

For The Serious User Of Apple][Computers

COMPUTIST

Issue No. 37

November 1986

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

Softkeys For:

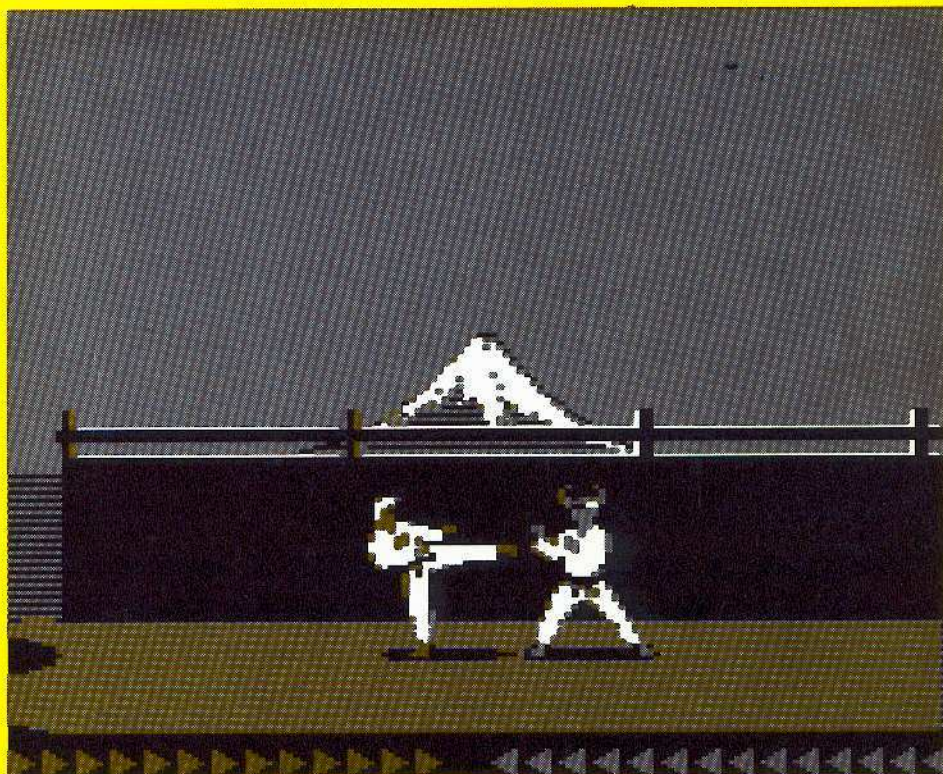
Under Fire
Pegasus][
Alter Ego
Dark Crystal
Magic Slate
Quicken
Story Tree
Rendezvous

Core:

Breaking In:
deprotection tips
for beginners

Feature:

The DOS Alterer



(Page 10)

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 **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 II 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 (from the Book of Softkeys vol I) or ZAP from Bag of Tricks.
- 3) **Disk Search Utility** such as The Inspector or The CIA or The CORE Disk Searcher (from the Book of Softkeys vol II).
- 4) **Assembler** such as the S-C Assembler from S-C software or Merlin/Big Mac.
- 5) **Bit Copy Program** such as Copy II Plus, Locksmith or The Essential Data Duplicator
- 6) **Text Editor** capable of producing normal sequential text files such as Appewriter II, Magic Window II or Screenwriter II.

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 II 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 II 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 II 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,AS6000

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 II and Apple II 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.

1000: A9 80 8D A9 AD A9 02 8D \$66E3
 1008: AA AD A9 88 8D 23 AE A9 \$35B6
 1010: 02 8D 24 AE AD 50 C0 AD \$B266
 1018: 56 C0 AD 52 C0 AD 54 C0 \$B83A
 1020: A2 00 8A A0 00 99 00 04 \$ADDC
 1028: 99 00 05 99 00 06 99 00 \$DABE
 1030: 07 C8 D0 F1 E8 D0 EB A0 \$2338
 1038: 00 B9 47 10 99 80 02 C8 \$1A5E
 1040: C0 67 D0 F5 4C 2F FB A9 \$83E3
 1048: 00 8D E6 02 4C 2F AE AC \$C705

1050: E6 02 30 2A B9 CA 02 AA \$8526
 1058: C8 B9 CA 02 F0 23 A8 88 \$AEFC
 1060: C4 FF C0 00 D0 F9 2C 30 \$6E6B
 1068: C0 CA D0 F2 EE E6 02 EE \$FCC1
 1070: E6 02 AD E6 02 C9 1C D0 \$7B3C
 1078: 05 A9 FF 8D E6 02 4C 2F \$5AE0
 1080: AE A9 80 A8 88 C4 FF C0 \$E1C7
 1088: 00 D0 F9 CA D0 F5 4C A5 \$431D
 1090: 02 80 BF 80 BF 80 AA 80 \$F591
 1098: 98 80 BF 80 98 C0 AA 40 \$D900

10A0: 00 80 BF 80 BF 80 AA 80 \$9E9C
 10A8: 98 00 BF C0 CA FF \$9FE0

BSAVE IT, A\$1000, L\$AE

FP

BRUN IT

BACKUP

ESSENTIAL DATA DUPLICATOR

Back up your copy-protected disks with **Essential Data Duplicator 4 PLUS.**

EDD 4 PLUS is new technology, not just "another" copy program. The **EDD 4 PLUS** program uses a specially designed hardware card which works with your disk drives to back up disks by accurately copying the bits of data from each track. Don't be fooled.

no other copy-program/system for Apples can do this! In addition to backing up disks, **EDD 4 PLUS** includes several useful utilities such as examining disk drives, certifying disks, displaying drive speed rpm's, plus more!

EDD 4 PLUS runs on Apple II, II Plus (including most compatibles), and IIe, and is priced at \$129.95 (duodisk/uni-disk 5.25 owners must add \$15 for a special cable adapter). Add \$5.00 (\$8.00 foreign) ship / handling

when ordering direct. A standard **EDD 4** version which doesn't include any hardware is available, and can be used on Apple IIc and III (using emulations mode) and

is priced at \$79.95. Add \$3.00 (\$6.00 foreign) ship/handling when ordering direct. If you own an earlier version, send us your **EDD** disk and deduct \$50 from your order. Ask for **EDD 4 PLUS** at your local computer store, or order direct.

Mastercard and Visa accepted. All orders must be prepaid. In addition, registered owners may purchase **EDD's** printed 6502 SOURCE CODE listing for educational purposes.

UTILICO MICROWARE

3377 SOLANO AVE., SUITE 352
 NAPA, CA 94558 / (707) 257-2420

WARNING: EDD is sold for making archival copies only



BACKUP YOUR SOFTWARE WITH LOCKSMITH 6.0™.

Locksmith, the controversial copy program that took the Apple world by storm in 1981, has evolved from a powerful bit-copy programmed into a complete disk utility system, allowing the Apple user to recover crashed disks, restore accidentally deleted files, and perform hardware diagnostics on the disk drive and memory boards. The NEW Locksmith version 6.0 is now available and includes an advanced disk recovery utility, a framing-bit analyzer, an automatic boot tracer, a sector editor, many file utilities, and of course, the most powerful bit-copy program available. A fast disk backup utility copies disks in eight seconds flat. Improvements to Locksmith Programming Language have made it more powerful and easier to use for you to write your own backup and repair procedures. Includes a library disk which contains automatic procedures to copy hundreds of Apple programs.

Locksmith requires no additional hardware, but will use any additional RAM memory that it finds, including RAM boards from Applied Engineering and Checkmate Technology.

Don't get caught with your hands tied. Order Locksmith 6.0 today.

Does copy protection have your hands tied?



NEW LOW PRICE \$79.95
 Registered Locksmith 5.0 owners may upgrade to version 6.0 for **\$29.95**.
 Available from your computer dealer or directly from:



Alpha Logic Business Systems, Inc.
 4119 North Union Road
 Woodstock, IL 60098
 (815) 568-5166



© Alpha Logic Business Systems, Inc. 1985
 Locksmith and Locksmith PC are registered trademarks of Alpha Logic Business Systems, Inc.

announcing new rates!

YES! COMPUTIST has DROPPED its annual subscription rate.

U.S. Domestic save \$8 per year
U.S. First Class save \$3 per year
Canada and Mexico save \$23 per year
All other foreign save \$45 per year

Additionally, COMPUTIST has incorporated a combination library disk and first class subscription rate to save you even more.

With this new 'COMBO' subscription, you will receive each monthly issue AND it's corresponding library disk for as much as 43% off the individual rate. Combination subscriptions are sent U.S. First Class mail.

If you have at least 3 issues left on your current subscription, you can upgrade to this special offer.

Yes I want to take advantage of the big money saving offer and subscribe to your fine publication. Enclosed are U.S. Funds for a 12 issue subscription.

New Subscriber Please renew my subscription

U.S. \$32 U.S. First Class/Canada/Mexico \$45 Other Foreign \$75

Combination magazine and corresponding disk subscriptions

U.S./Canada/Mexico \$100 Other Foreign \$140

To upgrade your subscription to a combo subscription, U.S./Canada/Mexico send \$5.50 and other Foreign send \$6.50 per remaining issue. You must have at least 3 issues remaining to take advantage of this upgrade offer.

Name _____ ID# _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone _____

  _____ Exp. _____

Signature _____ CP37

U.S. Funds drawn on U.S. bank. Subscription will not commence until funds are received. Send orders to: COMPUTIST PO Box 110846-T Tacoma, WA 98411

Big Deal! We really mean it. This is truly a big deal. We want to sell you a book or two. Need we say more?

The Book Of Softkeys

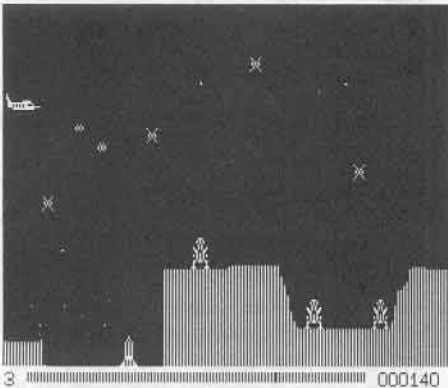
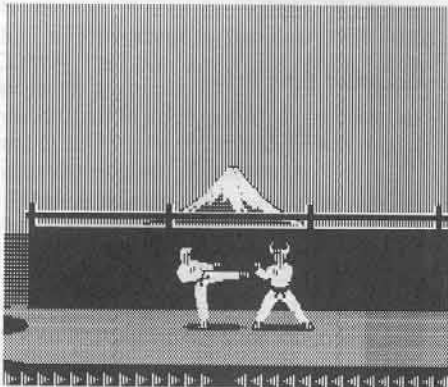
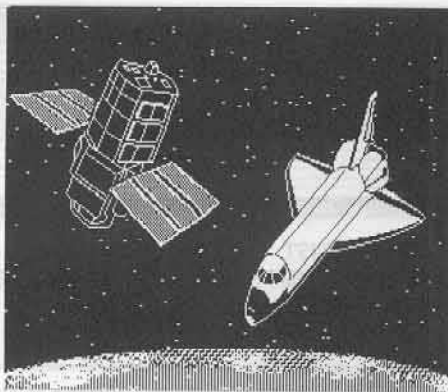
Volume I (\$12.95)

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

Volume II (\$17.95)

contains softkeys for: Apple Cider Spider | Apple Logo | Arcade Machine | The Artist | Bank Street Writer | Cannonball Blitz | Canyon Climber | Caverns of Freitag | Crush, Crumble & Chomp | Data Factory 5.0 | DB Master | The Dic*tion*ary | Essential Data Duplicator I & III | Gold Rush | Krell Logo | Legacy of Llylgamyn | Mask Of The Sun | Minit Man | Mouskattack | Music Construction Set | Oil's Well | Pandora's Box | Robotron | Sammy Lightfoot | Screenwriter II v2.2 | Sensible Speller 4.0, 4.0c, 4.1c | the Spy Strikes Back | Time Zone v1.1 | Visible Computer: 6502 | Visidex | Visiterm | Zaxxon | Hayden Software | Sierra Online Software | PLUS the complete listing of the ultimate cracking program...Super IOB 1.5 | and more!

To Order: Send \$17.95 + Shipping and Handling for Volume II and/or \$12.95 + S&H for Volume I. Shipping and handling is \$2.00 per book for US orders, \$5.00 per book for foreign orders. U.S. funds drawn on U.S. banks only. Washington State orders add 7.8% sales tax. Send your orders to: SoftKey Publishing, PO Box 110937-BK, Tacoma, WA 98411



This month's cover:

Graphics from Broderbund's "Karateka."

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.

Entire contents copyright 1986 by SoftKey Publishing. All rights reserved. Copying done for other than personal or internal reference (without express written permission from the publisher) is prohibited.

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

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

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

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

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

COMPUTIST

Issue 37

November 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:

11 Under Fire

by John Howard

12 Pegasus II

by Ed Croft

14 Take 1 (revisited)

by J. J. Gifford

27 Flight simulator II v1.05 (part 2)

This month we present the source code to the files used to deprotect Flight Simulator II v1.05 in COMPUTIST No. 36. by Eric Sunshine

features:

10 Playing Karateka on a //c

by Bill Hamshire

15 Track Finder

The Track Finder is a nifty little utility that tells you over what head the disk is currently located. This is particularly handy when trying to figure out what track(s) to nibble copy.

by Denny Colt

20 Sylk to Dif

The direct result of a letter to COMPUTIST, this program allows your Multiplan disk to read DIF files, the standard of data transfer.

by D. W. Walkey

core:

16 Breaking In: tips for beginners

If you are confused by the procedures used in softkeys and mystified by how the author could ever figure out how to make the disk copyable, then this article is for you.

by Dave Templin

18 Copy II Plus 6.0: a review

by Jerry D. Greer

22 The DOS Alterer

This program helps you wrap DOS around your little finger by modifying it to read several of the protected disk formats out there.

by J. J. Gifford

departments:

4 Input

6 Most Wanted List

7 Readers' Softkey & Copy Exchange

Softkeys for: Sunburst Education's *Magic Slate* by Glen Tatum, Activision's *Alter Ego* by Charles Taylor, Peachtree Software's *Rendezvous* by John Howard, Intuit's *Quicken* by Greg Robinson, Scholastic's *Story Tree* by Stanley Planton, Prentice Hall's *Assembly Language Tutor* by B. Croome, Avalon Hill games by John Howard, Sierra On Line's *Dark Crystal* by Andrew Swartz

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

COMPUTIST is an Asset

On my copy of "Halley Project" by Activision, the softkey in COMPUTIST No. 31 worked only partially. Except for Track 0, Sectors 4 through \$0D and Track 1, Sectors 0 through 6, the softkey worked. I used a sector editor under normal DOS 3.3 and read each sector from the master copy into memory and then wrote them to my backup copy. Other than that, it works fine.

"Video Vegas" by Activision is a set of four (Keno, Twenty-One, Poker and Slot Machine) gambling programs. Very well written in graphics and sound. Each of the programs can be listed to see what they've done, but it uses a nibble count routine to prevent making backup copies.

Rather than re-creating a controller, use Clay Harrell's softkey for "Take 1" from COMPUTIST No. 25. The same JSR DC44 is used, but on my copy it was on Track 0, Sectors 1 and 2. I changed the BNE instructions at address 16 and 17 from D0 F7 to D0 00.

It pays to have back issues of COMPUTIST.

Jeanne Edwards
Clinton, MD

Another Ghostbuster

It seems that the good folks at Activision decided to modify the protection that they used on Space Shuttle (softkey in COMPUTIST No. 30) and used it on another one of their releases: Ghostbusters. Instead of modifying the data prolog's last byte on tracks 4-6, the data epilogs have been changed (for the same function also) and the tracks are different. Follow the procedure listed below to get your Ghostbusters original "busted."

Requirements:

Ghostbusters original
A blank disk
Disk searcher
Sector editor
COPYA from the DOS 3.3 System Master
A desire to be a DISKbuster

- 1) Boot up a DOS 3.3 disk.
- 2) POKE 47426,24
- 3) RUN COPYA and copy the original onto the blank.
- 4) Get out your disk searcher and search for the sequence A0 00 98. On my disk this was on Track \$00, Sector \$0B, Byte \$00.
- 5) Replace these bytes with 20 E0 50.
- 6) In the same sector, enter these bytes starting at Byte \$E0.

```
xxE0: A9 AB 85 00 A9 BC 85 01  
xxE8: A9 CD 85 02 A0 00 98 60
```

- 7) Write this sector back to disk.

That's it. Your Ghostbusters is now deprotected. Enjoy the game!

Jim S. Hart
Jacksonville, NC

About Spellworks...

Let me begin by complimenting you for the outstanding job all of you do in producing COMPUTIST magazine. I have tried many of the softkeys that you have presented and have been very pleased with the results. Please keep up the good work!

I recently received a couple of back issues that I ordered (they were delivered in less than two weeks from the time I placed my order) and would like to help out Dr. Donald Schubert with his problem with Spellworks

(COMPUTIST No. 27). The version of Spellworks that I own is 1.5. Here are the steps that I used to deprotect it.

- 1) On the /SPELLWORKS side of disk -

```
BLOAD SPELL.SYSTEM,A$2000,TSYS  
CALL -151
```

- 2) Change these bytes:

	FROM	TO
20DC -	20	EA
20DD -	00	EA
20DE -	60	EA
20DF -	50	EA
20E0 -	03	EA
20E1 -	4C	EA
20E2 -	E1	EA
20E3 -	20	EA
20E4 -	20	EA
20E5 -	00	EA
20E6 -	BF	EA

- 3) Save the modified file:

```
BSAVE SPELL.SYSTEM,A$2000,TSYS
```

- 4) On the /MAILMERGE side of the disk -

```
BLOAD MERGE.SYSTEM,A$2000  
,TSYS  
CALL -151
```

- 5) Change these bytes:

	FROM	TO
2056 -	4C	EA
2057 -	56	EA
2058 -	20	EA
2079 -	20	EA
207A -	00	EA
207B -	60	EA
207C -	70	EA
207D -	08	EA

- 6) Save this modified file:

```
BSAVE MERGE.SYSTEM,A$2000  
,TSYS
```

That's all you have to do. Hopefully, this will work for the version of the program that Dr. Schubert has.

Thanks for an outstanding magazine.

Al Pezewski
Milwaukee, WI

input

Ghostbuster APT

Just a few days ago I was playing the game Ghostbusters. I used the softkey that Kenny Khoo told us about in the June 1986 issue (COMPUTIST No. 32). I still couldn't solve the game. I got out some of the software I wanted to use to disassemble Ghostbusters and went to it. I found out that there was a cheat within the original Ghostbusters.

When it asks for your name type OWEN. When it asks if you have an account, type YES. When it asks for the account name, type LIST. Now you should have about \$72000000 or so. Now, the game can easily be solved. You can get your high performance car that you've been wanting to use.

Thanks COMPUTIST, keep the good work going.

Steve Z
Glendora, CA

A Word to the Copy Protectors

You must feel that the wall is continually banging itself against your head. No matter how difficult, convoluted, and costly your protections get. No matter how much the original program is twisted, bent, and ruined to fit the protection, someone is always able to not only copy the program, but make it so simple to copy, that a program like COPYA is enough to do the job!

"Why do they do it," you ask, "if not to steal the fruits of my hard labor by PIRATING my program?" There could be no other reason for spending weeks of hours on cracking one program; right? WRONG!

I, and probably most of my "cohorts", do it for many reasons. However, not one of these reasons is theft. They do include:

protection: Who wants to spend money on postage & replacement charges and wait days or weeks when their only disk fails? Remember, even this can happen only if the publisher is still in business!

anger: Imagine how fast ZORK would be if it could be loaded into a RAM card, or Flight Simulator, or any other program for that matter. Look what they did for Appleworks!

boredom: Isn't it silly to wait almost a minute for a protected program to boot, when it only takes 10 seconds after it's broken and put on a fast DOS?

Many will agree, though, that the most important reason for spending hours hunched over a table deciphering yards of unreadable code is CHALLENGE. Imagine working on the largest jigsaw puzzle you can find and knowing before you start that all of the pieces are there. Put together a little corner here, the bright yellow fishing boat there, maybe put the outline together first, or try and jump straight into the middle. And what do I get after all that time and effort? Not a picture that gets torn up and put away and lose bits of itself; but a drawer full of original disks that don't need to be touched again except in an emergency, programs that may be modified to suit the way I like them, and the biggest rush of self esteem since the original pat on the back.

Copy protectors, I hate your guts for ruining those fantastic programs; and, I could kiss you for giving me a jigsaw puzzle that my cats won't steal the pieces to.

Robert T. Muir
Ferndale, CA

Back to F-15 Strike Eagle

First I'd like to thank those people who offered their help in response to my "Input" letter referring to F-15 Strike Eagle. Apparently, my assumption that others were having similar problems was true. Unfortunately, I've still been unable to softkey (or even copy for that matter) my disk.

The problem occurs after boot up. The three letters from SOLO 7, Neil Lemme, and M. M. McFadden all deal directly with the code in the file "AS" or code similar to it in another file. Mr. McFadden said that SIDTRANC.OBJ3 contained the other copy protection code but this was not the case with my version (2-F-86B-1 15 June 1985 change 2). In fact, I can find no other files on the disk which contain code similar to that in "AS" or with any references to RWTS.

I've modified "AS" so that my disk boots up fine and works well all the way to the request for the authentication code. After this is entered I hear the disk make the same kind of accessing that it does on boot up under the control of "AS". Then it kind of burps, slides around a bit and then the playing screen comes up with "HARDWARE FAILURE!!" indicating a bad load. I've looked long and hard through the code in the file B.OBJ for the JSR instruction which causes the disk access but can't seem to narrow it down. B.OBJ is the file which asks for the initial scenario set up and then for the authentication code (It's a long unsuccessful

story which I won't try to tell now.) Anyhow, I'm about at my wit's end with this thing and hope that maybe someone out there has softkeyed the version that I own.

As I said before, I'm new at this business and enjoy COMPUTIST thoroughly. Someone please come to my rescue!!

Dan Williams
Pensacola, FL

A Companion to the Companion

In COMPUTIST No. 32, I had some problems in backing up Print Shop Companion. It seems to me that there are many versions of this program. But anyway, I managed to deprotect my version.

1) Copy the original disk with COPYA or any other fast copy program. Ignore errors on track \$22.

2) Use your favorite sector editor to perform the following changes:

TRACK	SECTOR	BYTE	FROM	TO
\$00	\$0E	\$24	20	EA
\$00	\$0E	\$25	E0	EA
\$00	\$0E	\$26	BC	EA
\$0D	\$06	\$BD	20	EA
\$0D	\$06	\$BE	E0	EA
\$0D	\$06	\$BF	BC	EA

3) Write it back to the disk and happy printing!

The Scorpion
Pirates Harbor of Sao Paulo

Fixing ProDOS

I have found that much of the new ProDOS software written to run only on the enhanced //e and //c can be modified to be able to boot on a 128K //e with a 65C02 without an error message by searching for a simple sequence of bytes using almost any sector editor. Recently I was prevented from successfully booting Reportworks (an excellent adjunct to the Appleworks database) and Pinpoint (a desktop utility or "Sidekick" type of program including calendar, dialing, telecommunications, etc. which can be called from Appleworks). I was able to reverse this "Machine-specific protection" by using a sector editor to search for a sequence of bytes similar to the following:

input

```
AD C0 FB LDA $FBC0
F0 XX BEQ XXXX (to continue boot)
C9 E0 CMP #$E0
F0 XX BEQ XXXX (to continue boot)
4C XX XX JMP abort.boot
```

Simply reroute the "JMP abort.boot" to JUMP to the go location which continues the boot, and it will work just fine on a 128K //e with a 65C02 installed.

Pinpoint

In order to allow to boot on an unenhanced (ROM-wise) //e with 128K and 65C02 processor, change **4C A5 78** to **EA EA EA** (Track 8, Sector B, Bytes \$ 68, 69, 6A) on a copy of the original disk. It will run perfectly, except characters for the "mousetext" video ROM will be displayed as inverse control characters instead of "Open-Apple" "Closed-Apple" characters, etc.

Note: the sequence at **AD C0 FB** (located at 775F on my version) can be searched for (and disassembled on disk, by Copy II+ v 5.0) on any disk which requires an enhanced //e or //c to boot, in order to modify the software to allow it to boot on a 128K //e with 65C02 installed (but without enhanced ROM's which are used for identification only, but not needed for any ProDOS software).

Steve Wilson
Fountain Valley, CA

A Couple APT's

Here are some advanced playing techniques that I have come across.

Beer Run: More Men

BLOOD BEER RUN
CALL -151
C64: number of men wanted
7FDG

Wizardry: free gold pieces

- 1) Make a level one Bishop (INT=12, PIE=12, Good or Evil).
- 2) Take the Bishop and one other party member down into the dungeon.
- 3) While in camp, make sure the Bishop is first in line.
- 4) Press "1" to see the Bishop's stats.
- 5) Now press "I" to identify and then press "J" until the word "Success" appears at the bottom of the screen.
- 6) Leave the Bishop and look at the character

directly beneath him. That character now has 100,000,000 gold pieces!

I hope these can be of use to someone. By the way, if any of you would like to get a penpal who enjoys talking "COMPUTIST" talk, here is my address:

Jim S. Hart
311 Bordeaux St.
Jacksonville, NC 28540

bugs

COMPUTIST No. 33:

The Mapping of Ultima IV:

Lines 150 and 700 of the Ultimainland Editor should read as follows:

```
150 VTAB 19 : HTAB 8 : PRINT C$: GOSUB 370 : VTAB
20 : HTAB 5 : PRINT "<-"; : HTAB 36 : PRINT
"->"
```

700 IF B\$ = CHR\$(27) THEN 250

The new checksums for this program are as follows:

10	- \$BADD	380	- \$9C32
20	- \$9B13	390	- \$A264
30	- \$4D3B	400	- \$19E7
40	- \$AD92	410	- \$443A
50	- \$C899	420	- \$88E0
60	- \$FF65	430	- \$A84B
70	- \$A3BF	440	- \$04AF
80	- \$646B	450	- \$FFEE
90	- \$EC83	460	- \$03A3
100	- \$5BF9	470	- \$F612
110	- \$A345	480	- \$DA62
120	- \$4D8B	490	- \$2CC8
130	- \$75F3	500	- \$BA53
140	- \$FE5A	510	- \$73A9
150	- \$E9D6	520	- \$0222
160	- \$F428	530	- \$082C
170	- \$F791	540	- \$6EF5
180	- \$7FAC	550	- \$D999
190	- \$FCCB	560	- \$422C
200	- \$8CFA	570	- \$4ABD
210	- \$885C	580	- \$F137
220	- \$8EA3	590	- \$EC3B
230	- \$8DC6	600	- \$38DD
240	- \$C474	610	- \$564C
250	- \$2516	620	- \$0EB2
260	- \$317B	630	- \$E5CE
270	- \$97C0	640	- \$7E24
280	- \$CE8F	650	- \$DB44
290	- \$FB65	660	- \$300C
300	- \$A1DD	670	- \$7340
310	- \$FEF7	680	- \$A397
320	- \$8580	690	- \$9FDC
330	- \$30F8	700	- \$8C9A
340	- \$12DA	710	- \$ED90
350	- \$4A59	720	- \$2939
360	- \$7DEC	730	- \$60F8
370	- \$24CC		

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.

Gutenberg 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

Agent U.S.A. Scholastic

Handicapping System Sports Judge

Odin Odesta

Mabel's Mansion Datamost

Brain Bank The Observatory

Crimson Crown Penguin

Crypt of Medea Sir Tech

The Works First Star Software

Cross Clues Science Research

Peeping Tom Microlab

Jigsaw Microfun

Miner 2049er II Microfun

Create with Garfield DLM

Print Master Unision World

Bandits Sirius Software

Bank Street Filer Broderbund

readers' softkey & copy exchange

Glen Tatum's softkey for...

Magic Slate

Sunburst Education
39 Washington Ave.
Pleasantville, NY 10570

Requirements:

Locksmith fast copy or similar copier
ProDOS disk
Two blank disk sides

I recently purchased a word processing program called Magic Slate. As with most of the recent releases of this nature, it is on a ProDOS format. My only complaint with the software was that it was copy protected. Needless to say, if you read onward, you will discover how to foil this copy protection.

Step by Step

- 1) Use Locksmith fast copy to copy both sides of the disk. Ignore any errors you get on track 1.
- 2) Boot your ProDOS disk and get into BASIC.
- 3) Put in your copy of the 80 column side of Magic Slate and see how long the MS file is.

CATALOG

On an 80 column screen, the length of the file will be displayed under the ENDFILE heading. On a 40 column screen, the file length can be found on the third line under the ENDFILE heading. The length of my file was 12536. Write down this length somewhere.

- 4) Load the MS file and modify it.

```
BLOAD MS,TSYS,AS$2000
CALL -151
21DB:4C 64 20
```

This modification is located right at the beginning of the code that prints an error message on the screen if the nibble count routine fails. It makes this subroutine JuMP to the start of the program.

- 5) Save this modified code.

```
UNLOCK MS
BSAVE MS,TSYS,AS$2000,L12536
```

You should use the length you wrote down for the L parameter of the BSAVE command.

- 6) Now rename this file so that ProDOS will automatically boot it up.

```
RENAME MS,MS.SYSTEM
LOCK MS.SYSTEM
```

The other side of the disk can be deprotected using the same method. Except at step four you

```
You are in one of your
"ultra-cool" moods. While
cruising through the
house, you bump your foot
on a piece of furniture
and you let a swear word
sneak out. Your mother
calls you in from the
other room. She says, "Did
you say what I THOUGHT you
said?"
```

PRESS SPACE TO CONTINUE

```
"SO WHAT?" She decides to
show you so what. You are
not too big to get your
mouth washed out with
soap. With superhuman
strength, she grabs you by
your cool haircut, dumps
you into the cool
bathroom, and forces a
quarter of a bar of soap
(with lemon-fresh skin
conditioners) down your
ultra-cool throat.
```

PRESS SPACE TO CONTINUE

should type:

```
222C:4C 61 20
```

Both sides should now be completely deprotected and COPYAable.

Charles Taylor's softkey for...

Alter Ego

Activision
Box 7287
Mountain View, CA 94039

Requirements:

Three double-sided blank disks
COPYA
Sector Editor

Alter Ego is a role-playing game of life. This is your chance to replay your life with a different personality. You may begin at any of seven stages of life, living out your fantasies without the risks. The game is sort of R-rated, recommending parental guidance for those under 16. There is a male and a female version of Alter Ego. This softkey was done on the male version, but the softkey should also work for

REVIEW

Select a mood:

- TRUTHFUL
- LESS THAN TRUTHFUL
- TOO COOL TO CARE

Select an action:

- TELL HER YOU DID AND APOLOGIZE
 - TELL HER YOU DIDN'T
 - TELL HER YOU DID AND SO WHAT
- CONTINUE

REVIEW

Select a mood:

- ANGRY
- DEPRESSED
- UNFLAPPABLE

Select an action:

- WALK AWAY
 - SAY SOMETHING TO HER
 - GIVE HER AN ALMOST UNIVERSALLY-UNDERSTOOD FINGER SIGNAL
- CONTINUE

the female version.

The program consists of three double-sided disks, all of which can be copied by COPYA. Unfortunately, the boot side will not boot, that is until we make a few will-chosen sector edits. By scanning the disk for the bytes 8C C0, I discovered that the copy protection lies in the bootup program "Alterbas" on track 13, sector 5. By eliminating some of the conditional branch instructions in the code, I was able to produce a working copy of the boot side.

Summary

- 1) Copy all six sides with COPYA or equivalent.
- 2) Read track \$13, sector \$5 of side A, disk one and with a sector editor, make the following changes:

Byte	From	To
7C	D0 ED	EA EA
8D	10 FB	EA EA
91	D0 D8	EA EA

93	F0 EB	EA EA
9E	D0 CB	EA EA
A9	49 AA	EA EA

- 3) Write the changes back out to your disk.
- 4) Start living.

readers' softkey & copy exchange

John Howard's softkey for another...

Rendezvous

Peachtree Software
3445 Peachtree Road NE
Atlanta, GA 30326

Requirements:

At least 48K
Demuffin Plus (COMPUTIST No. 8)
(or Super IOB and FID)
A blank disk or two

After seeing the softkey for the newer version of Rendezvous in COMPUTIST No. 28, page 13, I thought that some of you out there might like a method to deprotect the old version.

- 1) Boot the Rendezvous disk.
- 2) Press Reset after tracks 0-1 have been read in (very shortly after the disk recalibrates). This allows the non-standard RWTS on the disk to be read into memory.
- 3) Type CALL -151 to get into the monitor.
- 4) Move the protected RWTS down into lower memory so that it will not be overwritten by booting DOS.

8600<B600.BFFF

- 5) Boot a 48K slave disk with no HELLO.
- 6) INITIALize the disk.

INIT EDU-WARE

- 7) Get into the monitor again and load in Demuffin Plus.

CALL -151
LOAD DEMUFFIN PLUS

- 8) Move the non-standard RWTS back up where Demuffin can use it to read the the original Rendezvous disk.

B600<8600.8FFF

- 9) Run Demuffin Plus.

803G

- 10) Follow the prompts to copy files from the original to the INITIALized disk.

You should now have a COPYAable copy. The Applesoft program EDU-WARE contains some code that you may want to change. Line 0 sets the & vector and line 1 sets the RUN flag \$D6 (214) to a value higher than \$7F, which has the effect of making Applesoft

ignore keyboard commands. The Reset vector at \$3F2 is also set so that you cannot reset out of the game. You may want to delete this part of the code so that the reset vector keeps its normal pointers, which will allow you to reset out of the game at any time.

Alternate method: Capture the RWTS as in steps 1-5. Save the RWTS to your Super IOB disk. Use the Swap Controller with Super IOB to convert the disk to a DOS 3.3 format. Copy the files on that disk to another disk INITIALized with the name 'EDU-WARE'.



Greg Robinson's softkey for...

Quicken

Intuit

Requirements:

COPYA
A sector editor
A notched, double sided disk

Here's a softkey for removing the copy protection from Quicken by Intuit. Quicken is a check writing and balancing utility with the capacity to transfer your check register to the spreadsheet in Appleworks. This Pascal based program is another example of the "deceptive track gap", to quote Bruce Jones from COMPUTIST No. 27.

The offensive routine locates and determines if a special sync byte is followed by four standard sync bytes, the address header, and a specific volume number. If any of these are altered the routine will not allow the disk to boot correctly. Copy II Plus version 6.4, Locksmith 5.5, and other miscellaneous copy programs were tried without success.

Deprotecting Quicken was accomplished by disabling the offensive routine (see below).

- 1) Copy both sides of Quicken using any standard (sector copy) program.
- 2) Use a sector editor and change track \$E, sector \$F, byte \$7F from a value of \$00 to \$01.
- 3) Remember... do it to BOTH sides.



Stanley Planton's softkey for...

Story Tree

Scholastic, Inc.
Box 7502
2931 E. McCarty St.
Jefferson City, MO 65102
\$39.95

Requirements:

64K Apple][Plus, //e, or //c
COPYA
2 Blank disks
Sector Editor

Scholastic seems to use an almost-standard DOS, with an error-checking routine added to track \$00. Sector \$05 to prevent its disks from being copied with COPYA.

Fortunately, the version of "Story Tree" I had available could be fairly easily backed up by first defeating COPYA's error checking, and then by changing three bytes on track \$00. This sector edit is based on Phil Pattengale's softkey for Microzines, on page 13 of COMPUTIST No. 27. It may well be that this is a "universal fix" for Scholastic's software.

Boot your system master disk and enter the monitor.

CALL -151

Next, instruct DOS to ignore any read errors.

B942:18

Reset into Applesoft and startup COPYA.

RUN COPYA

COPYA should now be capable of copying the boot disk of your "Story Tree". If you feel adventurous, you might try to boot the disk at this point, being rewarded with a screenful of garbage, and constant reboots. The next task is to shut down that pesky protection scheme!

Get out your favorite sector editor and read track \$00, sector \$05 (I used COPY][Plus 6.0). Jumping to addresses \$93-\$95 should reveal the sequence:

C6 2A D0

Enter your sector editor's write mode and replace these bytes with the following code:

4C 86 02

You now have a COPYAable boot disk for "Story Tree"! The "Story Disk" appears to be completely unprotected, and may be copied with COPYA.



readers' softkey & copy exchange

B. Croome's softkey for...

Assembly Language Tutor

Assembly Language Tutor
Prentice-Hall Inc. (1983)
Englewood Cliffs, NJ 07632

Requirements:

Apple with 48K
One disk drive
48K slave with HELLO program deleted
Demuffin Plus

Assembly Language Tutor is an assembly language teaching program which I found to be a great help to me as a novice programmer. It is not perhaps as sophisticated as some of the newer ones, but it certainly gave me a good grounding and start. Unfortunately, the original would not copy with COPYA although other programs like Copy][Plus would do the job. I decided that an unprotected backup would be an advisable safe-guard as copying the protected copy with Copy][Plus was a pain. It struck me that from the fast boot up and lack of disk access that this program might not be terribly long. Perhaps it could be saved as a single file!


On boot, the disk displayed the name of the program and the publishers name beside an Applesoft prompt character. This was a good sign that a fairly normal DOS was being used. Either Super IOB or perhaps even Demuffin Plus appeared to be good candidates for the deprotection. I first attempted to copy the disk with Locksmith Fast Copy. It copied track 0 but gave read errors on the remainder of the tracks. This indicated that the boot track was normal but the publishers had modified the remainder of the disk to protect it. Upon examination of the disk, the remainder of the disk appeared to have an address field header of D5 AA B5 rather than the normal D5 AA 96. I attempted to use Super IOB and the swap controller but I just couldn't get it to work.

Re-examination of the disk indicated that either some sectors contained invalid data or that some of them were written in a slightly weird format. Rather than trying to figure this out any further (and being a bit lazy), I attempted to catalog the disk with the strange RWTS installed. To my amazement, I saw a directory with two files (Hello and Tutor).

Well, this looked promising. What had to be done now was to convert these files from the strange RWTS format to a normal one. It turns out that Demuffin plus will do the job just fine.

After using Demuffin plus, I was very happy to have a fully functional deprotected copy of Assembly Language Tutor.

Cookbook Instructions

- 1) Boot the Assembly Language Tutor disk.
- 2) Hold down  so that it repeats.
- 3) Enter the monitor and move the tutor RWTS to a safe place.

CALL -151
6800<B800.BFFFM

- 4) Boot a 48K slave disk with no HELLO program.

6 

- 5) Load Demuffin Plus and put the strange RWTS where it belongs.

BLOAD DEMUFFIN PLUS
B800<6800.6FFFM

- 6) Start up Demuffin Plus and convert the files using the wildcard character "*" = "".

803G

John Howard's softkey for many...

Avalon Hill games

Avalon Hill Microcomputer Games
4517 Harford Rd.
Baltimore, MD 21214

Requirements:

Demuffin Plus (see issue No. 8, page 15)

Facts in Five, Gulf Strike, The Parthian Kings, Fortress of the Witch King, Talengard, and Galaxy are some Avalon Hill games that can be deprotected using Demuffin Plus (or Super IOB and FID). After moving the files to a normal disk, determine what the boot program is, initialize a disk with this name, copy the files to this new disk, and you are done. Here is the procedure:

- 1) INIT a blank DOS 3.3 disk.
- 2) Boot the game disk.
- 3) Press Reset after tracks 0-1 have loaded in.
- 4) Enter the monitor and move the protected RWTS to a safe place.

CALL -151
8600<B600.BFFFM

- 5) Boot normal DOS 3.3 and BLOAD DEMUFFIN PLUS.

- 6) Re-enter the monitor and move the protected RWTS into its original place.

CALL -151
B600<8600.8FFFM

- 7) Start Demuffin Plus with 803G and follow the prompts.

- 8) After the files are transferred to your formatted disk, determine what the HELLO program is, INITIALize a new disk with this name, and transfer the copied files to this disk.

Andrew Swartz' softkey for...

Dark Crystal

Sierra On-Line, Inc.
Sierra On-Line Building
Coarsegold, CA 93614

Requirements:

Apple][or better
Dark Crystal original disks
A sector editor
COPYA or equivalent
Two notched blank disks (4 sides)

Dark Crystal is another one of Sierra On-Line's hi-res text adventure games, which is a take off from the movie "The Dark Crystal" by Jim Henson, creator of the Muppets. As with other similar games by Sierra, the copy protection is not that complicated nor is it difficult to overcome.

Here's how to crack it:

- 1) Copy all four sides of the original disks to your blanks with COPYA.
- 2) Now, let's sector edit the first (boot) side. It is the only side that is protected.

Track	Sector	Byte	Change To
\$05	\$0F	\$A8	SEA (NOP)
\$05	\$0F	\$A9	SEA
\$05	\$0F	\$AA	SEA
\$07	\$0C	\$22	SEA
\$07	\$0C	\$23	SEA
\$07	\$0C	\$24	SEA

- 3) Put away your original in a safe spot and enjoy your cracked version!

Karateka

by Bill Hamshire

Requirements:

64K Apple II

Karateka

A sector editor

Note: COMPUTIST recommends that you NEVER, EVER modify an original disk. Use only a working bit copy of the original if possible.

Anyone who owns an Apple IIc may have noticed that some programs do not run on it. This is because of the new Apple IIc ROM. Unfortunately, Karateka by Broderbund Software uses one of the changed ROM routines making it unplayable.

This is a particular shame because Karateka is just as fun to watch as it is to play, because of the smooth flowing graphics and realistic action. The opponents get progressively harder as the karateka kicks and punches his way to save the beautiful princess. The only drawback is that after much practice, the main fighting skills are mastered, and the game gets rather easy and soon ends, but there is always a next time. Other than that, and the fact that it will not boot on my IIc, Karateka is a masterpiece.

There may be other versions that will boot properly, but my version is only compatible with the Apple II, II Plus, IIe. When the power is turned on with the disk in the drive, the initial boot process begins. After calibration, resetting the drive head to track 0, the screen darkens, the disk drive whirs, the drive head clicks from track to track, then surprisingly, the screen fills with a whole lot of nothing. All is fine so far, but a few seconds, plus a short tone indicating an error, later, all that is left is an "R" printed in the upper left-hand corner and the sound of recalibration.

The Problem

Every time Karateka goes to the disk drive, whether for booting or for advancing to higher levels, it brings down the initial boot routine, boot 0, from \$C600 and stores it in the second page of graphics at \$4600. Here, the program can make small modifications to boot 0 and use it in accessing the disk. The routine from \$C600 to \$C6FF in the Apple IIc is, of course,

different, so the problem seems to be getting the old "boot 0" into memory. This new version will run on any Apple II with a slot 0 RAM card (64K).

The Procedure

Let's start with the easy part.

- 1) Run a sector editor.
- 2) Read track 0, sector 0.
- 3) Change bytes \$E0, \$E1, and \$E2 from EA EA EA to 20 57 B7. This simply puts in a JSR \$B757 (jump to subroutine) to execute the routine explained below.

And now for the hard part. Since the boot 0 routine cannot be "squeezed" in anywhere, it will have to be read in separately. Karateka uses the entire memory from \$0000 to \$BFFF, but not the slot 0 RAM card (the language card). To use this area of memory, a simple routine is implemented (Examine hexdump #1, (\$57-\$72)).

This section will copy the monitor ROM (\$F800-\$FFFF) into the language card leaving \$D000-\$F7FF free to store the old boot ROM.

- 4) Run a sector editor, and read track 0, sector 1.
- 5) Type in hexdump #1 from \$48 to \$89. Do not worry about what is being wiped out.
- 6) Write the sector back to track 0, sector 1, and it is finished.

This routine has many purposes besides opening up the language card. The first part will set up the pointers \$D0, which points to the page where the boot 0 routine will be stored, and \$46, which points to where the routine is moved to for use by the program.

The rest, from \$57 to \$8A, makes up a routine to: 1) set up slot 0 as said earlier, and 2) set up the pointers to read in track 0 sector B. What is that? That is the sector which will hold the old ROM. But, how did the ROM get to track 0, sector B?

Getting It There

There are two ways to get the old ROM into track 0, sector B. I will start with the easiest.

- 7) Find an Apple II other than the IIc.
- 8) Run a sector editor similar to The Inspector.
- 9) Set the buffer to \$C600 (so that page \$C600-\$C6FF will be written to the sector).
- 10) Set the track to 0 and sector to \$B.

- 11) Write the sector to the Karateka disk.

If an "old" (non-IIc) Apple is not available, hexdump #2 can be typed in with a sector editor. Have fun with that, and have fun with Karateka on the Apple IIc.

Hexdump #1: The Patch

(checksummed at \$2048.2089)

xx48:	A9 D0 8D 43 03 A9 46 8D	\$3576
xx50:	47 03 A9 80 4C 00 03 AD	\$65AC
xx58:	81 C0 A0 00 B9 00 F8 99	\$142C
xx60:	00 F8 C8 D0 F7 EE 5E B7	\$1323
xx68:	EE 61 B7 AD 61 B7 D0 EC	\$8472
xx70:	AD 83 C0 A9 01 8D E1 B7	\$46DF
xx78:	A9 00 8D EC B7 A9 0B 8D	\$AEBF
xx80:	ED B7 A9 D0 8D F1 B7 20	\$89AA
xx88:	93 B7	\$A1F0

Hexdump #2: The Boot 0 ROM

(checksummed at \$2600.26FF)

x600:	A2 20 A0 00 A2 03 86 3C	\$2AE1
x608:	8A 0A 24 3C F0 10 05 3C	\$7D35
x610:	49 FF 29 7E B0 08 4A D0	\$49E1
x618:	FB 98 9D 56 03 C8 E8 10	\$C70B
x620:	E5 20 58 FF BA BD 00 01	\$4FD5
x628:	0A 0A 0A 0A 85 2B AA BD	\$A699
x630:	8E C0 BD 8C C0 BD 8A C0	\$E530
x638:	BD 89 C0 A0 50 BD 80 C0	\$A793
x640:	98 29 03 0A 05 2B AA BD	\$A041
x648:	81 C0 A9 56 20 A8 FC 88	\$1E16
x650:	10 EB 85 26 85 3D 85 41	\$3FDB
x658:	A9 08 85 27 18 08 BD 8C	\$FC07
x660:	C0 10 FB 49 D5 D0 F7 BD	\$CCB8
x668:	8C C0 10 FB C9 AA D0 F3	\$4294
x670:	EA BD 8C C0 10 FB C9 96	\$95AD
x678:	F0 09 28 90 DF 49 AD F0	\$9A85
x680:	25 D0 D9 A0 03 85 40 BD	\$332A
x688:	8C C0 10 FB 2A 85 3C BD	\$7D29
x690:	8C C0 10 FB 25 3C 88 D0	\$0A4B
x698:	EC 28 C5 3D D0 BE A5 40	\$DD96
x6A0:	C5 41 D0 B8 B0 B7 A0 56	\$ADE7
x6A8:	84 3C BC 8C C0 10 FB 59	\$5F14
x6B0:	D6 02 A4 3C 88 99 00 03	\$8838
x6B8:	D0 EE 84 3C BC 8C C0 10	\$785B
x6C0:	FB 59 D6 02 A4 3C 91 26	\$FBE3
x6C8:	C8 D0 EF BC 8C C0 10 FB	\$5B1B
x6D0:	59 D6 02 D0 87 A0 00 A2	\$9397
x6D8:	56 CA 30 FB B1 26 5E 00	\$54F7
x6E0:	03 2A 5E 00 03 2A 91 26	\$23E3
x6E8:	C8 D0 EE E6 27 E6 3D A5	\$0BB7
x6F0:	3D CD 00 08 A6 2B 90 DB	\$1E80
x6F8:	4C 01 08 00 00 00 00 00	\$AD2A

by John Howard

Requirements:

One disk formatted as a data disk
A utility to move DOS up to the language card
Two disks initialized with the name of the DOS mover
Demuffin Plus

Under Fire is one of Avalon-Hill's better war games. In addition to using the non-standard formatting used in my previous articles, it employs a number of other protection schemes. They are as follows:

- a) Altered address and data trailers in disk sector formatting.
- b) A DOS that has been moved to the language card.
- c) Does disk checks on quarter tracks between tracks \$09-\$0A.

determine how sector formatting has been changed. If disk formatting hasn't been changed too much, or sector data hasn't been encoded by altering the byte translate tables, and it is not a direct-load program, then files can usually be transferred to a normal DOS 3.3 disk using Super IOB 1.5, Copy II Plus, or Demuffin Plus.

With Under Fire it so happens that we can use Demuffin Plus to capture all the files to a normally formatted disk. When the game is first booted to the title screen, the only protection used up to this point is non-standard sector address and data trailers, so that resetting at this time will give you a normal reset with an Applesoft prompt and the non-standard RWTS in memory from \$B600 - \$BFFF. The files on all three disks can then be transferred using Demuffin Plus.

On boot the Applesoft file "HELLO" is run which loads the title screen and after the prompt bruns the binary file "SETUP". The rest of the HELLO program loads maps and does housekeeping. The file "SETUP" contains the code that moves the non-standard DOS up to the language card, resets the run flag and the

7) Load in Demuffin Plus.

BLOAD DEMUFFIN PLUS

8) Move the Under Fire RWTS back to it's normal location.

B600<8600.8FFFM

9) Start Demuffin Plus.

803G

10) Follow the prompts and copy the original units and scenarios disk to the formatted data disk, and the master and map maker disks to the disks initialized with a DOS mover utility.

11) Boot a normal DOS disk.

12) Load the file "SETUP" from the master or map maker copy.

BLOAD SETUP

13) Enter the monitor.

CALL -151

softkey for...

Under Fire

d) Sets the RUN flag of Applesoft to ignore keyboard input.

e) Wipes memory and initializes your disk if it finds an error in the disk check.

f) Points the reset vector to a routine in page three that will initialize your disk.

The first thing that I usually try on a protected disk is to read it with the Locksmith fast copy program. One of the nice things about this program is that read and write error types are displayed on screen in the program's track, sector map. This information gives a quick overview of possible protection schemes, giving some idea of where to start deprotecting. If no errors are shown after a read of a protected disk, then the protection probably employs some sort of disk check or nibble count. Defeating this type of protection requires that the code doing the disk check be found and altered so that the disk check code is bypassed or returns a good check of the disk.

If Locksmith fast copy shows errors in the disk read, then I use a nibble editor to try to

reset vector, does the disk check, wipes memory, and initializes your disk on a disk check if it doesn't find what it is looking for. The "SETUP" program code has to be in memory for the program to run properly, but by changing a few bytes of code we can bypass the protection.

Following is a step by step procedure:

1) Boot the Under Fire master disk.

2) Reset out of the program when the title page is displayed and the disk drive will stop.

3) Get into the monitor.

CALL -151

4) Move the Under Fire RWTS to a safe spot in memory.

8600<B600.BFFFM

5) Boot a normal DOS disk with no HELLO program.

6) Enter the monitor again.

CALL -151

14) Alter some code in the "SETUP" file that defeats the protection and allows a normal Reset out of the game.

602A:20 58 FF 60

15) Save the modified "SETUP" file to the master and map maker copies.

BSAVE SETUP,A\$6000,L\$4AB

16) Save the DOS mover utility to the master and map maker disks with the name that was used to initialize them. (Your DOS mover utility should be of the kind that will run a "HELLO" program after moving DOS to the language card.)

That's it

If you don't like some aspects of the game, you can change them by resetting out of the game at any time. Program files will be intact in memory or accessible from disk.



Pegasus II

by Ed Croft

Requirements:

- 1 blank disk
- Sector editor
- DOS 3.3 master disk

A lot of people I know are always asking how to "crack" disks. They know a little about the subject (mainly from reading this magazine). But they complain that most of the cracks are not explained thoroughly enough and that they just don't know where to start. Reading the right books helps. Two that are absolutely indispensable are "Beneath Apple DOS" and "What's Where in the Apple." I am certainly no expert, but I do hope to at least help a beginner out enough to get him started in disk deprotection.

First of all let me start off by explaining a few things. By no means can this or any article cover all the different protection schemes that exist. Another point to mention is that if you were to go get a pilot's license they don't start by putting you in a F-15! So don't get the latest release of Electronic Arts and expect to crack it the first night! As they say "you have to walk before you can run." Well, that's true in computers too. I know most of you probably won't have this particular program. But don't worry because the whole point of this article is to give a beginner an idea on how to start on a disk of their own. And the best way to start is on an older disk without the heavy protection on most of today's disks.


Well has everyone got their disks ready? OK, I think we can all agree that the first thing we should try is booting the disk. Here are a few things to look for:

- 1) Look for a prompt.
 - a) A prompt usually means that the disk is a relatively normal DOS.
 - b) The absence of the prompt means that possibly there is no DOS or a highly modified one.

- 2) Listen for any unusual disk arm movements.
 - a) The normal boot sounds (a recal, three short swooshes and a long one) could also indicate a somewhat normal DOS.
 - b) A long swooshing sound or something resembling a cockroach tap dance probably means that the arm is moving to or performing a nibble count.
- 3) Watch for hi-res & text page changes.
 - a) Some protections will turn on the hi-res screen to cover up their Applesoft prompt.
 - b) It's always nice to note just when text or graphics pages are changed.

OK, I have booted Pegasus and notice that I do have a prompt. I didn't hear any unusual disk arm movements and I have noted when the hi-res page comes on.

Next we want to see if we can reset into the program some way. Start with the basics and work your way up. Here are a few suggestions:

- 1) Let the program boot then hit Reset (you might have to hit it more than once).
 - 2) While disk is booting hit  to stop program.
 - 3) Use modified language card (instructions below).
 - a) Enter the monitor.
CALL -151
 - b) Write enable the language card.
C081 N C081
 - c) Copy the ROM to the language card.
D000<D000.FFFFM
 - d) Set the card to Reset into the monitor.
FFFC:59 FF
 - e) Turn the card on and ignore ROMs.
C080
 - 4) Reset into the monitor using any hardware device (Wildcard, Replay etc.).
- OK by hitting Reset after the boot I have gotten into Pegasus II. I hope all of you have reached this point by one way or another. Now what we want is to find out how much

protection there is. Also we want to keep our eyes open to try to figure out just what type of protection is being used. So here are a few things that I always try:

- 1) CATALOGing the disk (of course)
 - a) Try a "CATALOG" from BASIC.
 - b) Try an "A56EG" from the monitor.
- 2) Typing "CALL -151" to get into the monitor
- 3) Listing any program in memory
- 4) If you had a disk search utility, look for a catalog & VTOC.
- 5) I also like to try to use a track by track copier (such as Locksmith fast-copy) to try to locate unusual or nibble count tracks if necessary.

Pegasus II is CATALOGable from BASIC, I can LIST the program in memory and enter the monitor.

If you cannot make it this far, perhaps the protection on the disk you are trying is too hard to start out on. You should try to get an earlier release and start over.

Well our disk seems to be pretty normal, but now we'll need to try to get a copy of it to work further on. As before start with the simplest method and work your way up. First, though, try using COPYA to copy your disk. If that doesn't work then try removing error detection by POKEing 47426,24. If a copy still cannot be made go to a nibble copier (I use Copy II Plus).

I have tried COPYA and it cannot copy Pegasus. This seems really strange since the disk seems to be so normal, so before going to my nibble copier, I want to scan the disk with CIA. Trying the sector editor gets me a big I/O error. So I switch over to the Linguist (which is a raw nibble dump routine). Right away I notice a change in the address header from a normal D5 AA 96 to D5 AA B5. The light above my head glows and I return to Tricky Dick and switch from DOS 3.3 to DOS 3.2. After that little change there is no trouble at all. It seems that this is one of those disks that came out right after the change to DOS 3.3. What some

software companies did was to keep the disk in a 3.2 format and alter DOS on the disk so that it could boot with a 3.3 controller.

Fortunately for us, Apple supplied a program on the system master that will change 3.2 files over to a 3.3 formatted disk. This program is called Muffin and it can be found on the system master. So I tried running Muffin and transferring my Pegasus files over to a normal 3.3 formatted disk. This time I was lucky and everything goes OK so I won't have to try to use my nibble copier.

I then put a write-protect tab on my Pegasus disk and tried to boot it. I observed it closely and noticed a prompt followed by the hi-res screen clearing and the title page loading. Next, a message at the bottom of the screen asked me if I want to play a game or watch a demo. After hitting "P" to play a game, a few more prompts appeared and then several files are loaded. After the fourth file (approximately), the screen fills with inverse "at" signs, which means that memory has been zeroed out.

We of course want to know which file performs this dastardly feat. So, I loaded the HELLO file and LISTed it. I then removed line 5 (it turns on the hi-res screen) and typed "MONCIO" which tells DOS to show all commands that are passed to it. After making these changes, I noticed that everything works fine until a file called "SCRAM.800" is BRUN.

Now that we know which file it is we finally get to the fun part of loading this program in memory and listing it out to your printer. By BLOADing it and then going into the monitor and entering AA72.AA73 we get the starting address of the program which is \$800. (the bytes are reversed remember?) Then enter AA60.AA61 to get the length of the file, which is \$1800. So now get the printer going and enter the starting address with all the L's behind it you can get. A few are probably asking if it is really necessary to list out the whole program. The answer is yes, and when you start tracing the code in the listing you will see why.

All right everybody get their listing ready. Now what I do on my listing is to under line all the JMP's & JSR's to different locations (which is like a "GOTO" and "GOSUB" in Applesoft) and all the RTS's (which is like a "RETURN" in Applesoft). Everybody do this on the first page of the listing as this helps separate different routines within the program. Here is an example:

```
0800- 20 E2 09 JSR $09E2
-----
0803- A9 40 LDA #$40
0805- 85 59 STA $59
0807- A9 0A LDA #$0A
0809- 85 5A STA $5A
080B- 20 96 14 JSR $1496
-----
080E- AD 00 C0 LDA $C000
0811- C9 CA CMP #$CA
```

By the first instruction you can see that the program goes to location \$09E2 right away, so now lets go there. At \$09E2 we see this listing:

```
09E2- 20 AA 1D JSR $1DAA
```

```
09E5- A9 1F LDA #$1F
09E7- 85 59 STA $59
09E9- A9 0A LDA #$0A
09EB- 85 5A STA $5A
09ED- 20 96 14 JSR $1496
-----
09F0- AD 00 C0 LDA $C000
09F3- C9 D3 CMP #$D3
```

OK, so now we can see that the program takes a big leap down to location \$1DAA. So let's take a look at some of the code there.

```
1DA9- 16 A0 ASL $A0,X
1DAB- 00 BRK
1DAC- A9 55 LDA #$55
1DAE- 59 BA 1D EOR $1DBA,Y
1DB1- 99 BA 1D STA $1DBA,Y
1DB4- 88 DEY
1DB5- D0 F5 BNE $1DAC
1DB7- EA NOP
1DB8- EA NOP
1DB9- EA NOP
1DBA- FC ???
1DBB- 54 ???
```

Now we see something interesting; they have jumped to the middle of an instruction! Now is when you have to keep an eye out for anything unusual in the listing. And the first thing we see is that there are just a few valid instructions and then the code looks like garbage without even a RTS instruction to return to the previous routine.

By entering the monitor and listing the code at \$1DAA we see that they are loading the Y register with zero and the accumulator with \$55. They then EOR it with all that garbage that we

noticed. Since we are here to learn let's replace all those NOP's with zero's and execute the code beginning at \$1DAA. After you do this you will see that all that garbage has been changed to valid code. By looking over this code we finally find our disk protection. Since we want to skip this routine we simply put an RTS instruction at \$1DAA so that the program RETURNS before encountering it or we could NOP the jump to \$1DAA either one.

Well I hope this little adventure through disk deprotection has helped some of the beginners out on how to start out on a disk of their own. Any terms your not familiar with will be in the two books I've recommended.

Step by Step

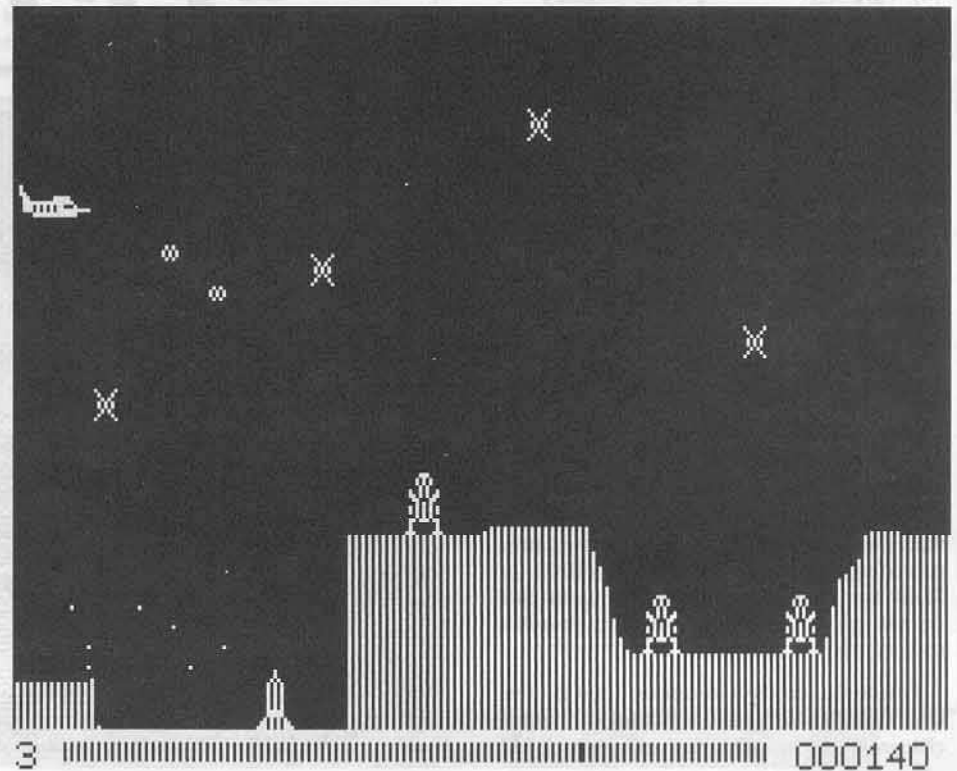
- 1) Muffin files to a formatted blank DOS 3.3 disk.
- 2) Bload scram.800 and insert \$60 at \$1DAA

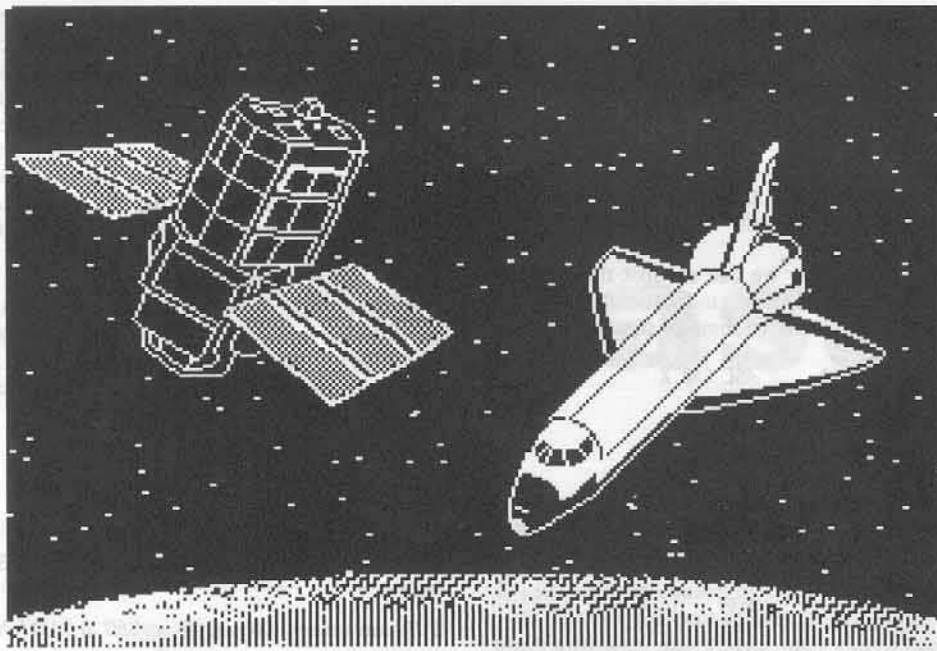
**BLOAD SCRAM.800
1DAA:60**

- 3) Bsave scram.800.

BSAVE SCRAM.800,A\$800,L\$1800

That's it!





Baudville
Software

Requirements:
COPYA or similar
A sector editor

Recently I purchased Take 1 and attempted to deprotect it using Clay Harrell's softkey printed in issue #25 (page 14). However, Baudville had apparently changed their protection scheme, foiling the softkey. Upon inspecting closer, I discovered that the changes were minimal and demanded but slight modification to the original softkey.

Clay Harrell's softkey performs the following sector edits:

Track	Sector	Byte	From	To
\$0	\$06	\$16	\$D0 F7	\$EA EA
\$0	\$0F	\$08	\$D0 F7	\$EA EA

The second sector edit worked fine on my disk, yet the original bytes for the first one were different. I found the D0 F7 to be five bytes earlier in the sector, at byte \$11. I also noticed that the CoMPare instruction immediately preceding the Branch if Not Equal (BNE, or \$D0) compared the accumulator with \$00 instead of the \$FF listed in Mr. Harrell's softkey.

I made the incorrect assumption that the five byte shift had been the only change; I simply changed the sector edit to replace bytes \$11-\$12

with \$EA instructions. The disk would not boot.

After examining a hexdump of sector \$06 for a while, I noticed a second compare (CMP) instruction further down in the sector, at byte \$2C. Just after this CMP, a branch instruction sent program flow back up to the first CMP, at byte \$0F. This must be the offending routine. To eliminate it, I simply replaced the branch instruction at byte \$2E with \$EA (No OPeration) instructions. The disk booted without a problem.

The complete procedure follows.

Step by Step

- 1) Copy the disk using COPYA.
- 2) Using a sector editor such as DiskEdit, make the following sector edits:

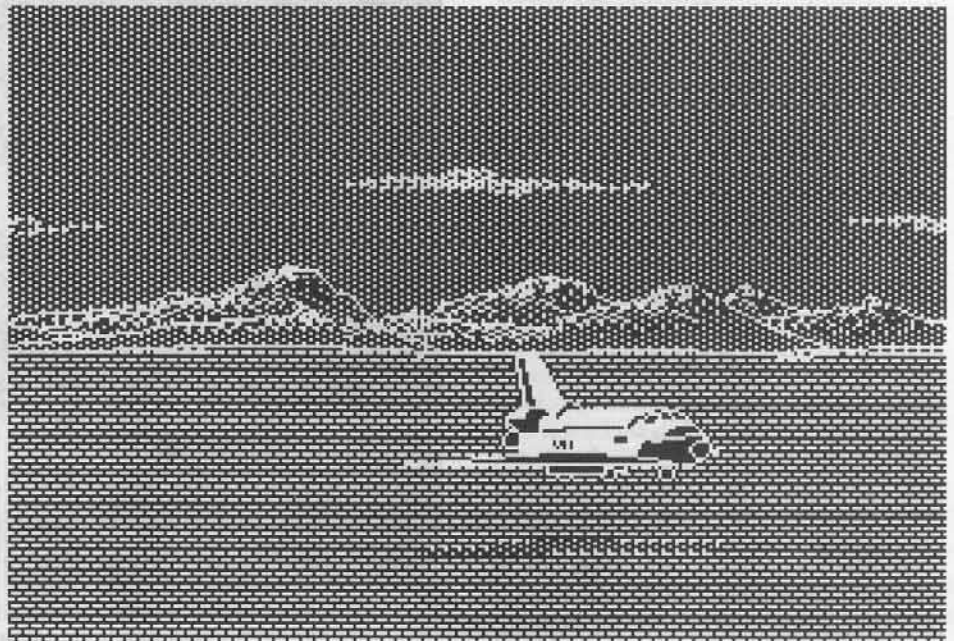
Track	Sector	Byte	To
\$00	\$06	\$11	\$EA
\$00	\$06	\$12	\$EA
\$00	\$06	\$2E	\$EA
\$00	\$06	\$2F	\$EA
\$00	\$0F	\$08	\$EA
\$00	\$0F	\$09	\$EA

- 3) You're finished! Take 1 is now completely deprotected.

softkey for...

Take 1 (Revisited)

by J. J. Gifford



Track Finder

by Denny Colt

Requirements:

A way of Resetting into the monitor
An Initialized disk.
DOS 3.3

Here is a neat little utility to add to your utility library. One of the hardest parts of copying a disk is finding out where the disk head was when the copied disk failed. In fact, COMPUTIST once printed an article on marking your disk drive cam just so you could more easily deal with this problem (Book of Sofkeys vol I).

In addition, there are a number of hardware modifications that can be added to your Apple that will tell you what track the disk is on, but they cost a lot of money. Since I don't have a lot of money, I wrote a software routine that tells the same thing.

It is a simple routine that turns on the disk after the copy fails, but it doesn't move the disk arm. Then it reads the position of the disk head off a standard disk. I call the routine "Track Finder."

Track Finder is an easy program to make. (I let DOS supply most of the code.) To make it:

- 1) Boot a standard disk

PR#6

- 2) Enter the Monitor

CALL -151

- 3) Move the DOS address read routine to page 3

303<B942.B99FM

- 4) Add a little code of my own to turn the disk on and off and stuff

300:18 90 5E

**361:A2 60 AD 81 C0 BD 89
368:C0 A9 00 20 A8 FC 20 A8
370:FC 20 A8 FC 20 A8 FC 20
378:A8 FC BD 8E C0 20 05 03
380:BD 88 C0 90 03 4C 2D FF
388:A5 2E 8D FF 02 4C DA FD**

- 5) Save this new code

**BSAVE TRACK FINDER,AS300
,LS90**

That's it! To start up Track Finder, type "300G." If all goes well, Track Finder will print out the track number that the disk head is currently over. If there is an error, it will print "ERR."

To test it, CATALOG a disk and then run Track Finder. The drive head should be over track \$11.

The following is an example of how to use the Track Finder to find out where the nibble count for Cannon Ball Blitz by On-Line is.

- 1) Copy Cannon Ball Blitz with COPYA (any whole disk copier will do).

- 2) Load in Track Finder

BLOAD TRACK FINDER

- 3) Hide it in a place that won't be destroyed by Cannon Ball Blitz

**CALL -151
C089
C089
D000<300.390M**

- 4) Boot the copy of Cannon Ball Blitz.

C600G

After a couple of seconds, the disk arm will move into position to do the nibble count. (you can hear it.)

When the nibble count fails, the disk will reboot. Just before the reboot, Reset into the Monitor with your favorite method. DON'T BOOT ANY DISK. If you do anything that accesses the disk, it will change the position of the drive head.

- 5) Move Track Finder back to \$300

**C089
C089
F800<F800.FFFFM
C08B
300<D000.D090M**

- 6) Put a normal disk in the drive and execute Track Finder

300G

The disk should whirl for a second and then Track Finder will print the number of the track where the head is when the boot stopped. In this case, it should be "00." This is the track that must be bit-copied.

- 7) Bit copy track 0 with nibble count option engaged.

That's all there is to it! Track Finder is also a good way to find the track of those generic I/O errors that show up in regular DOS.



Breaking In

by Dave Templin

If you are confused by the procedures used in softkeys and mystified by how the author could ever figure out how to manipulate the disk so that he could make it copyable, then this article is for you. It is an explanation of some of the basic procedures used in many softkeys. The methods in this article will work most often for single load programs that can run in a 48K Apple II and never access the disk.

Before a meaningful explanation of unprotection can be discussed, it is necessary to have at least a small knowledge of machine language programming. For those of you who have no experience in this art and wish to learn how to unprotect your hard-to-back-up disks, it would be well advised to find a good book that explains, in a simple format, the basics of Apple assembly language programming. One that I have found particularly clear and easy to follow is *Apple II Assembly Language Exercises* available at most bookstores. Once you have acquired a book such as this, read through it carefully and you should gain a basic knowledge of the fundamentals of machine language.

At this point it is not necessary to fully comprehend the functions of all of the mnemonics, like ROL and BIT; you may find that you will understand most of what's going on inside a program after you have learned about working with the instructions that involve manipulation of the three registers, the Accumulator and the X and Y registers, and the instructions that involve branching to other parts of the program. When you have this knowledge, and even a very basic understanding will suffice, then you are ready to proceed.

General Procedures

Perhaps the easiest way of going about unprotecting a disk is to try and reset out of the program. Many programs will just reboot if you try pressing Reset. Note that if the disk reboots, you will not be able to list any Applesoft program that may have been present before the reset. Some programs, like The Print Shop or Karateka, will go through and clear the memory before re-booting. You can tell if it does this if you detect a pause between the time you press Reset and the time it reboots; if it doesn't reboot right away, forget about this method. However, if there is no indication of this, go ahead and press Reset again. This should stop the boot

process and turn off the disk drive. Then, enter the monitor with a CALL-151 and probe around in the memory by listing and executing to see if you can find the start of the program.

Some good places to look are \$07FD, \$0800, \$1000, \$2000, \$4000, and \$6000. If the program uses the hi-res page(s) then you can rule out the area from \$2000-\$3FFF and/or \$4000-\$5FFF. Notice that the start address will be less than \$BFFF because, if you'll recall, memory above that address is ROM and I/O space and thus unusable, unless the language card area is active. However, you may find that a program almost never starts at a location greater than \$8000. Note also that the start location doesn't always have to be at an even address such as \$800. It could, in all possibility, reside at location \$1F42, for example.

When you break out of a program and enter the monitor, probe around the memory and see if you can find where it starts. If you find where the program starts up, you can just boot up a normal DOS 3.3 disk and save the memory it resides in (more on this later). If this method fails to work, there are other but more complicated ways of going about it.

If the reset method didn't work, what you should do next is see if you can look at the disk with a sector editor. It would be a great idea to have a program that has a disk search option. Copy II Plus 5.0 and later versions have this option. Probably your best choice, however, is a program designed especially for these purposes, like CIA or Tricky Dick. If it is possible to read the disk without errors, there are several actions you can take to try and break into the code. You have it easy if you own a card that allows you to reset into the monitor, but I'm going to assume that you don't.

One thing you can try is to find the place on the disk where it stores new values for the reset vector. The reset vector is a three location pointer that is the cause of a protected program's nasty habits of rebooting, etc. Since the reset vector is located at locations \$03F2 to \$03F4, it makes sense that the programmer would have to address those locations in order to change them, right? Exactly. Here is how he might go about doing that:

```
LDA #$FD
STA $03F2
LDA #$FE
STA $03F3
LDA #$FF
STA $03F4
```

or it may look like this:

```
LDA #$FD
LDX #$FE
LDY #$FF
STA $03F2
STX $03F3
STY $03F4
```

Now, take a look at what this looks like in a disassembled format which shows the opcodes (numbers that represent the mnemonics) and the numbers which represent the operands:

```
A9 FD    LDA #$FD
A2 FE    LDX #$FE
A0 FF    LDY #$FF
8D F2 03 STA $03F2
8E F3 03 STX $03F3
8C F4 03 STY $03F4
```

Note that it won't always look exactly like that, but usually very similar. Note also that I left out the addresses where the program resides; that's because it won't always be the same. At this point you should notice that each instruction has one thing in common, and that is that the operands are all pretty much the same. Therefore, if you searched the disk for the bytes \$F2 and \$03 you would probably come up with the place where it modifies the reset vector. You should be aware that there are several instances where this might not work. The first of which is the idea that, since there are only two bytes in the sequence, it is entirely possible that the sequence appears by chance; that is to say that the disassembly would be full of "???" which indicates that the code makes no sense to the computer. In this case, leave the bytes alone. Secondly, it is possible that the reset modification is being done through BASIC, in which case it would be pretty hard to find from machine language.

Okay, assuming that you did find that sequence, what now? What you should do is replace each byte (one, two or three bytes) of the unwanted instruction(s) with the hexadecimal value \$EA, meaning NOP or "No Operation". This is called "NOPing" an instruction. If an instruction takes up three bytes, like STA \$03F2, replace it with three \$EAs. The program will, in effect, ignore the instruction. What this does is to nullify the code that performs the operation of storing values to the reset vector. One note of importance is to be sure to write your altered sectors back to the disk. **Note Of Caution: NEVER, NEVER make any changes to a disk of which you have only one copy of**, for the obvious reason

that it is most probable that you will irrevocably destroy the program. **If you must modify a protected disk, do it ONLY to a bit copy of a protected disk.** If you succeed in changing the bytes mentioned, you should be able to reset without the disk re-booting. However, this method is nowhere near one hundred percent reliable.

If the above mentioned method fails to work, there is another possibility. You can try to alter code that is on the disk in order to make the program halt and dump you into BASIC or the monitor. This can be done by finding viable code other than that code which appears on tracks \$0 to \$2 because it is probable that that code is DOS and it is likely that you don't want to change that. When you find a stream of viable code, that is, code that has no "???" in the disassembly, you can replace the values of some of the bytes with the value \$00 which happens to be the opcode for the mnemonic "BRK". Perhaps a safer method of accomplishing the same goal is to find a sequence of bytes that JMPs. For example:

```
JMP $6000
```

In disassembled format it would look like this:

```
4C 00 60 JMP $6000
```

If you changed the bytes after the value \$4C to be \$69 and \$FF you would get:

```
4C 69 FF JMP $FF69
```

As mentioned above, if location \$FF69 is executed, it will place you in the monitor.

Through whichever method you have used to enter the monitor, there is one more procedure to follow. What we need to do is to get the data in memory over into a normal DOS 3.3 format. There are basically two ways of doing this. One is to use a program such as Super IOB to read the protected data on the disk and write it back to another disk in normal format. This method is the best route to go if the program has a lot of disk access. If, however, the program is all loaded in at once, the following method is probably the best way to go.

First, determine whether the program is written in BASIC or machine language. A good way to determine this is to clear the entire RAM memory and boot the disk. This can be done by using the following procedure:

```
CALL -151
```

Store a \$00 at location \$7FD:

```
7FD:00
```

Copy the \$00 to most of memory using a "feature" of the monitor's Move command:

```
07FE<07FD.95F0M
```

Boot the program you're working with.

```
C600G
```

Once the program is loaded in, enter the monitor with a "CALL -151" if necessary and do this:

```
07FD.95F0
```

The screen will be filled with scrolling numbers. These are all bytes put into the memory from the disk you just booted; nothing

is left over from before because, if you'll remember, we had just cleared everything up to \$95F0. It may be necessary on occasion to clear the memory up to \$BFFF, but this sometimes makes the computer hang up making you reset to regain control. Usually, it will be OK to go ahead and boot the disk at that time. Okay, back to the scrolling numbers. There will be two sets of numbers that you will have to identify. The numbers to the left of the dash (-) are the addresses in which the set of values to the right reside in.

```
07FD- 00 A9 FF 20 00 60 A9 00
```

The above example means that the value \$00 is stored in location \$07FD, and the value \$A9 is stored in location \$07FE, and so on. Just sit back and watch the numbers go by. Where there are just zeros, it probably means that nothing was loaded there and you can ignore those addresses when it comes time to save the data to DOS 3.3. Write down the location where you see data start and where it ends. Once you have completed that task, you are ready for the final stage of unprotecting your disk!

Retrieving Applesoft Programs

In order to determine whether the program is written in BASIC or machine language, you should first try to list the program in BASIC. To do this, exit the monitor if in fact you are in the monitor; remember your prompts! Then, type LIST. If the program lists and there is little or no garbage, you have successfully determined that it is written in BASIC. From here it's very simple! You need to find the start and length of your Applesoft program. This is done by getting the values from location \$67, \$68, \$AF and \$B0. To do this, re-enter the monitor with a CALL -151 if necessary and type:

```
67.68
```

Two values will be printed right under your entry. They should look something like this:

```
0067- F7
```

```
0068- 08
```

Write down the values following the addresses. Then type:

```
AF.B0
```

Write the values that are printed after "00AF-" and "00B0-". Next, you want to move the data of your program into a safe place in memory so you can boot the slave disk. If the program is large, change the size of the area that you move accordingly.

```
4000<800.3FF0M
```

Boot the slave disk. Make sure that it has a short or deleted HELLO program.

```
C600G
```

Type NEW from BASIC. Enter the monitor and type:

```
800<4000.6000M
```

Restore the values that you had written down (xx stands for the numbers you wrote down above):

```
67: xx
```

```
68: xx
```

```
AF: xx
```

```
B0: xx
```

Exit the monitor:

```
3D0G
```

And presto! You have just unprotected your Applesoft program! Now all that remains to be done is to save the program. If the program does not execute properly, it is likely that it is trying to access some machine language subroutines that you need to go back and get. Just go back and boot the disk and pull the data out of the memory in the same fashion, only use the method described below.

Retrieving Machine Language

If your program was not written in Applesoft or if your Applesoft program had some machine language subroutines, then what you need to do is to go back and look at the locations you wrote down when you dumped the memory. You may need to look at your Applesoft program to locate the machine language that it needs (PEEKs, POKEs and CALLs). Then, simply determine if it needs to be moved so as to keep it from being written over when you boot your DOS 3.3 disk. To do this, determine whether or not your program resides in locations before \$900 and/or after \$95FF. If it does, move it somewhere between the above mentioned locations. To do this type use the format below. Note that the text between the brackets are intended to represent hexadecimal values.

```
destination < first.last M
```

for example:

```
4000<2000.3FFFFM
```

This will move all the data that resides between locations \$2000 and \$3FFF to the locations between \$4000 and \$5FFF. Now that your data is safe, boot a DOS 3.3 disk and re-enter the monitor. Then, put your program back where it's supposed to be and save it to disk with the BSAVE command. This is done by typing:

```
BSAVE filename,A$start,L$length
```

Once that's done, you're done! You will have an unprotected machine language program.

Please note that when moving data out of the way to boot a DOS 3.3 disk, you may have to segment your memory moves in case the data is too large for one memory move. For example, if your data takes up most of the memory between locations \$800 and \$9FFF you will quickly see that you will have nowhere to put your data. So what you will have to do is to take a portion of the data and move it to a safe area, boot the DOS 3.3 disk, save the data, reboot the protected disk, get the rest of your data into a safe area, and boot your DOS 3.3 disk again and save the data. Then, when it comes time to use your program, simply load both files and start the program.

The methods described have been utilized in many COMPUTIST softkeys in past issues. Through practice and application of these methods, you should be able to derive your own ways of unprotecting disks in a short time.

Copy II Plus

by Jerry D. Greer

Copy II Plus Version 6.0
Central Point Software Inc.
9700 SW Capitol Hwy. #100
Portland, OR 97219
503-244-5782
\$39.95

Requirements:

Minimum 64K Apple II
At least one disk drive

I think that it may have been Jack London who said that people may be judged as well by their enemies as by their friends. When the national Software Publishers Association (SPA) voted to deny Central Point Software certification for having a successful product, I think that they showed us what most of the software publishing industry really thinks about software purchasers. The Copy II Plus series qualified for the award by selling more than 100,000 copies.

Despite this snub by the SPA, the Copy II Plus series of programs will continue to meet the needs of legitimate and honest users. As soon as Central Point announced that version 6 was out, I called up and ordered my copy through their user upgrade program. As usual, they sent me the program almost on the next day. Not as usual, the copy that I received would not boot. There was obviously a problem with the disk formatting because it couldn't get past track \$0 sector \$0. I whipped out a quick note explaining the nature of the problem. They practically hand delivered the replacement copy to me because I got it faster than the original! Read this to mean that in my experience with Central Point, they have repeatedly given me excellent support.

The Utility Features

Many readers of COMPUTIST are familiar with the Copy II Plus program. It has been

around now for several years. For readers who have not tried it and for those who are just getting started in their study of Apple computer programs, a brief overview of what the program does is appropriate.

Copy II Plus is a disk utility system that provides global opportunities to manage your program disks and files. The different programs on the disk can be divided into two major groups, utilities and copy procedures.

The utilities give you a method to catalog your disks to determine contents. Any subdirectories on ProDOS formatted disks are identified on screen so you may catalog them just like you do the directories. The format utility will format disks in either DOS 3.3 or in ProDOS. They provide dependable ways to lock, unlock, rename, delete, undelete and verify files.

You can easily change the boot-up ("hello") program on DOS 3.3 disks. The disk map routine will show you how programs are stored by sector (or block) on the disk. You can even verify that the surface of a disk is free of errors and defects. This is especially important if you use the reverse side of single sided disks like I do. One utility permits you to accurately set the speed of your drives. Use this to make adjustments to your drives before tackling those speed sensitive copy routines.

The copy programs are recognized as being some of the best available. Using these will help you to make backup copies of many of your program disks and file disks. Unprotected disks and files are quickly copied to other disks or to hard and RAM disks. Programs that are protected against copying can be duplicated by using either the automatic bit copy routines with system parameters supplied on disk or by using manual bit copy routines that require you to determine and set the parameters.

Some Things Have Been Added

The first point that I should make is that the disk is now in the ProDOS format. Several features have been added to help with the management of ProDOS based disks. They have provided a "set date" feature so that changes to files will be date stamped. The program

recognizes and supports the new Apple 3.5 inch Unidisk, any hard disk on your system, and the ProDOS set RAMdisk. I have already mentioned the ability to display all of the subdirectories on ProDOS volumes. When copying files, this is a critical feature to have. You can now rename ProDOS volumes (and that can be a real time saver). Users are given the option of choosing either 40 or 80 column displays. Of course, many more sets of copy parameters have been added. Now that ProDOS is supported, many of the ProDOS based programs have been included in the programs library of parameters.

A Few Things Have Been Changed Or Moved

A couple of significant changes have been made. The sector editor is now in the bit copy program area. Old habits die hard and I still search for it on the main menu.

The new disk map is excellent. The previous versions of the track-sector maps were good but, you had to write down or recall the file names after they were displayed just before the map took over the screen. The new one displays the file name at top of the screen as the files are accessed for display or plotting. It is always clear where your files are stored on the disk.

Some Version 5 Options Have Been Deleted

We will miss some things but not others. For example, I never did have any use for the previous versions' ability to handle DOS 3.2. Some of you will continue to have a need for this and should hold onto a copy of version 4 or 5. The lack of support for DOS 3.2 should pose no problem for most of us.

Of more immediate concern is the fact that the new version does not provide for automatic file following in the sector editor. It will be missed and is another reason to keep a copy of the previous version in your toolbox. In my opinion, deleting the auto follow feature is a serious loss to workers at the assembly language level. Hold onto your version 5! Without this feature, dumping a disassembled list to the printer isn't of much use.

V 6.0

You may end up with a series of disconnected 20 line listings. You can still do a manual file trace but it is easier to pull out your old version.

Version 6 will not help to verify that two files are identical and you will not be able to fix file sizes on disks.

Problems and Distractions

There are a couple of minor distractions. The "Hi-Res Disk Scan" seems painfully slow. The comparable function in Locksmith 5.0 runs a little more than twice as fast. In a test on a \$22 track disk, Locksmith laid out the display in 25 seconds. Copy II Plus took 55 seconds to do the same job. Perhaps the menu should list it as "Sort-of Quick Scan". Still, it is a very nice addition to the Copy II Plus program.

Can Version 6 be used with DOS 3.2? On page 7 of the generally well written manual, we find the statement, "The Utilities will not work with DOS 3.2 (13 sector) disks." I got confused when I got over to page 82 while reading about the Patch option. Here I find that "Normally the Sector Editor can read only standard 16-sector (DOS 3.3, ProDOS, etc.) or 13-sector (DOS 3.2) disks." The Patch option menu seems to indicate that DOS 3.2 disks can be read. A quick call to the helpful lady at Central Point revealed that the sector editor in this version IS able to read and write DOS 3.2 disks!

Summary

First of all, you need to consider getting Version 6 of Copy II Plus for your library if you do not already have it. However, you probably don't want to throw away your versions 4 and 5 of this very useful utility program. You will want to keep them for the rare occasions when a DOS 3.2 disk comes your way. If you do any automatic file following when you are doing sector edits or if you disassemble programs, you will need one of the earlier versions for that too. I have many occasions when I want to verify that two files are identical and my version 5 is coming out of the envelope nearly as often as it was before. You will also need to keep them handy for the times when you want to fix file sizes.

Considering the memory limitations that the Central Point Crew had to contend with when writing to cover both DOS 3.3 and ProDOS, I think they did an outstanding job. Even though several things can be picked out as problems, they have given us an improved and valuable product. In my opinion, Central Point has a right to proclaim on the Manual Cover that it is a Gold Medal Winner. Central Point, Please consider this your GOLD STAR from all the honest and legitimate users who depend upon Copy II Plus!

P. S.

After this review was written, Central Point came out with an updated version that they called 6.5. When I recieved my update notice in the mail, I called the helpful lady at Central Point and she sent me the updated version. By this time the version number was up to 6.6.

A quick review of the new disk revealed that almost all the changes are transparent to the average user. You will see little that will let you know that it is changed in any way. A change was made in the routine that does a sort in the ProDOS files. The problem was of so little consequence to me that I didn't even mention it in the review.

On this new version you will notice a couple of changes. First, when you load up the Bit Copy routine, you will be asked to specify which slot the program is operating from. Nearly everyone will simply enter "6" and the Bit Copy menu will come up. However, you can specify any other legal slot. This will apply to users who may have their disk controller card in some slot other than 6.

Also, when you left the Bit Copy program in version 6.0, you were required to re-enter the date for the utilities program. With 6.6, the date that is entered on boot-up is remembered and this small task is eliminated.

The program appears otherwise like its previous version. Current owners are given the opportunity to purchase the most recent Parameter Update for \$5 and this includes the latest version of Copy II Plus! New purchasers will receive the most recent version if ordered from Central Point.

My job has given me the opportunity to use systems from "those other people," you know who I mean. I'm not free to mention names, so let's just call one Big Blew and the other TRaSh Inc. The computers made by these companies have an extended list of opcodes that have been (until now) kept secret from the general population. I accidentally came across a copy of these codes and felt it my duty to expose them to the world. Hopefully they will explain many of the strange occurances any of you may have had while using these systems.

*** Sir William ***

BAH	Branch And Hang
AIM	Add IMproper
DAO	Divide And Overflow
CMA	Circulate Memory Array
ARZ	Add and Reset to Zero
SRZ	Subtract and Reset to Zero
RIV	Read InValid
RPR	Read from PRinter
WWR	Write Wrong Record
EJD	Eject Disk
RWD	ReWind Disk
BSD	BackSpace Disk
RDD	Reverse Drive Direction
AWF	Access Wrong File
UER	Update and Erase Record
MDB	Move and Drop Bits
MLR	Move and Lose Records
PBR	Play Beatles Record
OTL	Out To Lunch
MCT	Move ConTinuously
HCF	Halt and Catch Fire
CVU	ConVert to Unary
CRN	Convert to Roman Numerals (Italian models only)
ARN	Add Roman Numerals (Italian models only)
BAE	Branch on Almost Equal
LCC	Load and Clear Core memory
EPI	Execute Programmer Immediately
BVS	Blankout Video Screen
BRW	BRanch on Whim
JRL	Jump to Random Location
IOI	Ignore Operator Input
RDF	Randomly Destroy Files
GNL	Goto Nonexistent Line
UDH	Unalign Drive Head
MFN	Make Funny Noise
DTS	Display Television Show
BST	Backspace and Stretch Tape
RBT	Rewind and Break Tape
SPS	Scramble Program Status
ERS	Erase ROM Storage
GVS	Generate Voltage Spike
SRSD	Seek Record and Scar Disk
DWIM	Do What I Mean
EI	Execute Improperly
EIO	Execute Invalid Opcode (Old McDonald's Opcode)

Sylk to Dif

by D. W. Walkey

All spreadsheets were not created equal. Each has features that endear it to its devoted followers, or make it better suited to a given task. Microsoft's **Multiplan** is my favorite with its abilities to sort and transfer data between spreadsheets. Another feature of Multiplan is its transportability between the 64K Apple II Plus and the 128K Apple IIc.

A major drawback of Multiplan is its inability to use DIF files. DIF, or Data Interchange Format, files are a sort of industry standard for transferring information between programs. A wide variety of programs support DIF files ranging from **Visicalc** and **Visifile** to graphic packages, data bases, even other computers. (Can I mention IBM in this magazine?) (Ed. note: Nope.)

Multiplan on the other hand uses SYLK files or Symbolic Linking Code. Where DIF files transfer tabular data in a very strict format, SYLK files can transfer an entire spreadsheet between programs, formulas and all. They vary in both philosophy and in structure, and that's where the problems begin.

Using a top-of-the-line spreadsheet is one thing, but for many applications the ability to transfer the information to a data base program or a graphics package is a real asset. I contacted Microsoft about information on SYLK files, or a conversion program and after a long wait I received a very polite reply. They wished me the best of luck and couldn't help me. Thanks anyway.

The only remaining course of action was to write my own program. The resulting program, **CONVERT**, will change a SYLK file to a DIF file and back again if you want it to.

How The Program Works

Spreadsheet information is arranged in a grid format. The information can take three forms, formulas, labels, or numbers. We are interested in the transfer of data in particular, so the formulas are unimportant.

DIF files store information from that grid. The data (meaning only numbers or labels) is arranged in "tuples and vectors" which relate to columns and rows. DIF files can be created in two ways, by column or row, which affects the final layout of the file.

SYLK files by comparison always are created by column. They always include the entire spreadsheet, and they always include any formulas included in the sheet.

Given these differences, it's actually an easy conversion. For example, to convert SYLK to DIF all you do is:

- 1) Create the SYLK file using Multiplan
- 2) Load the data from the file into a memory array
- 3) Dispose of the unnecessary formulas, leaving only numbers and labels.
- 4) Print out a new file in DIF layout.

Conversion back to SYLK is actually easier. Read the DIF file (which is just about all data anyway) into the array, then create the new SYLK file. Much of the SYLK file formatting (i.e. formulas) is optional, so all we need is the coordinates of the number or label and the value involved.

One Last Secret

The secret of discovering the secrets of text data files, is to load them into your favorite word processor. You can see all their protocols, formats, special characters, everything you need! There you have it, straight, clean, and simple. Good luck.

SYLK/DIF Converter

```
10 REM *****
20 REM * SYLK <-> DIF *
30 REM * FORMAT CONVERTER *
40 REM *
50 REM * BY DOUG WALKEY, BSC. *
60 REM * COPYRIGHT 1986 *
70 REM * SOFTKEY PUBLISHING *
80 REM *****
90 GOTO 1100 : REM TO MENU
100 HTAB (40 - LEN (T$)) / 2 : PRINT T$ : NORMAL
    : RETURN : REM CENTERING ROUTINE
110 REM GET X, Y ROUTINE (SYLK)
120 X = 0 : Y = 0 : FOR Z = 1 TO LEN (Z$) : Z1$ = MID$
    (Z$, Z, 4) : IF LEFT$ (Z1$, 1) = "Y" THEN
    Y = VAL (RIGHT$ (Z1$, LEN (Z1$) - 1))
130 IF LEFT$ (Z1$, 1) = "X" THEN X = VAL (RIGHT$
    (Z1$, LEN (Z1$) - 1))
140 IF LEFT$ (Z1$, 1) = "K" THEN Z = LEN (Z$)
150 NEXT Z : RETURN
160 AC$ = ""
170 GET A$ : IF A$ = CHR$ (13) OR LEN (AC$) = 254
    THEN RETURN : REM EXIT LOOP IF DONE
180 AC$ = AC$ + A$ : GOTO 170 : REM
    ELSE CONTINUE READING
190 REM WAIT FOR Y/N
200 GET A$ : IF NOT (A$ = "Y" OR A$ = "N") THEN 200
210 RETURN
220 REM *** SYLK TO DIF
230 TEXT : HOME : VTAB 4 : HTAB 15 : INVERSE :
    PRINT "SYLK^ TO^ DIF" : NORMAL
240 VTAB 7 : HTAB 1 : PRINT "SYLK^ FILE^ NAME:^
    " : INPUT "" : SFIL$
250 INPUT "BEGINNING^ ROW:^ " : BROW : IF BROW <
    1 OR BROW > 253 THEN PRINT CHR$ (7) : GOTO
    250
260 INPUT "BEGINNING^ COLUMN:^ " : BCOL : IF BCOL
    < 1 OR BCOL > 62 THEN PRINT CHR$ (7) : GOTO
    260
270 INPUT "ENDING^ ROW:^ " : EROW : IF EROW < BROW
    OR EROW > 254 THEN PRINT CHR$ (7) : GOTO 270
```

```

280 INPUT "ENDING^ COLUMN:^ " ; ECOL : IF ECOL <
BCOL OR ECOL > 63 THEN PRINT CHR$ (7) : GOTO
280
290 INPUT "DIF^ FILE^ NAME:^ " ; DFILES : IF
RIGHT$ (DFILES , 4) <> ".DIF" THEN DFILES$
= DFILES$ + ".DIF"
300 PRINT "USE^ THESE^ VALUES?^ (Y/N)" ; : GOSUB
200
310 IF AS$ = "Y" THEN 350
320 HTAB 1 : CALL - 868 : PRINT "RETURN^ TO^
MENU?^ (Y/N)" ; : GOSUB 200
330 IF AS$ = "Y" THEN 1100
340 GOTO 230
350 NROW = EROW - BROW + 1 : NCOLUMN = ECOLUMN -
BCOLUMN + 1
360 HOME : VTAB 10 : FLASH : PRINT "CONVERTING^
" SFILES$ " TO^ " DFILES$ : NORMAL
370 VTAB 21 : CALL - 868 : TS$ = "READING^" + SFILES$
: INVERSE : GOSUB 100
380 REM ** READ SYLK FILE
390 PRINT CHR$ (4) ; "OPEN" SFILES : PRINT CHR$
(4) ; "READ" SFILES
400 GOSUB 160 : IF LEFT$ (AC$ , 1) <> "B" THEN 400
: REM SEARCH FOR BOTTOM RECORD
410 Z$ = AC$ : GOSUB 120 : ROW = Y : COL = X : REM
TOTAL NUMBER OF CELLS
420 DIM ARRAY$(ROW , COL) : REM
CREATE DATA ARRAY
430 FOR I = 1 TO ROW : FOR J = 1 TO COL : ARRAY$(I
, J) = CHR$ (34) + CHR$ (34) : NEXT J , I
440 GOSUB 160
450 IF LEFT$ (AC$ , 1) = "E" THEN 510 : REM
END OF FILE
460 IF LEFT$ (AC$ , 1) <> "C" THEN 440 : REM
NEXT RECORD
470 Z$ = AC$ : GOSUB 120 : IF Y > 0 THEN ROW = Y :
REM GET CO-ORDINATES
480 IF X > 0 THEN COL = X
490 FOR Z = 1 TO LEN (AC$) - 1 : Z1$ = MID$ (AC$ , Z
, 2) : IF Z1$ = " ; " ; K" THEN ARRAY$(ROW , COL)
= RIGHT$ (AC$ , LEN (AC$) - Z - 1) : Z = LEN
(AC$) - 1
500 NEXT Z : GOTO 440
510 REM FILE CONVERTED TO MEMORY
520 PRINT CHR$ (13) ; CHR$ (4) ; "CLOSE" SFILES$
530 VTAB 21 : HTAB 1 : CALL - 868 : TS$ = "WRITING^
" + DFILES$ + CHR$ (7) : INVERSE : GOSUB 100
540 REM *** WRITE DIF FILE
550 PRINT CHR$ (4) ; "OPEN" DFILES : PRINT CHR$
(4) ; "WRITE" DFILES
560 PRINT "TABLE" : PRINT 0 , " , " 1 : PRINT CHR$ (34
) ; CHR$ (34)
570 PRINT "VECTORS" : PRINT 0 , " , " NCOLUMNS :
PRINT CHR$ (34) ; CHR$ (34)
580 PRINT "TUPLES" : PRINT 0 , " , " NROWS : PRINT
CHR$ (34) ; CHR$ (34)
590 PRINT "DATA" : PRINT 0 , " , " 0 : PRINT CHR$ (34
) ; CHR$ (34)
600 PRINT - 1 , " , " 0
610 FOR I = 1 TO NROWS : REM
ONCE FOR EACH TUPLE OR ROW
620 PRINT "BOT" : REM BEGINNING OF TUPLE
630 FOR J = 1 TO NCOLUMNS : REM
ONCE FOR EACH VECTOR OR COLUMN
640 IF LEFT$ (ARRAY$(I + BROW - 1 , J + BCOL - 1)
, 1) = CHR$ (34) THEN PRINT 1 , " , " 0 : PRINT
ARRAY$(I + BROW - 1 , J + BCOL - 1) : GOTO 660
650 PRINT 0 , " , " VAL (ARRAY$(I + BROW - 1 , J + BCOL
- 1)) : PRINT "V"
660 NEXT J : PRINT - 1 , " , " 0 : REM END OF TUPLE
670 NEXT I
680 PRINT "EOD" : PRINT CHR$ (4) ; "CLOSE" DFILES$
690 GOTO 1100 : REM COMPLETED

```

```

700 REM DIF TO SYLK
710 TEXT : HOME : VTAB 4 : INVERSE : TS$ = "DIF^ TO^
SYLK" : GOSUB 100 :
720 VTAB 7 : HTAB 1 : INPUT "DIF^ FILE^ NAME:^ "
: DFILES
730 VTAB 9 : HTAB 1 : INPUT "SYLK^ FILE^ NAME:^ "
: SFILES
740 IF RIGHT$ (SFILES , 5) <> ".SYLK" THEN
SFILES$ = SFILES$ + ".SYLK"
750 VTAB 11 : HTAB 1 : PRINT "USE^ THESE^ FILES?^
(Y/N)" ; : GOSUB 200
760 IF AS$ = "Y" THEN 800
770 VTAB 11 : HTAB 1 : CALL - 868 : PRINT "RETURN^
TO^ MENU?^ (Y/N)" ; : GOSUB 200
780 IF AS$ = "Y" THEN 1100
790 GOTO 700
800 VTAB 11 : HTAB 1 : CALL - 958 : FLASH : PRINT
"CONVERTING^ " DFILES$ " TO^ " SFILES$ :
NORMAL
810 VTAB 13 : HTAB 1 : CALL - 868 : TS$ = "READING^
" + DFILES$ : INVERSE : GOSUB 100
820 REM ** READ DIF FILE
830 PRINT CHR$ (4) ; "OPEN" DFILES : PRINT CHR$
(4) ; "READ" DFILES
840 FOR I = 1 TO 5 : GOSUB 160 : NEXT I : REM
READ TO VECTOR NUMBER
850 AC$ = RIGHT$ (AC$ , LEN (AC$) - 2) : NCOL =
VAL (AC$)
860 FOR I = 1 TO 3 : GOSUB 160 : NEXT I : REM
READ TO TUPLE NUMBER
870 AC$ = RIGHT$ (AC$ , LEN (AC$) - 2) : NROW =
VAL (AC$)
880 DIM ARRAY$(NROW , NCOL)
890 ROW = 0 : COL = 0
900 GOSUB 160 : COL = COL + 1 : IF LEFT$ (AC$ , 3)
= "BOT" THEN ROW = ROW + 1 : COL = 0 : GOTO 900
910 IF LEFT$ (AC$ , 3) = "EOD" THEN 950
920 IF LEFT$ (AC$ , 1) = "1" THEN GOSUB 160
: ARRAY$(ROW , COL) = AC$ : GOTO 900
930 IF LEFT$ (AC$ , 1) = "0" THEN ARRAY$(ROW , COL)
= RIGHT$ (AC$ , LEN (AC$) - 2) : GOSUB 160
940 GOTO 900
950 PRINT CHR$ (13) ; CHR$ (4) ; "CLOSE" DFILES$
960 VTAB 13 : HTAB 1 : CALL - 868 : TS$ = "WRITING^
" + SFILES$ : INVERSE : GOSUB 100
970 REM WRITE SYLK FILE
980 PRINT CHR$ (4) ; "OPEN" SFILES : PRINT CHR$
(4) ; "WRITE" SFILES
990 PRINT "ID;PMP"
1000 PRINT "B;Y" NROW ; X" NCOL
1010 FOR I = 1 TO NROW : K = 0 : FOR J = 1 TO NCOL
1020 IF ARRAY$(I , J) = CHR$ (34) + CHR$ (34)
THEN 1060
1030 PRINT "C ; " ;
1040 IF NOT K THEN PRINT "Y" I " ; " ; : K = 1 : REM
PRINT Y CO-ORDINATE
1050 PRINT "X" J " ; " ; K" ; ARRAY$(I , J)
1060 NEXT J , I
1070 PRINT "E" : REM END OF FILE
1080 PRINT CHR$ (4) ; "CLOSE" SFILES$
1090 PRINT CHR$ (7) : GOTO 1100
1100 TEXT : HOME : CLEAR : REM MAIN MENU
1110 VTAB 6 : TS$ = "SYLK^ <->^ DIF^ CONVERTER" :
INVERSE : GOSUB 100 : I = 1
1120 VTAB 10 : HTAB 1
1130 IF I = 1 THEN INVERSE
1140 TS$ = "SYLK^ ->^ DIF" : GOSUB 100 : NORMAL
1150 IF I = 2 THEN INVERSE
1160 PRINT : TS$ = "DIF^ ->^ SYLK" : GOSUB 100 :
NORMAL
1170 IF I = 3 THEN INVERSE
1180 PRINT : TS$ = "EXIT" : GOSUB 100 : NORMAL
1190 VTAB 20 : PRINT "SELECT:^ ^ " ; : GET AS
1200 IF AS$ = CHR$ (13) THEN ON I GOTO 230 , 700

```

```

, 1260
1210 IF AS$ = "S" THEN I = 1
1220 IF AS$ = "D" THEN I = 2
1230 IF AS$ = "E" THEN I = 3
1240 IF AS$ = "^" OR AS$ = CHR$ (8) OR AS$ = CHR$ (21
) THEN I = I + 1 : IF I = 4 THEN I = 1
1250 GOTO 1120
1260 TEXT : HOME : VTAB 10 : PRINT "... THAT'S^
ALL^ FOLKS!" : END

```

checksums

10	- \$BADD	640	- \$B1BB
20	- \$9B13	650	- \$D5E6
30	- \$4D3B	660	- \$19AF
40	- \$AD92	670	- \$1200
50	- \$C899	680	- \$2D17
60	- \$FF65	690	- \$33DD
70	- \$A3BF	700	- \$76B8
80	- \$A900	710	- \$4021
90	- \$252D	720	- \$1151
100	- \$7BD4	730	- \$B545
110	- \$560D	740	- \$0299
120	- \$7F17	750	- \$1A43
130	- \$69D0	760	- \$039C
140	- \$03A6	770	- \$B574
150	- \$A13D	780	- \$4D75
160	- \$192E	790	- \$1750
170	- \$8457	800	- \$041E
180	- \$8AE8	810	- \$0C16
190	- \$5288	820	- \$4B58
200	- \$3C92	830	- \$08D6
210	- \$C37D	840	- \$3A41
220	- \$533A	850	- \$0A03
230	- \$B031	860	- \$A0BB
240	- \$8068	870	- \$6713
250	- \$785A	880	- \$D45E
260	- \$DA6D	890	- \$266D
270	- \$7C87	900	- \$968E
280	- \$37C2	910	- \$D9DC
290	- \$0BC1	920	- \$83B6
300	- \$CB68	930	- \$AC31
310	- \$C0A1	940	- \$5BA7
320	- \$5F4E	950	- \$BB96
330	- \$F230	960	- \$0512
340	- \$F3B3	970	- \$EF1E
350	- \$4F3E	980	- \$B78A
360	- \$C434	990	- \$659C
370	- \$FA15	1000	- \$3F9A
380	- \$D126	1010	- \$01DA
390	- \$3976	1020	- \$10BA
400	- \$99C8	1030	- \$92BE
410	- \$B3CB	1040	- \$1DE2
420	- \$FB44	1050	- \$F181
430	- \$84E6	1060	- \$0FA8
440	- \$7B32	1070	- \$0676
450	- \$458D	1080	- \$2335
460	- \$017D	1090	- \$D7BB
470	- \$6FBB	1100	- \$03ED
480	- \$3BE3	1110	- \$92E6
490	- \$E24C	1120	- \$60F1
500	- \$50E8	1130	- \$75B7
510	- \$809B	1140	- \$2CBB
520	- \$EC88	1150	- \$DEDA
530	- \$0ADE	1160	- \$F320
540	- \$3173	1170	- \$22B3
550	- \$32E2	1180	- \$176E
560	- \$B638	1190	- \$B26A
570	- \$708A	1200	- \$D653
580	- \$2BD1	1210	- \$67CD
590	- \$1803	1220	- \$C7E9
600	- \$F968	1230	- \$1AED
610	- \$C510	1240	- \$CC8F
620	- \$7B41	1250	- \$FA5B
630	- \$525C	1260	- \$D963

a tool for examining protected disks...

The DOS

by J. J. Gifford

Perhaps one of the greatest advantages in subscribing to COMPUTIST is the continual flow of programs and utilities that are printed. Some of my most used software tools were taken from the pages of COMPUTIST. Without such programs as DiskEdit, DiskView, Armonitor and the Lone Catalog Arranger, I would be forced to either manually perform the laborious tasks they simplify, or make do with lesser quality utilities.

I use DiskEdit to examine and debug disks that I attempt to crack. Unfortunately, DiskEdit suffers from one serious flaw; it lacks the ability to read disks with altered address or data marks. Thus I often am stuck using different sector editors that I do not find as powerful in other respects when I want to examine the code on disks protected by altered marks. In a final act of rebellion, I decided to create a program that would allow me to use DiskEdit, as well as all my other disk related utilities. Thus was the DOS Alterer born.

Entering the Program

To enter the DOS Alterer, simply type in the listing which follows this article. Check for typographical errors by using Checksoft. Then save it under the name "DOS ALTERER"

```
SAVE DOS ALTERER
```

What Does It Do?

Simply put, the DOS Alterer shows the user what the current address and data marks are, and allows the user to change them. Then, after the user has altered DOS to suit his needs, the program saves the changes as a single sector, EXECable text file. Thus, in order to use DiskEdit on a disk with altered marks, you must:

```
RUN DOS ALTERER
```

Change DOS to conform to the marks on the protected disk.

Have DOS Alterer save the changes as a text file (option 8).

```
LOAD DISKEDIT  
EXEC (name of DOS change file)  
RUN
```

The user may now examine and edit the protected disk!

How Does It Work?

When run, the DOS Alterer first PEEKs several locations in DOS memory in order to determine the current marks. It then converts these decimal numbers into their hexadecimal equivalents. After it has performed these preparatory functions, it goes to the main menu. At the top of the screen is displayed the header, below which is the word, "OPTIONS:" followed by eight choices, and then the prompt, "OPTION # =>". The status lines at the bottom of the screen show the current hexadecimal values for address header, address end, data header, and data end.

From this menu, you may choose to either: alter address markers; alter data markers; ignore read errors; alter sync bytes; alter slot, drive; ignore checksums; catalog disk; create a text file containing the altered values; or reset DOS to its standard values. If you wish to exit the program, a simple press of the ESC key from the main menu will clear the screen and replace the menu with a prompt to confirm the exit, as exiting restores the DOS values to normal, destroying any changes that may have been made. The next few sections discuss each option in detail.

A Note of Warning

Before I describe each option, I wish to present a short note of warning. The computer always rewrites the data sync bytes, header, end marks, and checksum for every sector written to a disk. It ignores those of the address field after the disk has been initialized. Keep this in mind when writing to a disk.

Options 1 and 2: Alter Address Markers and Alter Data Markers

If the disk you are examining is protected by address or data marks that deviate from the DOS 3.3 standard of D5 AA 96...DE AA (address) or D5 AA AD...DE AA (data), you may alter standard DOS to match the marks on the protected disk by choosing these options. Let's assume for example that you wanted to alter the address marks of a disk. After pressing "1" followed by "Return" at the main menu, you are greeted by a screen that displays the current address or data header in inverse and asks what you would like to alter it to. Should you wish to simply leave it and only change the address end marks, press ESC and the screen will be replaced by a similar screen showing the current address end marks. Pressing "Esc" at the second screen will return you to the main menu. To alter either the header or end marks merely enter the desired values at the appropriate screen and press Return. Be sure you enter all values in hexadecimal (base sixteen) notation.

Option 3: Ignore Read Errors

Some disks have altered epilogue values that change from track to track or even sector to sector. In order to read a disk that is protected in this manner, you might wish to tell DOS to ignore any errors it encounters while reading the disk. This option allows you to do that. Thus many such disks can be read. Caution should be exercised in the use of this option as it overrides DOS's safety mechanism and could allow scrambled or incorrect data to pass as valid. One way to be sure the data is correct is to read the sector being examined over and over, checking that the data returned is consistent.

When this option is chosen, the screen displays the current read error status (set/ignore) in inverse and allows you to toggle

Alterer

its status. Pressing Esc returns you to the main menu without altering the read error status. This option is powerful but risky and imprecise.

Option 4: Alter Sync Bytes

This option allows you to change the sync bytes that are written between sectors and the address and data fields within sectors. These bytes do not generally cause trouble, but occasionally they can confuse the drive and make reading the disk difficult. The data sync bytes are written out to the disk every time a sector is written and, if a protected disk senses the presence of normal sync bytes, could prevent a protected disk from booting correctly. To change either the data or address sync bytes, select option 4 from the main menu and then respond to the prompts. "Esc" advances you to the next screen or to the main menu, as with the other options.

Option 5: Alter Slot, Drive

This option is of no use towards reading protected disks but simply lets you alter the default slot and drive on which to save the text file containing your changes. It displays the current values and allows you to input new ones. "Esc" sends you back to the main menu without altering the default slot or drive.

Option 6: Ignore Address Checksums

Sometimes you will have been able to discover all of the changes a protected disk's DOS requires, yet still be unable to read the disk. This is usually due to altered checksums. The checksums provide DOS with a way to be sure the data and address has been correctly written to or read from the disk. Rather than try and figure out how the protected disk arrived at its checksums, it is easier to simply ignore them. This allows many disks to be read. This option will tell DOS to ignore the Address Checksums on the protected disk. It will not, in its current stage, allow DOS to ignore the Data Checksums.

If you wish to alter the program to ignore

the data field checksum, read the sections describing the requirements and operations of each subroutine and write your own subroutine to ignore the checksums. Be sure your subroutine incorporates well with the rest of the program; it should be easy. Be careful: ignoring the data field's checksums can sometimes allow erroneous information to slip by. Follow the procedure outlined under the description of option 3 to attain some degree of confidence in the data returned. It is not as risky as ignoring read errors, however. Ignoring both read errors and data checksums is very dangerous and should always be avoided.

The program displays the current status for the address checksums and allows you to toggle the status similar to the manner in other options. Again, to the main menu. I use this option quite frequently and have found it to be powerful as a semi-"shotgun" approach to reading protected disks.

Option 7: Catalog Disk

This is self explanatory. Shows the contents of the disk in the default drive. This option may also be used to verify if the current marks match those on the protected disk, providing the disk has a normal catalog.

Option 8: Create Text File

Select this option after you have customized DOS to suit your needs and wish to save the changes to disk. In order to prevent the file from exceeding one sector (any longer file would alter DOS before it had a chance to read in the second sector and thus be unable to do so), the program analyzes what changes have been made to DOS and only saves the deviations from normal DOS. Thus if only the address marks have been altered, the program will only save the address marks and ignore the rest of DOS.

In its first generation, the program saved the changes as a series of POKE commands. I realized that, even with the screening described above, it would be possible to create a combination of changes that would require more than one sector. Thus the text file, upon being

EXECed, enters the monitor and enters the changes as a series of direct monitor commands. This ensures that all the data will fit in one sector, with room to spare.

Be sure you save your changes, even if you plan to use the altered DOS immediately, because the program resets DOS to standard values when it is exited.

Option 9: Normalize DOS

Again, this is self explanatory. This option resets DOS to the standard values in case you wish to restart from scratch.

The Routines

The following is a description of the actions and requirements of each of the program's routines. Unless you are interested in the more technical aspects of the program or plan to write your own routines to augment it, skip this section.

Initialization

Here the program, through a series of PEEKs, determines the current DOS values. To convert the Address, Data, and Sync marks to hex, it accesses a routine beginning at 2680. It then assigns the variables their appropriate values.

Main Menu

This routine prints a header on the top line, preserves it from destruction by a POKE 34,1 and prints the main menu. At the bottom of the screen are displayed the current values for the Address and Data marks, in hex. In order to protect the bottom two lines from destruction, the program performs a POKE 35,22. Then the program prints the various options, gets the user's input and GOSUBs the appropriate routine.

Altered Address Markers...

This, together with the next few subroutines, forms the workhorse of the program. It displays

the current Address Markers and performs any changes to them that the user may desire. Since the next few routines perform the same basic actions to different areas of DOS, I will explain them all together. By maintaining a fairly standard pattern for each of the main routines, I have kept the program open to expansion.

First, each routine clears the screen and sets X and Y values. These values will be used by the input routine in order to place the cursor accurately. Then each routine prints a header across the top of the screen, identifying the current routine. Next the current status of the particular subject (sync bytes, etc.) is displayed, and the user is given the option to change the value(s). The program sets the variable AL to equal the desired input length and GOSUBS the input routine at 2100.

After changing the appropriate variables to equal the input, the routine sets variables AD, BD, and CD to the addresses each byte of input is to be POKEd into. If the string is only two bytes (4 characters) long the only AD and BD are used. The program the GOSUBs 2220, which POKEs the proper values into the correct addresses. Then the routine returns program control to the main menu.

Note that routines that required simply toggling a value such as the "Read Errors" routine skipped the input routine and POKE routines in favor of faster direct statements, such as a direct POKE or a simple GET.

Create EXECable File

This routine performs little work, yet is essential to the program. All it does is open a text file and print a list of hexadecimal addresses, each address followed by the appropriate values. When it is later EXECed, the file simply enters the monitor and enters a plethora of monitor commands of the form:

XXYY:ZZ

where XXYY is the address and ZZ is the value. The routine performs no calculations.

Error Handling

The error handling routine is an ordinary error routine. It PEEKs address 222 to find the error code, compares the value returned against a abbreviated list and prints the appropriate message. It finds the line number the error occurred in by the equation listed in line 1750.

GET a String

This routine HTABs and VTABs the cursor to the locations specified by the variables X and Y. Then it GETS a character of input, which it assigns to the variable A\$. It compiles the complete string of input by adding A\$ to A1\$. Then it returns and gets the next character of input. If the ESCape key is pressed during the input, the routine immediately returns to where it was called from. The length of input is limited to the number of characters specified by the variable AL. Once it has a complete string of length AL characters, it returns to where it was called from.

Hex to Decimal Conversion and POKE

This routine converts whatever the variable SR\$ currently contains to its decimal equivalent and enters it into memory. To convert it uses the same method used in most Applesoft programs, with a slight alteration: since the variable SR\$ might be longer than a single hexadecimal two digit number, the routine must first chop it into smaller, two digit numbers.

It does this by using the MID\$ command to selectively extract two digit packets from the string. Then it further divides each packet into single digits. It finds the decimal value of the sixteens digit, multiplies it by ten and adds it to the value of the ones digit, the result is the decimal value for the two digit packet. To find the values for each of the digits, the routine uses a FOR-NEXT loop. It compares the digit to a single character in a separate string which consists of the consecutive hexadecimal digits (\$0-F); when it finds a match, it knows that the number of loops already executed equals the decimal value of the digit.

The routine POKEs each decimal number to the address(es) stored in AD, BD, and CD.

PEEK a # and Convert to Hex

This routine is used by the initialization routine and is quite similar to the routine described directly above. It PEEKs a number from a specified location, divides the number to get a two values each between zero and fifteen. These numbers represent the hexadecimal equivalents of the decimal number that resulted from the PEEK. It then takes the xth character in a string of the consecutive hexadecimal characters to convert the numbers to a single digit between zero and SF. By adding the two numbers together in a string it arrives at the final, two digit hexadecimal number, which it assigns to the appropriate variable.

To Conclude

I have found the DOS Alterer to be extremely versatile and useful. I use it in tandem with almost all of my utilities that do not have built in provisions for protected disks, as well as to CATALOG and otherwise inspect protected disks. I hope it finds the same use in your collection of utilities.

DOS Alterer

```

100 REM DOS ALTERER
110 REM
120 REM BY
130 REM J. J. GIFFORD
140 REM
150 REM COPYRIGHT 1986
160 REM SOFTKEY PUBLISHING
170 REM
180 REM GOSUB INITIALIZATION
190 GOSUB 2450
200 REM MENU
210 TEXT : HOME : NORMAL
220 INVERSE : PRINT "THE^ DOS^ ALTERER" : NORMAL
230 POKE 34 , 1 : HOME
240 VTAB 23 : PRINT "ADDRESS^ HEADER:" ; ;
INVERSE : PRINT SA$ ; ; NORMAL : HTAB 23 :

```

```

PRINT "ADDRESS^ END:" ; ; INVERSE : PRINT
EAS
250 NORMAL : VTAB 24 : PRINT "DATA^ HEADER:" ; ;
INVERSE : PRINT SD$ ; ; NORMAL : HTAB 23 :
PRINT "DATA^ END:" ; ; INVERSE : PRINT ED$ ;
; NORMAL
260 POKE 35 , 22 : HOME
270 HTAB 6 : VTAB 6 : PRINT "OPTIONS:"
280 HTAB 9 : VTAB 8 : PRINT "1)^ ALTERED^
ADDRESS^ MARKERS"
290 HTAB 9 : VTAB 9 : PRINT "2)^ ALTERED^ DATA^
MARKERS"
300 HTAB 9 : VTAB 10 : PRINT "3)^ IGNORE^ READ^
ERRORS"
310 HTAB 9 : VTAB 11 : PRINT "4)^ ALTER^ SYNC^
BYTES"
320 HTAB 9 : VTAB 12 : PRINT "5)^ ALTER^ SLOT , ^
DRIVE"
330 HTAB 9 : VTAB 13 : PRINT "6)^ IGNORE^
ADDRESS^ CHECKSUM"
340 HTAB 9 : VTAB 14 : PRINT "7)^ CATALOG^ DISK"
350 HTAB 9 : VTAB 15 : PRINT "8)^ CREATE^ TEXT^
FILE"
360 HTAB 9 : VTAB 16 : PRINT "9)^ NORMALIZE^ DOS"
370 HTAB 6 : VTAB 18 : PRINT "OPTION^ #^ =>" ; ;
GET OP$ :
380 OP = VAL (OP$ )
390 IF OP$ = CHR$ (27 ) THEN 1760
400 ON OP GOSUB 430 , 620 , 790 , 890 , 1060 , 1200
, 1580 , 1330 , 2940
410 GOTO 230
420 END
430 REM ALTERED ADDRESS MARKERS
440 HOME : Y = 10 : X = 19
450 HTAB 9 : VTAB 4 : INVERSE : PRINT "ALTER^
ADDRESS^ MARKERS" : NORMAL : POKE 34 , 4
460 HTAB 6 : VTAB 7 : PRINT "START^ OF^ ADDRESS:"
470 HTAB 8 : VTAB 9 : PRINT "CURRENT^ VALUE^ =>"
; SA$
480 HTAB 8 : VTAB 10 : PRINT "ALTER^ TO^ =>" ;
490 AL = 6 : GOSUB 2100 : IF A$ = CHR$ (27 ) THEN 520
500 SA$ = A1$
510 SR$ = SA$ : AD = 47445 : BD = 47455 : CD = 47466
: GOSUB 2220
520 HOME : AL = 4
530 HTAB 6 : VTAB 7 : PRINT "END^ OF^ ADDRESS:"
540 HTAB 8 : VTAB 9 : PRINT "CURRENT^ VALUE^ =>"
; EA$
550 HTAB 8 : VTAB 10 : PRINT "ALTER^ TO^ =>" ;
560 GOSUB 2100 : IF A$ = CHR$ (27 ) THEN 590
570 EA$ = A1$
580 SR$ = EA$ : AD = 47505 : BD = 47515 : GOSUB 2220
590 REM
600 RETURN
610 REM
620 REM ALTERED DATA MARKERS
630 HTAB 9 : VTAB 4 : INVERSE : PRINT "ALTER^
DATA^ MARKERS" : NORMAL : POKE 34 , 4
640 Y = 10 : X = 19 : HOME : AL = 6
650 HTAB 6 : VTAB 7 : PRINT "START^ OF^ DATA:"
660 HTAB 8 : VTAB 9 : PRINT "CURRENT^ VALUE^ =>"
; SD$
670 HTAB 8 : VTAB 10 : PRINT "ALTER^ TO^ =>" ;
680 GOSUB 2100 : IF A$ = CHR$ (27 ) THEN 710
690 SD$ = A1$
700 SR$ = SD$ : AD = 47335 : BD = 47345 : CD = 47356
: GOSUB 2220
710 HOME : AL = 4
720 HTAB 6 : VTAB 7 : PRINT "END^ OF^ DATA:"
730 HTAB 8 : VTAB 9 : PRINT "CURRENT^ VALUE^ =>"
; ED$
740 HTAB 8 : VTAB 10 : PRINT "ALTER^ TO^ =>" ;
750 GOSUB 2100 : IF A$ = CHR$ (27 ) THEN 780
760 ED$ = A1$

```

```

770 SR$ = ED$ : AD = 47413 : BD = 47423 : GOSUB 2220
780 RETURN
790 REM IGNORE READ ERRORS
800 HOME
810 INVERSE : VTAB 4 : HTAB 6 : PRINT "READ^
ERRORS:" : NORMAL
820 VTAB 8 : HTAB 8 : PRINT "CURRENTLY^ =>" : RS$
830 VTAB 9 : HTAB 8 : INVERSE : PRINT "S" : ;
NORMAL : PRINT "ET^ / ^" : ; INVERSE : PRINT
"I" : ; NORMAL : PRINT "GNORE^ =>" :
840 GET AS : IF AS = CHR$ (27) THEN RETURN
850 IF AS = "I" THEN RS$ = "IGNORE" : POKE 47426
,24 : RETURN
860 IF AS = "S" THEN RS$ = "SET" : POKE 47426 ,56
: RETURN
870 GOTO 840
880 RETURN
890 REM ALTER ADDRESS SYNC BYTE
900 HOME : AL = 2 : X = 19 : Y = 10
910 HTAB 9 : VTAB 4 : INVERSE : PRINT "ALTER^
ADDRESS^ SYNC^ BYTE" : NORMAL : POKE 34 ,4
920 HTAB 6 : VTAB 7 : PRINT "ADDRESS^ SYNC:"
930 HTAB 8 : VTAB 9 : PRINT "CURRENT^ VALUE^ =>"
: AS$
940 HTAB 8 : VTAB 10 : PRINT "ALTER^ TO^ =>" :
950 GOSUB 2100 : IF AS = CHR$ (27) THEN 980
960 AS$ = A1$
970 SR$ = AS$ : AD = 48224 : GOSUB 2220
980 HOME : AL = 2 : X = 19 : Y = 10
990 HTAB 6 : VTAB 7 : PRINT "DATA^ SYNC:"
1000 HTAB 8 : VTAB 9 : PRINT "CURRENT^ VALUE^ =>"
: DS$
1010 HTAB 8 : VTAB 10 : PRINT "ALTER^ TO^ =>" :
1020 GOSUB 2100 : IF AS = CHR$ (27) THEN 1050
1030 DS$ = A1$
1040 SR$ = DS$ : AD = 47166 : GOSUB 2220
1050 RETURN
1060 REM CHANGE SLOT ,DRIVE
1070 HOME
1080 HTAB 9 : VTAB 4 : INVERSE : PRINT "CHANGE^
SLOT ,DRIVE" : NORMAL : POKE 34 ,4
1090 HTAB 6 : VTAB 7 : PRINT "DRIVE^ =>" : DR
1100 HTAB 14 : VTAB 7 : GET AS : PRINT AS$ : IF AS
= CHR$ (27) THEN RETURN
1110 IF AS = CHR$ (13) THEN 1140
1120 IF AS <> "I" AND AS <> "2" THEN 1090
1130 DR = VAL (AS)
1140 HTAB 6 : VTAB 9 : PRINT "SLOT^ =>" : SL
1150 HTAB 13 : VTAB 9 : GET AS : PRINT AS$ : IF AS
= CHR$ (27) THEN RETURN
1160 IF AS = CHR$ (13) THEN RETURN
1170 IF VAL (AS) > 6 OR VAL (AS) < 1 THEN 1150
1180 SL = VAL (AS) : RETURN
1190 RETURN
1200 REM IGNORE ADDRESS CHECKSUM
1210 HOME
1220 HTAB 12 : VTAB 4 : INVERSE : PRINT "KILL^
ADDRESS^ CHECKSUM" : NORMAL : POKE 34 ,4
1230 HTAB 6 : VTAB 7 : PRINT "CURRENTLY^ =>" : CH$
1240 HTAB 6 : VTAB 8 : INVERSE : PRINT "S" : ;
NORMAL : PRINT "ET^ / ^" : ; INVERSE : PRINT
"I" : ; NORMAL : PRINT "GNORE^ =>" :
1250 GET AS : PRINT AS$ : IF AS = CHR$ (27) THEN
RETURN
1260 IF AS <> "S" AND AS <> "I" THEN 1240
1270 IF AS = "S" THEN CH$ = "SET" : GOTO 1310
1280 IF AS = "I" THEN CH$ = "IGNORE"
1290 POKE 47498 ,0
1300 RETURN
1310 POKE 47498 ,183
1320 RETURN
1330 REM CREATE EXECABLE FILE
1340 HOME : AS = "" : NS = ""

```

```

1350 VTAB 4 : HTAB 9 : INVERSE : PRINT "CREATE^
TEXT^ FILE" : NORMAL : POKE 34 ,4
1360 GOSUB 1830 : REM GET FILE NAME
1370 IF AS = CHR$ (27) THEN RETURN
1380 TX = 1 : GOSUB 2800 : REM NORMALIZER
1390 PRINT DS "OPEN^ " ; NS ; "D" ; DR ; "S" ; SL ; "V"
: VO : PRINT DS "CLOSE^ " ; NS
1400 PRINT DS "DELETE^ " ; NS
1410 PRINT DS "OPEN^ " ; NS
1420 PRINT DS "WRITE^ " ; NS
1430 PRINT "CALL^ -151"
1440 PRINT "B955:" : LEFT$ (SA$ ,2) : PRINT
"B95F:" : MIDS (SA$ ,3 ,2) : PRINT "B96A:"
: RIGHT$ (SA$ ,2)
1450 PRINT "BC7A:" : LEFT$ (SA$ ,2) : PRINT
"BC7F:" : MIDS (SA$ ,3 ,2) : PRINT "BC84:"
: RIGHT$ (SA$ ,2)
1460 PRINT "B991:" : LEFT$ (EA$ ,2) : PRINT
"B99B:" : RIGHT$ (EA$ ,2)
1470 PRINT "BCAE:" : LEFT$ (EA$ ,2) : PRINT
"BCB3:" : RIGHT$ (EA$ ,2)
1480 PRINT "B8E7:" : LEFT$ (SD$ ,2) : PRINT
"B8F1:" : MIDS (SD$ ,3 ,2) : PRINT "B8FC:"
: RIGHT$ (SD$ ,2)
1490 PRINT "B853:" : LEFT$ (SD$ ,2) : PRINT
"B858:" : MIDS (SD$ ,3 ,2) : PRINT "B85D:"
: RIGHT$ (SD$ ,2)
1500 PRINT "B935:" : LEFT$ (ED$ ,2) : PRINT
"B93F:" : RIGHT$ (ED$ ,2)
1510 PRINT "B89E:" : LEFT$ (ED$ ,2) : PRINT
"B8A3:" : RIGHT$ (ED$ ,2)
1520 PRINT "BC60:" : AS$ : PRINT "B83E:" : DS$
1530 IF RS$ = "IGNORE" THEN PRINT "B942:18"
1540 IF CH$ = "IGNORE" THEN PRINT "B98A:00"
1550 PRINT DS "CLOSE^ " ; NS
1560 GOSUB 2450
1570 RETURN
1580 REM CATALOG DISK
1590 HOME
1600 PRINT : PRINT DS "CATALOG,D" ; DR ; "S" ; SL ;
"V" : VO
1610 PRINT : INVERSE : VTAB 22 : PRINT "PRESS^
ANY^ KEY^ =>" : ; GET AS : NORMAL
1620 RETURN
1630 REM ERROR HANDLING ROUTINE
1640 ERR = PEEK (222)
1650 IF ERR > 15 AND ERR < 254 THEN POKE 216 ,0 : AS
= "ERROR" : GOTO 1740
1660 IF ERR = 254 THEN PRINT "TYPE^ AGAIN^
PLEASE:" : PRINT : RESUME
1670 IF ERR = 255 THEN STOP
1680 IF ERR = 2 THEN AS = "VOLUME^ MISMATCH^
ERROR"
1690 RETURN
1700 IF ERR = 6 THEN AS = "FILE^ NOT^ FOUND^
ERROR"
1710 IF ERR = 8 THEN AS = "I/O^ ERROR"
1720 IF ERR = 11 THEN AS = "SYNTAX^ ERROR"
1730 IF ERR = 12 THEN AS = "ERROR"
1740 HOME
1750 PRINT AS "IN^ LINE^ #" : PEEK (219) * 256
+ PEEK (218) : PRINT CHR$ (7) : VTAB 20 :
HTAB 4 : PRINT "PRESS^ ANY^ KEY^ =>" : ; GET
AS : GOTO 230
1760 REM ESCAPE AT MENU
1770 HOME
1780 HTAB 8 : VTAB 6 : PRINT "EXIT^ PROGRAM^
(Y/N)?" :
1790 GET AS :
1800 IF AS = "Y" THEN TEXT : HOME : END
1810 IF AS = "N" THEN 230
1820 GOTO 1790
1830 REM GET FILE NAME ,DRIVE ,SLOT
1840 REM

```

```

1850 HTAB 6 : VTAB 7 : PRINT "FILENAME^ >" ; NS ;
1860 GET AS : IF AS = CHR$ (27) THEN RETURN
1870 IF AS = CHR$ (8) AND LEN (NS) <= 1 THEN NS
= "" : GOTO 1850
1880 IF AS = CHR$ (8) THEN NS = LEFT$ (NS , LEN
(NS) - 1) : GOTO 1850
1890 IF LEN (NS) > 25 THEN PRINT CHR$ (7) : GOTO
1850
1900 IF AS = CHR$ (13) AND LEN (NS) >= 1 THEN 1950
1910 IF AS = CHR$ (8) THEN NS = LEFT$ (NS , LEN
(NS) - 1) : GOTO 1850
1920 IF AS = CHR$ (8) THEN NS = LEFT$ (NS , LEN
(NS) - 1) : GOTO 1850
1930 IF AS = CHR$ (13) AND NS = "" THEN 1850
1940 NS = NS + AS : GOTO 1850
1950 HTAB 6 : VTAB 9 : PRINT "DRIVE^ #^ =>" ; DR
1960 HTAB 16 : VTAB 9 : GET AS
1970 IF AS = CHR$ (13) THEN 2000
1980 IF AS <> "2" AND AS <> "I" THEN 1960
1990 DR = VAL (AS)
2000 HTAB 6 : VTAB 11 : PRINT "SLOT^ #^ =>" ; SL
2010 HTAB 15 : VTAB 11 : GET AS
2020 IF AS = CHR$ (13) THEN 2050
2030 IF VAL (AS) > 6 OR VAL (AS) < 1 THEN 2010
2040 SL = VAL (AS)
2050 HTAB 6 : VTAB 20 : INVERSE : PRINT "PRESS^
<RETURN>^ OR^ <ESCAPE>^ =>" : NORMAL :
GET AS : PRINT AS :
2060 IF AS = CHR$ (27) THEN RETURN
2070 IF AS = CHR$ (13) THEN 2090
2080 GOTO 2050
2090 RETURN
2100 REM GET A STRING
2110 A1$ = ""
2120 HTAB X : VTAB Y : PRINT A1$ : GET AS
2130 IF AS = CHR$ (27) THEN RETURN
2140 IF AS = CHR$ (8) AND LEN (A1$) = 1 THEN A1$
= "" : GOTO 2120
2150 IF AS = CHR$ (8) AND LEN (A1$) = 0 THEN PRINT
CHR$ (7) : GOTO 2120
2160 IF AS = CHR$ (8) THEN A1$ = LEFT$ (A1$ , (LEN
(A1$) - 1)) : GOTO 2120
2170 IF LEN (A1$) = AL AND AS = CHR$ (13) THEN
RETURN
2180 IF LEN (A1$) = AL AND AS <> CHR$ (8) THEN
PRINT CHR$ (7) : GOTO 2120
2190 IF (ASC (AS) < 48 OR ASC (AS) > 58) AND (
ASC (AS) < 65 OR ASC (AS) > 70) THEN PRINT
CHR$ (7) : GOTO 2120
2200 IF LEN (A1$) < AL THEN A1$ = A1$ + AS : GOTO
2120
2210 RETURN
2220 REM HEX => DEC CONV. & POKE
2230 REM SR$ IS STRING TO BE POKED
2240 A = 0 : B = 0 : C = 0
2250 L = LEN (SR$) : AS = LEFT$ (SR$ ,2)
2260 IF L > 2 THEN BS = MIDS (SR$ ,3 ,2)
2270 IF L = 6 THEN CS = MIDS (SR$ ,5 ,2)
2280 A1$ = LEFT$ (AS ,1) : A2$ = RIGHT$ (AS ,1)
2290 B1$ = LEFT$ (BS ,1) : B2$ = RIGHT$ (BS ,1)
2300 C1$ = LEFT$ (CS ,1) : C2$ = RIGHT$ (CS ,1)
2310 FOR I = 1 TO 16 : IF A1$ = MIDS (HX$ ,I ,1)
THEN A = A + ((I - 1) * 16)
2320 IF A2$ = MIDS (HX$ ,I ,1) THEN A = A + (I - 1)
2330 IF LEN (SR$) > 2 AND B1$ = MIDS (HX$ ,I ,1)
THEN B = B + ((I - 1) * 16)
2340 IF LEN (SR$) > 2 AND B2$ = MIDS (HX$ ,I ,1)
THEN B = B + (I - 1)
2350 IF LEN (SR$) > 4 AND C1$ = MIDS (HX$ ,I ,1)
THEN C = C + ((I - 1) * 16)
2360 IF LEN (SR$) > 4 AND C2$ = MIDS (HX$ ,I ,1)
THEN C = C + (I - 1)
2370 NEXT I
2380 IF TX = 1 THEN RETURN

```

```

2390 REM POKE A , B , C
2400 POKE AD , A
2410 IF LEN (SR$) > 2 THEN POKE BD , B
2420 IF LEN (SR$) > 4 THEN POKE CD , C
2430 RETURN
2440 END
2450 REM INITIALIZATION
2460 ONERR GOTO 1630
2470 DR = 1 : SL = 6 : VO = 0
2480 TK = 0 : ST = 0
2490 PS = 1 : RD = 0 : WR = 0 : DF = RD
2500 HX$ = "0123456789ABCDEF"
2510 W = PEEK (47445) : Q = PEEK (47455) : R = PEEK
(47466) : GOSUB 2680 : SA$ = WS + QS + RS
2520 W = PEEK (47505) : Q = PEEK (47515) : GOSUB
2680 : EA$ = WS + QS
2530 W = PEEK (47335) : Q = PEEK (47345) : R = PEEK
(47356) : GOSUB 2680 : SD$ = WS + QS + RS
2540 W = PEEK (47413) : Q = PEEK (47423) : GOSUB
2680 : ED$ = WS + QS
2550 W = PEEK (48224) : GOSUB 2680 : AS$ = WS : W
= PEEK (47166) : GOSUB 2680 : DS$ = WS
2560 X = 1 : Y = 1 : SR$ = "" : A1$ = "" : AS = ""
2570 A2$ = "" : B1$ = "" : B2$ = "" : C1$ = "" : C2$ = ""
2580 A = 0 : B = 0 : C = 0 : AD = 0 : BD = 0 : CD = 0 : REM
AD-CD ARE ADDRESS LOCATIONS TO POKE
AT .
2590 DS = CHR$ ( 4 )
2600 IF PEEK (47426) = 24 THEN RS$ = "IGNORE"
2610 IF PEEK (47426) = 56 THEN RS$ = "SET"
2620 TX = 0 : REM TEXT
2630 IF PEEK (47498) = 0 THEN CH$ = "IGNORE" :
GOTO 2660
2640 IF PEEK (47498) = 183 THEN CH$ = "SET" : GOTO
2660
2650 CH$ = "?"
2660 REM
2670 RETURN
2680 REM PEEK & CONV A # TO HEX
2690 REM
2700 W1 = INT (W / 16) + 1 : W2 = (((W / 16) - INT
(W / 16)) * 16) + 1
2710 IF Q = 0 THEN 2750
2720 Q1 = INT (Q / 16) + 1 : Q2 = (((Q / 16) - INT
(Q / 16)) * 16) + 1
2730 IF R = 0 THEN 2750
2740 R1 = INT (R / 16) + 1 : R2 = (((R / 16) - INT
(R / 16)) * 16) + 1
2750 WS = MID$ (HX$ , W1 , 1) + MID$ (HX$ , W2 , 1)
2760 QS = MID$ (HX$ , Q1 , 1) + MID$ (HX$ , Q2 , 1)
2770 RS = MID$ (HX$ , R1 , 1) + MID$ (HX$ , R2 , 1)
2780 RETURN
2790 RETURN
2800 REM NORMALIZER SUBROUTINE
2810 REM
2820 POKE 47445 , 213 : POKE 47455 , 170 : POKE
47466 , 150
2830 POKE 47505 , 222 : POKE 47515 , 170
2840 POKE 47335 , 213 : POKE 47345 , 170 : POKE
47356 , 173
2850 POKE 47413 , 222 : POKE 47423 , 170
2860 POKE 47498 , 183
2870 POKE 48250 , 213 : POKE 48255 , 170 : POKE
48260 , 150
2880 POKE 47262 , 222 : POKE 47267 , 170
2890 POKE 47166 , 255 : POKE 48224 , 255
2900 POKE 47187 , 213 : POKE 47192 , 170 : POKE
47197 , 173
2910 POKE 48302 , 222 : POKE 48307 , 170
2920 POKE 47426 , 56
2930 RETURN
2940 REM NORMALIZE DOS
2950 GOSUB 2800 : REM NORMALIZER

```

```

2960 GOSUB 2450 : REM INITIALIZATION
2970 RETURN
2980 END

```

controller checksums

100	- \$0236	1550	- \$1238	820	- \$F406	2270	- \$32D0
110	- \$F454	1560	- \$8647	830	- \$50EA	2280	- \$4B5F
120	- \$F699	1570	- \$5D1B	840	- \$61D1	2290	- \$64B3
130	- \$F780	1580	- \$928E	850	- \$80D4	2300	- \$4813
140	- \$BAAC	1590	- \$66FC	860	- \$244E	2310	- \$0140
150	- \$26B4	1600	- \$08AB	870	- \$37A8	2320	- \$0594
160	- \$661F	1610	- \$653F	880	- \$C638	2330	- \$1BD3
170	- \$4918	1620	- \$0553	890	- \$B5BD	2340	- \$1383
180	- \$52B6	1630	- \$F2FB	900	- \$95C4	2350	- \$2B84
190	- \$1B1E	1640	- \$45B9	910	- \$8BB4	2360	- \$32ED
200	- \$2C66	1650	- \$3275	920	- \$B578	2370	- \$2E71
210	- \$0BBC	1660	- \$2AA7	930	- \$CBE1	2380	- \$A023
220	- \$39F3	1670	- \$21FF	940	- \$446D	2390	- \$5331
230	- \$0CBA	1680	- \$281B	950	- \$1E24	2400	- \$1D8B
240	- \$5006	1690	- \$E199	960	- \$F411	2410	- \$DA79
250	- \$18C3	1700	- \$E236	970	- \$3C6E	2420	- \$3147
260	- \$07F6	1710	- \$9212	980	- \$1655	2430	- \$8CE9
270	- \$6562	1720	- \$C780	990	- \$94B1	2440	- \$9B73
280	- \$B37B	1730	- \$C567	1000	- \$177F	2450	- \$8BAC
290	- \$2C34	1740	- \$7BAC	1010	- \$4713	2460	- \$2CBB
300	- \$0E20	1750	- \$BAE5	1020	- \$AEDD	2470	- \$4CA3
310	- \$B3FE	1760	- \$9CF7	1030	- \$39E3	2480	- \$57C6
320	- \$C067	1770	- \$FBEE	1040	- \$5D07	2490	- \$28DB
330	- \$B478	1780	- \$91B4	1050	- \$B0E2	2500	- \$895C
340	- \$3C20	1790	- \$3664	1060	- \$C233	2510	- \$16FD
350	- \$A41D	1800	- \$807E	1070	- \$CAA8	2520	- \$8604
360	- \$EF64	1810	- \$6053	1080	- \$C1A2	2530	- \$1333
370	- \$916E	1820	- \$1E73	1090	- \$522D	2540	- \$852A
380	- \$13E1	1830	- \$6801	1100	- \$F04B	2550	- \$5741
390	- \$2ABD	1840	- \$3A6C	1110	- \$194D	2560	- \$BF77
400	- \$CE17	1850	- \$7A1F	1120	- \$3462	2570	- \$64A4
410	- \$76AE	1860	- \$CEFD	1130	- \$7C4B	2580	- \$3581
420	- \$B4A9	1870	- \$68D2	1140	- \$9967	2590	- \$6ECD
430	- \$A5E5	1880	- \$34F1	1150	- \$B61C	2600	- \$1B92
440	- \$9215	1890	- \$D9F1	1160	- \$924B	2610	- \$17D3
450	- \$5EF2	1900	- \$A310	1170	- \$2E90	2620	- \$D825
460	- \$915F	1910	- \$4E2D	1180	- \$FCDB	2630	- \$9D32
470	- \$888C	1920	- \$5FBB	1190	- \$74F9	2640	- \$A599
480	- \$3141	1930	- \$D332	1200	- \$C5B3	2650	- \$9E02
490	- \$3537	1940	- \$BBC8	1210	- \$22D1	2660	- \$596B
500	- \$5D9B	1950	- \$BFDB	1220	- \$A059	2670	- \$1752
510	- \$1FD3	1960	- \$A49F	1230	- \$19B1	2680	- \$E1F0
520	- \$0CD9	1970	- \$2C91	1240	- \$F015	2690	- \$9313
530	- \$5A02	1980	- \$372F	1250	- \$187B	2700	- \$7554
540	- \$8EB8	1990	- \$644D	1260	- \$A78F	2710	- \$8C41
550	- \$95C6	2000	- \$6863	1270	- \$D974	2720	- \$351E
560	- \$F79C	2010	- \$1044	1280	- \$75DE	2730	- \$6F82
570	- \$3D04	2020	- \$BB4F	1290	- \$423B	2740	- \$05B4
580	- \$669E	2030	- \$8B83	1300	- \$E243	2750	- \$62D5
590	- \$59A2	2040	- \$D8E9	1310	- \$9B2D	2760	- \$199A
600	- \$36BF	2050	- \$A1D3	1320	- \$58B4	2770	- \$CC06
610	- \$CCBE	2060	- \$C80F	1330	- \$D6F1	2780	- \$6EF4
620	- \$21B5	2070	- \$050D	1340	- \$1B59	2790	- \$6CD3
630	- \$16FC	2080	- \$2BF3	1350	- \$1549	2800	- \$9C31
640	- \$EB7F	2090	- \$6A70	1360	- \$9D7B	2810	- \$5ED4
650	- \$9D6F	2100	- \$9681	1370	- \$F821	2820	- \$3013
660	- \$B3C3	2110	- \$B35B	1380	- \$4F99	2830	- \$673B
670	- \$244D	2120	- \$AFE4	1390	- \$5495	2840	- \$A97A
680	- \$A610	2130	- \$5762	1400	- \$959E	2850	- \$1A5D
690	- \$FB2B	2140	- \$FD30	1410	- \$DDDE	2860	- \$CAA8
700	- \$08DB	2150	- \$1ED3	1420	- \$E520	2870	- \$9C6E
710	- \$7C51	2160	- \$0174	1430	- \$1A95	2880	- \$2028
720	- \$8A16	2170	- \$8745	1440	- \$6724	2890	- \$A88B
730	- \$9226	2180	- \$3CCB	1450	- \$971A	2900	- \$EFA5
740	- \$78B8	2190	- \$6382	1460	- \$54D8	2910	- \$37EC
750	- \$DFC2	2200	- \$7022	1470	- \$6750	2920	- \$7C68
760	- \$31CC	2210	- \$07B1	1480	- \$B106	2930	- \$4D02
770	- \$F806	2220	- \$5BA6	1490	- \$1C91	2940	- \$0BFA
780	- \$5851	2230	- \$989D	1500	- \$DF91	2950	- \$EDF4
790	- \$3867	2240	- \$7944	1510	- \$31BE	2960	- \$B732
800	- \$2B0E	2250	- \$9E87	1520	- \$FE12	2970	- \$8825
810	- \$E9C5	2260	- \$1E5C	1530	- \$3C2C	2980	- \$F2B3
				1540	- \$34FF		

Flight Simulator II v1.05

Source Code files

Source Code for

COPY

```

*          FLIGHT SIMULATOR COPY          *
          .OR $1000
          .TA $800
          .TF OBJ.COPY
FDED- COUT      .EQ $FDED
F948- PRBLNK   .EQ $F948
FB2F- SETTXX  .EQ $FB2F
FB40- SETGR    .EQ $FB40
FC58- HOME     .EQ $FC58
FD0C- KEYIN    .EQ $FD0C
FD0A- PRBYTE   .EQ $FD0A
B7E9- SLOT     .EQ $B7E9
B7EA- DRIVE    .EQ $B7EA
B7EB- VOL      .EQ $B7EB
B7EC- TRACK    .EQ $B7EC
B7ED- SCTR     .EQ $B7ED
B7F0- BUF      .EQ $B7F0
B7F4- CMND     .EQ $B7F4
B7F8- OLDDR   .EQ $B7F8
C08A- DRIVE1   .EQ $C08A
C088- DRVOFF  .EQ $C088
C089- DRVON    .EQ $C089
*
1000: 20 2F FB      JSR SETTXX
1003: 20 40 FB      JSR SETGR
1006: 20 58 FC      JSR HOME
1009: 2C 57 C0      BIT $C057
*
* GET .NUMBER. OF .DRIVES
*
100C: A9 BF          LDA #$BF
100E: 20 ED FD      JSR COUT
1011: 20 0C FD KEY1 JSR KEYIN
1014: C9 B3          CMP #$B3
1016: B0 F9          BCS KEY1
1018: C9 B1          CMP #$B1
101A: 90 F5          BCC KEY1
101C: 29 03          AND #$03
101E: 8D EA B7      STA DRIVE
1021: 4A             LSR
1022: 85 FF          STA $FF
1024: AE E9 B7      LDX SLOT
1027: 8E 08 1E      STX $1E08
*
* RECALIBRATE .DRIVE1
*
102A: 9D 8A C0      STA DRIVE1,X
102D: 9D 89 C0      STA DRVON,X
1030: A9 60          LDA #$60
1032: 20 95 BE      JSR $BE95
1035: A9 00          LDA #$00
1037: 20 5A BE      JSR $BE5A
*

```

* SETUP. FOR .COPY

```

*
103A: 20 F9 1E      JSR $1EF9
103D: 20 C0 10      JSR OFF
1040: A0 02          LDY #$02
1042: 8C F4 B7      STY CMND
1045: A0 00          LDY #$00
1047: 8C F0 B7      STY BUF
104A: 8C 0A 1E      STY $1E0A
*

```

* READ .SOURCE

```

*
104D: 8C 01 1E LOOP1 STY $1E01
1050: 20 58 FC      JSR HOME
1053: A5 FF          LDA $FF
1055: D0 05          BNE TWO1
1057: A9 D3          LDA #$D3
1059: 20 B7 10      JSR KEY2
105C: BD 8A C0 TWO1 LDA DRIVE1,X
105F: 20 AC 1F      JSR $1FAC
1062: 20 74 20      JSR $2074
1065: 20 C0 10      JSR OFF
1068: AD 0B 1E      LDA $1E0B
106B: 4A             LSR
106C: 8D EC B7      STA TRACK
106F: 20 DA FD      JSR PRBYTE
*

```

* WRITE .TO .TARGET

```

*
1072: A5 FF          LDA $FF
1074: D0 0B          BNE TWO2
1076: 20 48 F9      JSR PRBLNK
1079: A9 D4          LDA #$D4
107B: 20 B7 10      JSR KEY2
107E: EE 78 04      INC $478
1081: A9 35 TWO2    LDA #$35
1083: A2 01          LDX #$01
1085: A0 0F          LDY #$0F
1087: 8D F1 B7      STA BUF+1
108A: 8E F8 B7      STX OLDDR
108D: 8C ED B7 LOOP2 STY SCTR
1090: 20 AB 10      JSR WRITE
1093: CE F1 B7      DEC BUF+1
1096: AC ED B7      LDY SCTR
1099: 88             DEY
109A: 10 F1          BPL LOOP2
*

```

* ANOTHER .TRACK?

```

*
109C: AC 01 1E      LDY $1E01
109F: C8             INY
10A0: C8             INY
10A1: C8             INY
10A2: C8             INY
10A3: C0 88          CPY #$88
10A5: 90 A6          BCC LOOP1
10A7: 20 2F FB      JSR SETTXX
10AA: 60             RTS
*
10AB: A0 00 WRITE   LDY #$00
10AD: 8C EB B7      STY VOL

```

```

10B0: A0 E8          LDY #$E8
10B2: A9 B7          LDA #$B7
10B4: 4C B5 B7      JMP $B7B5
10B7: 20 ED FD KEY2 JSR COUT
10BA: 20 0C FD      JSR KEYIN
10BD: 4C 58 FC      JMP HOME
10C0: AE E9 B7 OFF   LDX SLOT
10C3: 9D 88 C0      STA DRVOFF,X
10C6: 60             RTS

```

DOS LOADER

```

*          FLIGHT SIM DOS LOADER          *
          .OR $1F06
          .TF OBJ.DOS LOADER
*
* SAVE .PAGES .AFFECTED .BY .BOOT
*
1F06: A0 00          LDY #$00
1F08: B9 00 00 MOVE1 LDA $00,Y
1F0B: 99 00 36      STA $3600,Y
1F0E: B9 00 01      LDA $0100,Y
1F11: 99 00 37      STA $3700,Y
1F14: B9 00 03      LDA $0300,Y
1F17: 99 00 38      STA $3800,Y
1F1A: B9 00 08      LDA $0800,Y
1F1D: 99 00 39      STA $3900,Y
1F20: 88             DEY
1F21: D0 E5          BNE MOVE1
*
* TELL .BOOT .TO .LOAD .2000 .25FF
*
1F23: A9 49          LDA #$49
1F25: 85 00          STA $00
1F27: A9 23          LDA #$23
1F29: 85 01          STA $01
*
* BOOT .DISK
*
1F2B: AD 08 1E      LDA $1E08
1F2E: 4A             LSR
1F2F: 4A             LSR
1F30: 4A             LSR
1F31: 4A             LSR
1F32: 09 C0          ORA #$C0
1F34: 8D 39 1F      STA CALL+2
1F37: 20 00 C6 CALL JSR $C600
*
* RETURN .MEM .TO .CORRECT .PLACE
*
1F3A: A0 00          LDY #$00
1F3C: B9 00 36 MOVE2 LDA $3600,Y
1F3F: 99 00 00      STA $00,Y
1F42: B9 00 37      LDA $3700,Y
1F45: 99 00 01      STA $0100,Y
1F48: B9 00 38      LDA $3800,Y
1F4B: 99 00 03      STA $0300,Y
1F4E: B9 00 39      LDA $3900,Y

```

```

1F51: 99 00 08      STA $0800,Y
1F54: 88            DEY
1F55: D0 E5        BNE MOVE2
*
* SATISFY TRK. SEEK, UPON RETURN
*
1F57: A9 00        LDA #$00
1F59: 60            RTS

```

```

241F: 10 FB
2421: 2A
2422: 85 B7
2424: BD 8C C0 READ5
2427: 10 FB
2429: 25 B7
242B: 88
242C: D0 EC
242E: 85 B7
2430: A5 B8
2432: C5 AC
2434: D0 0B
2436: A4 AB
2438: B9 A7 24
243B: C5 B7
243D: D0 BE
243F: 18
2440: 60
*
* ERROR HANDLER
*
2441: 38          ERROR SEC
2442: 60          RTS
*
* READ DATA
*
2443: BD 8C C0 RDDATA LDA $C08C,X
2446: 10 FB      BPL RDDATA
2448: C9 D5      CHK2   CMP #$D5
244A: D0 F7      BNE RDDATA
244C: BD 8C C0 READ7 LDA $C08C,X
244F: 10 FB      BPL READ7
2451: C9 AA      CMP #$AA
2453: D0 F3      BNE CHK2
2455: BD 8C C0 READ8 LDA $C08C,X
2458: 10 FB      BPL READ8
245A: C9 AD      CMP #$AD
245C: D0 EA      BNE CHK2
245E: A9 00      LDA #$0
2460: A0 56      LDY #$56
2462: 84 B7      RDLP1  STY TEMP0
2464: BC 8C C0 READ9 LDY $C08C,X
2467: 10 FB      BPL READ9
2469: 59 00 3F  EOR CONVERT,Y
246C: A4 B7      LDY TEMP0
246E: 88          DEY
246F: 99 00 36  STA BUFF2,Y
2472: D0 EE      BNE RDLP1
2474: 84 B7      RDLP2  STY TEMP0
2476: BC 8C C0 READ10 LDY $C08C,X
2479: 10 FB      BPL READ10
247B: 59 00 3F  EOR CONVERT,Y
247E: A4 B7      LDY TEMP0
2480: 91 A9      STA (BUFF),Y
2482: C8          INY
2483: D0 EF      BNE RDLP2
2485: BC 8C C0 READ11 LDY $C08C,X
2488: 10 FB      BPL READ11
248A: D9 00 3F  CMP CONVERT,Y
248D: D0 B2      BNE ERROR
*
* POSTNIBBLE
*
248F: A0 00      LDY #$0
2491: A2 56      LDX #$56
2493: CA          DEX
2494: 30 FB      BMI NEWX
2496: B1 A9      LDA (BUFF),Y
2498: 5E 00 36  LSR BUFF2,X
2499: 2A          ROL
249C: 5E 00 36  LSR BUFF2,X
249F: 2A          ROL
24A0: 91 A9      STA (BUFF),Y
24A2: C8          INY

```

```

24A3: D0 EE      BNE PSTNB
24A5: 18          CLC
24A6: 60          RTS
24A7: 00 0D 0B
24AA: 09 07 05
24AD: 03 01      TABLE1 .DA #$00
          #$0D,$$0B,$$09,$$07,$$05,$$03,$$01
24AF: 0E 0C 0A
24B2: 08 06 04
24B5: 02 0F      .DA #$0E,
          #$0C,$$0A,$$08,$$06,$$04,$$02,$$0F

```

NEW DOS / LC DOS

```

*          FLIGHT SIM NEW DOS          *
* CHANGE ORIGIN TO $D7B5 FOR "LC DOS" *
          .OR $23D0
          .TF OBJ.NEW DOS
00A9- BUFF .EQ $A9
00AB- SECTOR .EQ $AB
00AC- TRACK .EQ $AC
00B7- TEMP0 .EQ $B7
00B8- TRACKFND .EQ $B8
3600- BUFF2 .EQ $3600
3F00- CONVERT .EQ $3F00
*
* SETUP READ
*
23D0: A9 00      LDA #$00
23D2: A2 35      LDX #$35
23D4: A0 0F      LDY #$0F
23D6: 85 A9      STA BUFF
23D8: 86 AA      STX BUFF+1
23DA: 84 AB      STY SECTOR
23DC: AD 0B 1E   LDA $1E0B
23DE: AE 08 1E   LDX $1E08
23E2: 4A          LSR
23E3: 85 AC      STA TRACK
23E5: BD 8E C0   LDA $C08E,X
*
* READ
*
23E8: 20 FD 23 RDLOOP JSR ADDRESS
23EB: B0 0F      BCS END
23ED: 20 43 24   JSR RDDATA
23F0: B0 F6      BCS RDLOOP
*
* TRK. SCTR. COUNTER
*
23F2: AE 08 1E   LDX $1E08
23F5: C6 AA      DEC BUFF+1
23F7: C6 AB      DEC SECTOR
23F9: 10 ED      BPL RDLOOP
23FB: 18          CLC
23FC: 60          END   RTS
*
* READ ADDRESS
*
23FD: BD 8C C0 ADDRESS LDA $C08C,X
2400: 10 FB      BPL ADDRESS
2402: C9 D5      CHK1   CMP #$D5
2404: D0 F7      BNE ADDRESS
2406: BD 8C C0 READ2 LDA $C08C,X
2409: 10 FB      BPL READ2
240B: C9 AA      CMP #$AA
240D: D0 F3      BNE CHK1
240F: BD 8C C0 READ3 LDA $C08C,X
2412: 10 FB      BPL READ3
2414: C9 96      CMP #$96
2416: D0 EA      BNE CHK1
2418: A0 03      LDY #$3
241A: 85 B8      STRTRK STA TRACKFND
241C: BD 8C C0 READ4 LDA $C08C,X

```

WRITE / LC WRITE

```

*          FLIGHT SIM WRITE          *
* CHANGE ORIGIN TO $D5C8 FOR LC WRITE *
* AND CHANGE "ADDRESS .EQ 23FD" TO $D7E2 *
          .OR $21E3
          .TF OBJ.WRITE
23FD- ADDRESS .EQ $23FD
1E55- WRTBL .EQ $1E55
3600- PRBUF .EQ $3600
3700- SEBUF .EQ $3700
00A8- BUFPTR .EQ $A8
00A9- TEMP .EQ $A9
00AA- ZSLOT .EQ $AA
00AB- SECTOR .EQ $AB
00AC- TRACK .EQ $AC
1E08- SLOT .EQ $1E08
1E0B- PHASE .EQ $1E0B
21E3: AD 0B 1E   LDA PHASE
21E6: A2 00      LDX #$0
21E8: A0 0F      LDY #$0F
21EA: 4A          LSR
21EB: 85 AC      STA TRACK
21ED: 86 A8      STX BUFPTR
21EF: 84 AB      STY SECTOR
21F1: 20 0F 22 LOOP1 JSR PRENIB
21F4: AE 08 1E   LDX SLOT
21F7: 20 FD 23   JSR ADDRESS
21FA: B0 12      BCS END
*
* DON'T WRITE OVER ADRS. FLD
*
21FC: A0 03      LDY #$3
21FE: BD 8C C0 READ LDA $C08C,X
2201: 10 FB      BPL READ
2203: 88          DEY
2204: 10 F8      BPL READ
*
* JUMP TO WRITE SUBROUTINE
*
2206: 20 40 22   JSR WRITE
2209: C6 AB      DEC SECTOR
220B: 10 E4      BPL LOOP1
220D: 18          CLC
220E: 60          END   RTS
*
* PRENIBBLE DATA
*
220F: A5 AB      PRENIB LDA SECTOR
2211: 18          CLC
2212: 69 26      ADC #$26
2214: 85 A9      STA BUFPTR+1
2216: A2 00      LDX #$0
2218: A0 02      LDY #$2
221A: 88          DEY
221B: B1 A8      PLOOP1 LDA (BUFPTR),Y
221D: 4A          LSR
221E: 3E 00 37  ROL SEBUF,X
2221: 4A          LSR

```

2222: 3E 00 37 ROL SEBUF, X
 2225: 99 00 36 STA PRBUF, Y
 2228: E8 INX
 2229: E0 56 CPX #56
 222B: 90 ED BCC PLOOP1
 222D: A2 00 LDX #50
 222F: 98 TYA
 2230: D0 E8 BNE PLOOP1
 2232: A2 55 LDX #55
 2234: BD 00 37 PLOOP2 LDA SEBUF, X
 2237: 29 3F AND #3F
 2239: 9D 00 37 STA SEBUF, X
 223C: CA DEX
 223D: 10 F5 BPL PLOOP2
 223F: 60 RTS

22B8: 20 D1 22
 22BB: A9 AA
 22BD: 20 D1 22
 22C0: A9 EB
 22C2: 20 D1 22
 22C5: A9 FF
 22C7: 20 D1 22
 22CA: BD 8E C0
 22CD: BD 8C C0
 22D0: 60 ERROR
 22D1: 18 WRBYTE1
 22D2: 48 WRBYTE2
 22D3: 68
 22D4: 9D 8D C0 WRBYTE3
 22D7: 1D 8C C0
 22DA: 60 RTS

JSR WRBYTE1
 LDA #5AA
 JSR WRBYTE1
 LDA #5EB
 JSR WRBYTE1
 LDA #5FF
 JSR WRBYTE1
 LDA \$C08E, X
 LDA \$C08C, X
 RTS
 CLC
 PHA
 PLA
 STA \$C08D, X
 ORA \$C08C, X
 RTS

* FLIGHT SIM BOOT1
 .OR \$0800
 .TF OBJ. BOOT1
 0800: 01 .DA #51
 0801: A9 60 BOOT1 LDA #560
 0803: 8D 01 08 STA \$801
 0806: A6 2B LDX \$2B
 0808: 8A TXA
 0809: 4A LSR
 080A: 4A LSR
 080B: 4A LSR
 080C: 4A LSR
 080D: 09 C0 ORA #5C0
 080F: 8D 5C 08 STA JUMP+2

*
 * WRITE SECTOR
 *

*
 * CHECK IF FIRST BOOT
 *

LC LOADER

2240: 38 WRITE SEC
 2241: AE 08 1E LDX SLOT
 2244: 86 AA STX ZSLOT
 2246: BD 8D C0 LDA \$C08D, X
 2249: BD 8E C0 LDA \$C08E, X
 224C: 10 03 BPL OKAY
 224E: 4C D0 22 JMP ERROR
 2251: AD 00 37 OKAY LDA SEBUF
 2254: 85 A9 STA TEMP
 2256: A9 FF LDA #5FF
 2258: 9D 8F C0 STA \$C08F, X
 225B: 1D 8C C0 ORA \$C08C, X
 225E: 48 PHA
 225F: 68 PLA
 2260: EA NOP
 2261: A0 04 LDY #54
 2263: 48 SYNC PHA
 2264: 68 PLA
 2265: 20 D2 22 JSR WRBYTE2
 2268: 88 DEY
 2269: D0 F8 BNE SYNC
 226B: A9 D5 LDA #5D5
 226D: 20 D1 22 JSR WRBYTE1
 2270: A9 AA LDA #5AA
 2272: 20 D1 22 JSR WRBYTE1
 2275: A9 AD LDA #5AD
 2277: 20 D1 22 JSR WRBYTE1
 227A: 98 TYA
 227B: A0 56 LDY #556
 227D: D0 03 BNE FIRST
 227F: B9 00 37 OTHER LDA SEBUF, Y
 2282: 59 FF 36 FIRST EOR SEBUF-1, Y
 2285: AA TAX
 2286: BD 55 1E LDA WRTBL, X
 2289: A6 AA LDX ZSLOT
 228B: 9D 8D C0 STA \$C08D, X
 228E: BD 8C C0 LDA \$C08C, X
 2291: 88 DEY
 2292: D0 EB BNE OTHER
 2294: A5 A9 LDA TEMP
 2296: EA NOP
 2297: 59 00 36 WRLOOP EOR PRBUF, Y
 229A: AA TAX
 229B: BD 55 1E LDA WRTBL, X
 229E: AE 08 1E LDX SLOT
 22A1: 9D 8D C0 STA \$C08D, X
 22A4: BD 8C C0 LDA \$C08C, X
 22A7: B9 00 36 LDA PRBUF, Y
 22AA: C8 INY
 22AB: D0 EA BNE WRLOOP
 22AD: AA TAX
 22AE: BD 55 1E LDA WRTBL, X
 22B1: A6 AA LDX ZSLOT
 22B3: 20 D4 22 JSR WRBYTE3
 22B6: A9 DE LDA #5DE

* FLIGHT SIM LCARD LOADER *

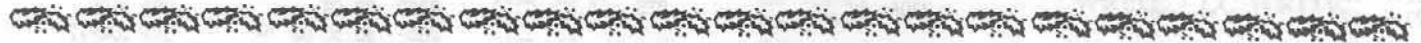
.OR \$20AF
 .TF OBJ. LC LOADER
 2110- RDTRK .EQ \$2110
 1E01- TRACK .EQ \$1E01
 1E03- INDEX .EQ \$1E03
 1E04- COUNT .EQ \$1E04
 00A5- FROM .EQ \$A5
 00A7- TO .EQ \$A7
 20AF: A2 00 LDX #50
 20B1: 8E 03 1E LOOP1 STX INDEX
 20B4: BD FE 20 LDA NUMRD, X
 20B7: F0 44 BEQ END
 20B9: 8D 04 1E STA COUNT
 20BC: BD 01 21 LDA TRCK, X
 20BF: 8D 01 1E STA TRACK
 20C2: 20 10 21 LOOP2 JSR RDTRK
 20C5: AE 03 1E LDX INDEX
 20C8: BD 03 21 LDA SCTR, X
 20CB: 18 CLC
 20CC: 69 26 ADC #526
 20CE: 85 A6 STA FROM+1
 20D0: BD 05 21 LDA BUFF, X
 20D3: 85 A8 STA TO+1
 20D5: A0 00 LDY #500
 20D7: 84 A5 STY FROM
 20D9: 84 A7 STY TO
 20DB: B1 A5 MOVE LDA (FROM), Y
 20DD: 91 A7 STA (TO), Y
 20DF: 88 DEY
 20E0: D0 F9 BNE MOVE
 20E2: DE 05 21 DEC BUFF, X
 20E5: DE 03 21 DEC SCTR, X
 20E8: 10 08 BPL OKAY
 20EA: A9 0F LDA #50F
 20EC: 9D 03 21 STA SCTR, X
 20EF: CE 01 1E DEC TRACK
 20F2: CE 04 1E OKAY DEC COUNT
 20F5: D0 CB BNE LOOP2
 20F7: AE 03 1E LDX INDEX
 20FA: E8 INX
 20FB: D0 B4 BNE LOOP1
 20FD: 60 END RTS
 20FE: 04 21 00 NUMRD .DA #504, #521, #500
 2101: 21 21 TRCK .DA #521, #521 TR
 UE TRACK MINUS 1
 2103: 0F 08 SCTR .DA #50F, #50B
 2105: F9 F3 BUFF .DA #5F9, #5F3

0812: A9 6A LDA #56A
 0814: 45 00 EOR \$0
 0816: 45 01 EOR \$1
 0818: A8 TAY
 0819: F0 1E BEQ SECOND
 081B: 2C 82 C0 BIT \$C082
 081E: 20 2F FB JSR \$FB2F
 0821: 20 58 FC JSR \$FC58
 0824: A0 40 LDY #540
 0826: 84 E6 STY \$E6
 0828: 20 F2 F3 JSR \$F3F2
 082B: 2C 50 C0 BIT \$C050
 082E: 2C 52 C0 BIT \$C052
 0831: 2C 55 C0 BIT \$C055
 0834: 2C 57 C0 BIT \$C057
 0837: A0 01 LDY #51
 0839: B9 7C 08 SECOND LDA SEC, Y
 083C: 85 51 STA \$51
 083E: B9 7E 08 LDA BUF, Y
 0841: 85 50 STA \$50
 0843: B9 80 08 LDA LAST, Y
 0846: 8D 64 08 STA COMPARE+1
 0849: B9 82 08 LDA COMMAND, Y
 084C: 8D 69 08 STA END
 * DO READ
 *
 084F: A4 51 LOOP1 LDY \$51
 0851: B9 6C 08 LDA TABLE, Y
 0854: 85 3D STA \$3D
 0856: A5 50 LDA \$50
 0858: 85 27 STA \$27
 085A: 20 5C 00 JUMP JSR \$005C
 085D: C6 50 DEC \$50
 085F: C6 51 DEC \$51
 0861: A5 51 LDA \$51
 0863: C9 06 COMPARE CMP #56
 0865: D0 E8 BNE LOOP1
 0867: A6 2B LDX \$2B
 0869: 4C 00 1D END JMP \$1D00
 086C: 00 0D 0B
 086F: 09 07 05
 0872: 03 01 TABLE .DA #500, #500
 #50B, #509, #507, #505, #503, #501
 0874: 0E 0C 0A
 0877: 08 06 04
 087A: 02 0F .DA #50E, #50C
 #50A, #508, #506, #504, #502, #50F
 087C: 06 09 SEC .DA #506, #509
 087E: 25 1F BUF .DA #525, #51F
 0880: 00 06 LAST .DA #500, #506
 0882: 60 4C COMMAND .DA #560, #54C

HAPPY HOLIDAYS! HAPPY HOLIDAYS!

HAPPY HOLIDAYS! HAPPY HOLIDAYS!

In light of the upcoming holiday season, *COMPUTIST* has decided to run a series of specials for our readers. The following pages contain a set of specials that were conceived with our special readership in mind. To take advantage of any of these offers, please refer to the special code in each box when ordering. Please hurry though, offers are only good while supplies last and expire on December 31, 1986.



X-86-01

FREE COLOR CODED LIBRARY CASE
with the purchase of 10 blank floppy disks.

DISK PRICES (includes shipping)
U.S. \$10
Canada & Mexico \$11
Other Foreign \$15

Includes tyvek sleeves, hub rings,
and write-protect tabs

X-86-04

SAVE AS MUCH AS \$19.75
on back issues when purchased in
quantities of 3.

Sale price U.S./Canada/Mexico \$10
Sale price, other foreign \$20

Regular prices \$4.75 and \$13.25, respectively.

X-86-02

**SAVE \$19.75 ON LIBRARY DISKS
AND GET A FREE CASE**
with the purchase of 5 (five) library disks.

Get 5 library disks and
a free color coded case for only \$30.00.
(Foreign orders add \$5.00 shipping & handling)

Regular prices are U.S./Canada/Mexico \$9.95
per disk and other foreign \$11.94 per disk.

X-86-05

SAVE \$5 off the price of Volume II
of the Book of Softkeys, or
SAVE \$8 off the total price when you
purchase both volumes at the same time.

Sale Prices: Vol II \$12.95, BOTH \$22.90
Regular Prices: Vol II \$17.95, Vol II \$30.90

Shipping - US/Canada/Mexico \$2 per book
Shipping - other foreign \$5 per book

X-86-03

SAVE \$3 on each back issue and
corresponding library disk

Normal retail price \$12.95 per set.
With this special only \$9.95 per set.

Foreign shipping add \$3.00.

X-86-06

GET A FREE CORE SPECIAL or
FREE COMPUTIST T-SHIRT
with any total order of \$100.00 or more.

Save On Software

Recommended Literature/Software:

- Beneath Apple DOS (Book).....Quality Software.....\$16.00
- Beneath Apple ProDOS (Book).....Quality Software.....\$16.00
- Global Program Line Editor.....Beagle Bros.....\$35.25
- Magic Window II (specify II or II/e).....Artsci.....\$106.00
- Bag of Tricks II.....Quality Software.....\$39.75

Miscellaneous Bargains:

- Dazzle Draw.....Broderbund.....\$47.50
- F-15 Strike Eagle.....Microprose.....\$28.00
- The Print Shop.....Broderbund.....\$39.75
- Flight Simulator II.....Sublogic.....\$44.00
- Night Mission Pinball.....Sublogic.....\$30.75
- Exodus Ultima III.....Origin Systems.....\$47.75
- Hitchhiker's Guide to the Galaxy.....Infocom.....\$31.00
- Witness.....Infocom.....\$31.00
- Zork III.....Infocom.....\$35.00



*Domestic Shipping/Handling: \$2/Item. Five or more items FREE.

Name _____ ID# _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone _____

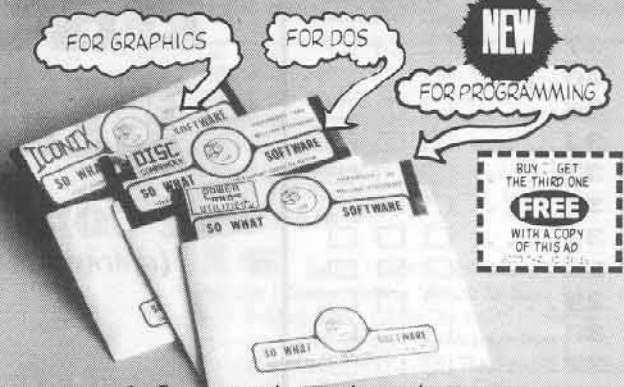
  _____ Exp. _____

Signature _____ CP37

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.

So What Software Presents MORE BANG FOR THE BUCK



3 Gigantic utility packages for your
Apple II series computers, self documenting
and unprotected!

- for dos Disc Commander _____ \$29.95
- for graphics Iconix _____ \$29.95
- for programming Power & Utilities _____ \$29.95 **NEW!**
- for free FREE catalog _____ \$00.00

FREE ~~5.50~~ SHIP & HANDLE FOREIGN-ADD ~~2.50~~ CA ~~5.00~~

SO WHAT SOFTWARE, 10221 SLATER AVE., SUITE 103,
FOUNTAIN VALLEY, CA 92708

Apple is a registered trademark of Apple Computer, Inc.

HOLIDAY SPECIAL ORDER FORM

Yes I want to take advantage of the Holiday offers mentioned on the facing page. Enclosed is U.S. funds to cover my order.

Please send me _____ sets of 10 floppy disks. I understand that I will get a FREE color coded disk case with each set.
Sale Price: U.S. \$10, Canada/Mexico \$11, Other Foreign \$15 per set.

I want to take advantage of the Book of Softkeys special prices.

- Please send me volume II of the book of softkeys. Enclosed is 12.95 plus shipping & handling. (U.S., Canada & Mexico add \$2. Other Foreign add \$5).
- Please send me Both volume I and volume II of the book of softkeys. Enclosed is \$22.90 plus shipping & handling. (U.S., Canada & Mexico add \$4. Other Foreign add \$10).

For remaining holiday sales, please consult 'Back Issues and Library Disks' order form on page 32.

Send orders to: COMPUTIST
PO Box 110846-T
Tacoma, WA 98411

Name _____ ID# _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone _____

  _____ Exp. _____

Signature _____ CP37

U.S. funds drawn on U.S. bank. In Washington add 7.8% sales tax. Most orders shipped UPS so please use street address.
Offer good while supplies last.

Description of Available Back Issues

36 *Softkeys* | Flight Simulator II v 1.05 | AutoDuel | Critical Reading | Troll's Tale | Robot War | General Manager | Plasmania | Telarium Software | Kidwriter v1.0 | Color Me | *Features* | ScreenWriter meets Flashcard | The Bus Monitor | Mousepaint for non-Apples | *Core* | The Bard's Dressing Room | *Advanced Playing Techniques* | Championship Lode Runner |

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 installing a CPS Clock Driver |

34 *Softkeys* | Crisis Mountain | Terripin Logo | Apple Logo II | Fishies 1.0 | SpellWorks | Gumball | *Readers' Softkeys* | Rescue at Rigel | Crazy 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 Bard's 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's Softkeys* | The Report Card V1.1 | Kidwriter | *Feature* | Apple II 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 I |

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 I | 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 | Zephyr Wars | Vector Graphics | Review of Bit Copiers | Boot Code Tracing | Softkey IOB | Interview with 'Mike' Markkula |

*Don't forget . . .
through X-mas, for every order
of \$100 or more, you can get a
FREE 'CORE SPECIAL'.
See page 30 for more information
on this and more special savings.*

The Super IOB Collection

NEW

- What could possibly be better than receiving COMPUTIST every month, typing in the Super IOB controllers and deprotecting your favorite software? How about having **all the controllers ever printed in COMPUTIST** at your fingertips? With *The Super IOB Collection* Volumes I & II, you have just that and more.
- Each volume (supplied on a DOS 3.3 disk) contains at least 60 Super IOB controllers including the standard, swap, newswap and fast controllers. In addition, each disk has the Csave program from COMPUTIST No. 13. But wait! You also get version 1.5 of Super IOB and a menu hello program that lists the available controllers and, when you select one, automatically installs it into Super IOB and RUNs the resulting program.*
- Several of the controllers deprotect the software completely with no further steps. This means that some programs are only minutes away from deprotection (with virtually no typing).
- The issue of COMPUTIST in which each controller appeared is indicated in case further steps are required to deprotect a particular program.†

Volume 1

Volume 1 of the Super IOB collection covers all the controllers appearing in COMPUTIST No. 9 through No. 26. In addition, the newswap and fast controllers from COMPUTIST No. 32 are included. The following 60 controllers are on volume 1:

Advanced Blackjack, Alphabet Zoo, Arcade Machine, Archon II, Archon, Artsci Software, Bank Street Writer, Barrons SAT, Beyond Castle Wolfenstein, BSW //c Loader, Castle Wolfenstein, Computer Preparation: SAT, Dazzle Draw, DB Master 4 Plus, Death in the Carribean, Dino Eggs, DLM Software, Electronic Arts, F-15 Strike Eagle, Fast Controller, Fathoms 40, Financial Cookbook, Gessler Software, Grandma's House, The Heist, In Search of the Most Amazing Thing, Instant Recall, Kidwriter, Lions Share, Lode Runner, Mastertype, Match Maker, Miner 2049er, Minit Man, Mufplot, Newsroom, Newswap controller, Penguin Software, Print Shop Graphic Library, Print Shop, Rendezvous with Rama, Rockys' Boots, Sargon III, Sea Dragon, Shiela, Skyfox, Snooper Troops, Standard controller, Stoneware Software, Summer Games, Super Controller, Super Zaxxon, Swap Controller, TAC, Ultima III, Word Challenge, Xyphus, Zaxxon

Volume 2

Volume 2 of the Super IOB collection covers all the controllers appearing in COMPUTIST No. 27 through No. 38. The following 65 controllers are on volume 2:

Alice in Wonderland, Alphabetic Keyboarding, Alternate Reality, Autoduel, Checkers, Chipwits, Color Me, Conan.data, Conan.prog, CopyDOS, Crisis Mountain, Disk Director, Dragonworld, Early Games, Easy as ABC, F-15 Strike Eagle, Fantavision, Fast controller, Fishies, Flight Simulator, Halley Project, Hartley Software (a), Hartley Software (b), Jenny of the Prairie, Jingle Disk, Kidwriter, Kracking Vol II, Lode Runner, LOGO II (a), LOGO II (b), Masquerade, Mastering the SAT, Microtype: The Wonderful World of Paws, Microzines 1, Microzines 2-5, Miner 2049er, Mist & View to a Kill, Murder on the Zinderneuf, Music Construction Set, Newswap controller, Olympic Decathlon, Other Side, Phi Beta Filer, Pitstop II, Print Shop Companion, RDOS, Robot War, Spy vs Spy, Standard controller, Sundog V2, Swap controller, Sword of Kadash, Synergistic Software, Tawala's last Redoubt, Terripin Logo, Threshold, Time is Money, Time Zone, Tink! Tonk!, Troll's Tale, Ultima IV, Wilderness, Word Attack & Classmate, World's Greatest Baseball, World's Greatest Football

To Order: Send \$9.95 for each volume or \$19.95 for a complete package that includes: both disks, a reprint of "Disk Inspection and the use of Super IOB" and COMPUTIST No. 32. U.S. funds drawn on U.S. banks. Foreign orders (other than Canada or Mexico) add 20% shipping. Washington state residents add 7.8% sales tax. Mail orders to: Super IOB Collection; POB 110846; Tacoma, WA 98411.

*Requires at least 64K of memory.

†Although some controllers will completely deprotect the program they were designed for, some will not and therefore require their corresponding issue of COMPUTIST to complete the deprotection procedure.