Gato V1.3, I; Damiano, Wilderness, Golf's Best.

Features: The Enhanced/Unenhanced //e, Looking into Flight Simulator's DOS.

Core: Applesoft Variable Extractor, Installing a RAM disk into DOS 3.3.

Description of Available Back Issues

35 *Softkeys* | Hi-res Cribbage | Olympic Decathlon | Revisiting F-15 Strike Eagle | Masquerade | The Hobbit | *Readers' Softkeys* | Pooyan | The Perfect Score | Alice in Wonderland | The Money Manager | Good Thinking | Rescue Raiders | *Feature* | Putting a New F8 on Your Language Card | *Core* | Exploring ProDOS by installng a CPS Clock Driver | .....

34 *Softkeys* | Crisis Mountain | Terripin Logo | Apple Logo II | Fishies 1.0 | SpellWorks | Gumball | *Readers' Softkeys* | Rescue at Rigel | Crazey Mazey | Conan | Perry Mason: The Case of the Mandarin Murder | Koronis Rift | *Feature* | More ROM Running | *Core* | Infocom Revealed | .....

33 *Softkeys* | Word Juggler | Tink! Tonk! | Sundog v2.0 | G.I. Joe & Lucas Film's Eidolon | Summer Games II | Thief | Instant Pascal | World's Greatest Football Game | *Readers' Softkeys* | Graphic Adventure #1 | Sensible Grammar & Extended Bookends | Chipwits | Hardball | King's Quest II | The World's Greatest Baseball Game | *Feature* | How to be the Sound Master | *Core* | The Mapping of Ultima IV |

32 *Softkeys* | Revisiting Music Construction Set | Cubit | Baudville Software | Hartley Software | Bridge | Early Games for Young Children | Tawala's Last Redoubt | *Readers' Softkeys* | Print Shop Companion | Kracking Vol II | Moebius | Mouse Budget, Mouse Word & Mouse Desk | Adventure Construction Set | *Feature* | Using Data Disks With Microzines | *Core* | Super IOB v1.5 a Reprint | .....

31 *Softkeys* | Trivia Fever | The Original Boston Computer Diet | Lifesaver | Synergistic Software | Blazing Paddles | Zardax | *Readers' Softkeys* | Time Zone | Tycoon | Earthly Delights | Jingle Disk | Crystal Caverns | Karate Champ | *Feature* | A Little Help With The Bard's Tale | *Core* | Black Box | Unrestricted Ampersand | .....

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 | *Feature* | 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 | *Feature* | 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 | *Feature* | 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 | *Feature* | Essential Data Duplicator copy parms | *Core* | Direct Sector Access From DOS | ..

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 | .....

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 | *Features* | 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 | .....

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 | ....

Hardcore Computing 3 ..... HyperDOS Creator | Menu Hello | Zyphyr Wars | Vector Graphics | Review of Bit Copiers | Boot Code Tracing | Softkey IOB | Interview with 'Mike' Markkula | .....

For special savings, order our 'Core Special' and receive all three CORE magazines for only $10.00

Issue	Mag $4.75	Disk $9.95	Both $12.95
36	☐	☐	☐
35	☐	☐	☐
34	☐	☐	☐
33	☐	☐	☐
32	☐	☐	☐
31	☐	☐	☐
30	☐	☐	☐
29	☐	☐	☐
28	☐	☐	☐
27	☐	☐	☐
26	☐	☐	☐
25	☐	☐	☐
24	☐	☐	☐
23	NA	☐	NA
22	☐	☐	☐
21	NA	☐	NA
20	☐	☐	☐
19	☐	☐	☐
18	NA	☐	NA
☆ 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	☐	☐	☐
Core 3	☐	☐	☐
Computing 3	☐	NA	NA
Best of Hardcore Computing	NA	☐	NA
Core Special $10.00 (All three CORE magazines)			☐

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 and LIBRARY DISKS

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.

LIBRARY DISKS are perfect companions for COMPUTIST

Documentation for each 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 _____-_____-_____-_____ Exp. _____

Signature _____ CP36

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

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.

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