

HOW TO BACKUP, UNLOCK, OR MODIFY COPY-PROTECTED SOFTWARE

Hardcore

COMPUTIST

Issue No. 12

\$2.50

**Deprotecting
Zoom Graphix**
Pg. 9

Psychedelic Symphony
Pg. 16

**The CORE
Disk Searcher**
Pg. 19

**Pseudo-ROMS On The
Franklin ACE**
Pg. 24



Softkey for Lion's Share
Pg. 14

(CA5/340) 5 issues

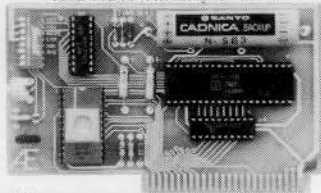
Hardcore COMPUTIST
P.O. Box 44549-T
Tacoma, WA 98444

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

APPLIED ENGINEERING IS 100% APPLE

That's Why We're So Good At It!

THE NEW TIMEMASTER II



Automatically date stamps files with PRO-DOS

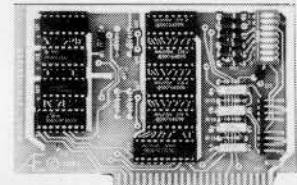
NEW 1984 DESIGN
An official PRO-DOS Clock

- Just plug it in and your programs can read the year, month, date, day, and time to 1 millisecond! The only clock with both year and ms.
- A rechargeable NiCad battery will keep the TIMEMASTER II running for over ten years.
- Powerful 2K ROM driver — No clock could be easier to use.
- Full emulation of most other clocks, including Thunderclock and Appleclock (but you'll like the TIMEMASTER II mode better). We emulate other clocks by merely dropping off features. We can emulate them but they can't emulate us.
- Basic, Machine Code, CP/M and Pascal software on 2 disks!
- Eight software controlled interrupts so you can execute two programs at the same time (many examples are included).
- On-board timer lets you time any interval up to 48 days long down to the nearest millisecond.

The TIMEMASTER II includes 2 disks with some really fantastic time oriented programs (over 40) including appointment book so you'll never forget to do anything again. Enter your appointments up to a year in advance then forget them. Appointment book will remind you in plenty of time. Plus DOS dater so it will automatically add the date when disk files are created or modified. The disk is over a \$200.00 value along—we give the software others sell. All software packages for business, data base management and communications are made to read the TIMEMASTER II. If you want the most powerful and the easiest to use clock for your Apple, you want a TIMEMASTER II.

PRICE \$129.00

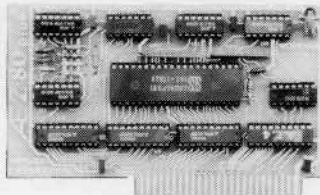
Super Music Synthesizer Improved Hardware and Software



- Complete 16 voice music synthesizer on one card. Just plug it into your Apple, connect the audio cable (supplied) to your stereo, boot the disk supplied and you are ready to input and play songs.
- It's easy to program music with our compose software. You will start right away at inputting your favorite songs. The Hi-Res screen shows what you have entered in standard sheet music format.
- Now with new improved software for the easiest and the fastest music input system available anywhere.
- We give you lots of software. In addition to Compose and Play programs, 2 disks are filled with over 30 songs ready to play.
- Easy to program in Basic to generate complex sound effects. Now your games can have explosions, phaser zaps, train whistles, death cries. You name it, this card can do it.
- Four white noise generators which are great for sound effects.
- Plays music in true stereo as well as true discrete quadraphonic.
- Full control of attack, volume, decay, sustain and release.
- Will play songs written for ALF synthesizer (ALF software will not take advantage of all our card's features. Their software sounds the same in our synthesizer.)
- Our card will play notes from 30HZ to beyond human hearing.
- Automatic shutoff on power-up or if reset is pushed.
- Many many more features.

PRICE \$159.00

Z-80 PLUS!



- TOTALLY compatible with ALL CP/M software.
- The only Z-80 card with a special 2K "CP/M detector" chip.
- Fully compatible with microdisk disks (no pre-boot required).
- Specifically designed for high speed operation in the Apple IIe (runs just as fast in the II+ and Franklin).
- Runs WORD STAR, dBASE II, COBOL-80, FORTRAN-80, PEACHTREE and ALL other CP/M software with no pre-boot.
- A semi-custom I.C. and a low parts count allows the Z-80 Plus to fly thru 4M/M programs at a very low power level. (We use the Z-80A at fast 4MHZ.)
- Does EVERYTHING the other Z-80 boards do, plus Z-80 interrupts.

Don't confuse the Z-80 Plus with crude copies of the microsoft card. The Z-80 Plus employs a much more sophisticated and reliable design. With the Z-80 Plus you can access the largest body of software in existence. Two computers in one and the advantages of both, all at an unbelievably low price.

PRICE \$139.00

Viewmaster 80

There used to be about a dozen 80 column cards for the Apple, now there's only ONE.

- TOTALLY Videx Compatible.
- 80 characters by 24 lines, with a sharp 7x9 dot matrix.
- On-board 40/80 soft video switch with manual 40 column override
- Fully compatible with ALL Apple languages and software—there are NO exceptions.
- Low power consumption through the use of CMOS devices.
- All connections are made with standard video connectors.
- Both upper and lower case characters are standard.
- All new design (using a new Microprocessor based C.R.T. controller) for a beautiful razor sharp display.
- The VIEWMASTER incorporates all the features of all other 80 column cards, plus many new improvements.

	PRICE	BUILT IN SOFTSWITCH	SHIFT KEY SUPPORT	LOW POWER DESIGN	80 COLUMN MENU	7X9 DOT MATRIX	LIGHT PEN INPUTS	40 COLUMN OVERRIDE	INVERSE CHARACTERS
VIEWMASTER 169		YES	YES	YES	YES	YES	YES	YES	YES
SUPRTERM	MORE	NO	YES	NO	NO	NO	NO	YES	YES
WIZARD80	MORE	NO	NO	NO	NO	YES	NO	YES	YES
VISION80	MORE	YES	YES	NO	NO	YES	NO	NO	NO
OMNIVISION	MORE	NO	YES	NO	NO	NO	NO	YES	YES
VIEWMAX80	MORE	YES	YES	NO	NO	YES	NO	NO	YES
SMARTERM	MORE	YES	YES	NO	NO	NO	YES	YES	NO
VIDEOTERM	MORE	NO	NO	YES	NO	YES	YES	NO	YES

The VIEWMASTER 80 works with all 80 column applications including CP/M, Pascal, WordStar, Format II, Easywriter, Apple Writer II, VisiCalc, and all others. The VIEWMASTER 80 is THE MOST compatible 80 column card you can buy at ANY price!

PRICE \$179.00

- Expands your Apple IIe to 192K memory.
- Provides an 80 column text display.
- Compatible with all Apple IIe 80 column and extended 80 column card software (same physical size as Apple's 64K card).
- Can be used as a solid state disk drive to make your programs run up to 20 times FASTER (the 64K configuration will act as half a drive).
- Permits your IIe to use the new double high resolution graphics.
- Automatically expands Visicalc to 95 K storage in 80 columns! The 64K config. is all that's needed, 128K can take you even higher.
- PRO-DOS will use the MemoryMaster IIe as a high speed disk drive.

MemoryMaster IIe 128K RAM Card

- Precision software disk emulation for Basic, Pascal and CP/M is available at a very low cost. NOT copy protected.
 - Documentation included, we show you how to use all 192K.
- If you already have Apple's 64K card, just order the MEMORYMASTER IIe with 64K and use the 64K from your old board to give you a full 128K. (The board is fully socketed so you simply plug in more chips.)

MemoryMaster IIe with 128K \$249
Upgradeable MemoryMaster IIe with 64K \$169
Non-Upgradeable MemoryMaster IIe with 64K \$149

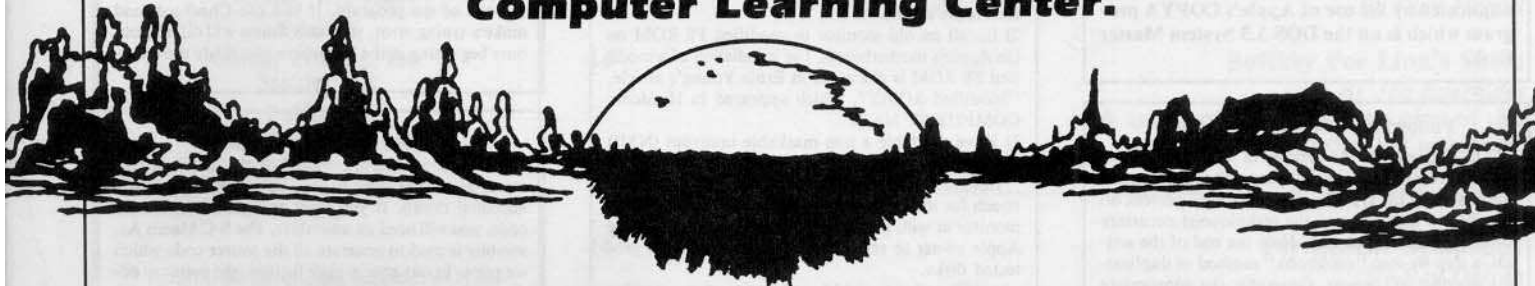
Our boards are far superior to most of the consumer electronics made today. All I.C.'s are in high quality sockets with mil-spec. components used throughout. P.C. boards are glass-epoxy with gold contacts. Made in America to be the best in the world. All products work in the APPLE IIe, II+, II+ and Franklin. The MemoryMaster IIe is IIe only. Applied Engineering also manufactures a full line of data acquisition and control products for the Apple: A/D converters and digital I/O cards, etc. Please call for more information. All our products are fully tested with complete documentation and available for immediate delivery. All products are guaranteed with a no hassle THREE YEAR WARRANTY.

Texas Residents Add 5% Sales Tax
Add \$10.00 If Outside U.S.A.
Dealer Inquiries Welcome

Send Check or Money Order to:
APPLIED ENGINEERING
P.O. Box 798
Carrollton, TX 75006

Call (214) 492-2027
8 a.m. to 11 p.m. 7 days a week
MasterCard, Visa & C.O.D. Welcome
No extra charge for credit cards

**Whether You're A Potential Mozart
Or A Wall Street Wizard,
You Can Gain Greater Insight Into Your Profession
With These Disk Packages From The
Computer Learning Center.**



Take Your Pick Of The TOP Ten
Best Selling Public Domain Software Packages
At The Low Cost Of \$4.00 Per Volume:

- | | |
|-----------------------------------------|-----------------------------------------------|
| Bus&Fin 19 <input type="checkbox"/> | VisiCalc, Stocks, And Letters |
| Bus&Fin 20 <input type="checkbox"/> | Inventories, Home Finance & Stock Forecasting |
| Util 86 <input type="checkbox"/> | Exercise Your Epson |
| AppTut 1 <input type="checkbox"/> | Teach Yourself BASIC |
| Math&Stats 61 <input type="checkbox"/> | Regressions, Permutations, And Quadratics |
| Math&Stats 59 <input type="checkbox"/> | Statistics, Curves, And Calculus |
| Educ 28 <input type="checkbox"/> | School Daze |
| Hello&Menu 58 <input type="checkbox"/> | HELLOHELLOHELLOHELLO |
| Game 41 <input type="checkbox"/> | Star Trek |
| Music&Sound 65 <input type="checkbox"/> | Music & Mozart |

OR Send For Our New PDS Catalog which lists
over 140 PDS volumes.*

Yes, rush me the PDS volumes I have checked above. HC12
 Please send me the new PDS Catalog.

Name _____
 Address _____
 City _____ St _____ Zip _____
 Country _____
 VISA/MC _____ Exp _____
 Signature _____

Send to: **Computer Learning Center, P.O. Box 45202HC, Tacoma, WA 98445**
 US funds only. Washington residents add 7.8% sales tax. Foreign orders add 20% shipping and handling.

* Most Public Domain Software supplied by the Computer Learning Center has no copyright attached and may be copied and distributed by anyone without regard for origin or ownership. Public Domain Software is not commercial quality, and is supplied as-is.

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

Hardcore COMPUTIST also contains a center CORE section which generally 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.

New readers are advised to read over the rest of this page carefully in order to avoid frustration when following any of the softkeys or typing in any of the programs printed in this issue. Longtime readers should know what to do next: Make a pot of coffee, get out some blank disks and settle in for a long evening at the keyboard.

What Is a Softkey Anyway?

A softkey is a term which we coined to describe a procedure that removes, or at least circumvents, any copy protection that may be present on a disk. Once a softkey procedure has been performed, the disk can usually be duplicated by the use of Apple's COPYA program which is on the DOS 3.3 System Master Disk.

Following A Softkey Procedure

The majority of the articles in Hardcore COMPUTIST which contain a softkey will also include a discussion of the type of copy protection present on the disk in question and the technique(s) necessary to remove that protection. Near the end of the article, a step-by-step "cookbook" method of duplicating the disk will appear. Generally, the appropriate actions for the reader to perform will appear in bold-face type. Examples are:

1) Boot the disk in slot 6

PR#6

or

2) Enter the monitor

CALL -151

It is assumed that the reader has some familiarity with his or her Apple, i.e. knowing that the RETURN key must be hit following the commands illustrated above.

Hardcore COMPUTIST tries to verify the softkeys which are published, although occasionally this is not possible. Readers should be aware that different, original copies of the same program will not always contain an identical protection method. For this reason, a softkey may not work on the copy of a disk that you own, but it may work on a different copy of the same program. An example of this is Zaxxon, by Datasoft, where there are at least 3 different protection methods used on various releases of the game.

Software Recommendations

Although not absolutely necessary, the following categories of utilities are recommended for our readers who wish to obtain the most benefit from our articles:

- 1) **Applesoft Program Editor** such as Global Program Line Editor (GPLE).
- 2) **Disk Editor** such as DiskEdit, ZAP from Bag of Tricks or Tricky Dick from The CIA.
- 3) **Disk Search Utility** such as The Inspector, or The Tracer from The CIA.
- 4) **Assembler** such as the S-C Macro Assembler or Big Mac.
- 5) **Bit Copy Program** such as COPY II+, Locksmith or The Essential Data Duplicator.
- 6) **Text Editor** capable of producing normal sequential text files such as Applewriter II, Magic Window II or Screenwriter II.

Three programs on the DOS 3.3 System Master Disk, COPYA, FID and MUFFIN, also come in very handy from time to time.

Hardware Recommendations

Certain softkey procedures require that the computer have some means of entering the Apple's system monitor during the execution of a copy-protected program. For Apple II+ owners there are three basic ways this can be achieved:

- 1) Place an INTEGER BASIC ROM card in one of the Apple's slots.
- 2) Install an old monitor or modified F8 ROM on the Apple's motherboard. The installation of a modified F8 ROM is discussed in Ernie Young's article, "Modified ROMS", which appeared in Hardcore COMPUTIST No. 6.
- 3) Have available a non-maskable interrupt (NMI) card such as Replay or Wildcard.

Longtime readers of Hardcore COMPUTIST will vouch for the fact that the ability to RESET into the monitor at will, greatly enhances the capacity of the Apple owner to remove copy protection from protected disks.

A 16K or larger RAM card is also recommended for Apple II or II+ owners. A second disk drive is handy, but is not usually required for most programs and softkeys.

Requirements

Most of the programs and softkeys which appear in Hardcore COMPUTIST require an Apple II+ computer (or compatible) with a minimum 48K of RAM and at least one disk drive with DOS 3.3. Occasionally, some programs and procedures have special requirements such as a sector editing program or a "nonautostart" F8 monitor ROM. The prerequisites for deprotection techniques or programs will always be listed at the beginning article under the "Requirements:" heading.

Recommended Literature

The Apple II and II+'s come bundled with an Apple Reference Manual, however this book is not included with the purchase of an Apple //e. This book is necessary reference material for the serious computerist. A DOS 3.3 manual is also recommended.

Other helpful books include:

Beneath Apple DOS, Don Worth and Peter Lechner, Quality Software. \$19.95.

Assembly Lines: The Book, Roger Wagner, Softalk Books. \$19.95.

What's Where In The Apple, Professor Lubert, Micro Ink. \$24.95.

Typing in BASIC Programs

When typing in basic programs, you will often encounter a delta ("Δ") character. These are the spaces you MUST type in if you wish your checksums to match ours. All other spaces are merely printed for easier reading and don't have to be keyed in. Any spaces after the word DATA that aren't delta characters MUST be omitted!

It is a good idea to SAVE your BASIC program to disk frequently while typing it in to minimize the loss of data in the event of a power failure.

Checksum

Checksum is a Binary program that checks Applesoft programs to ensure that you have keyed them in properly. Every bin program we print has companion checksums which consist of the Applesoft program's line numbers and a hexadecimal (base 16) number for each line. After keying in a BASIC program, BRUN checksum and compare the checksums for every line that Checksoft generates with those at the end of the program. If you use Checksoft and make a typing error, your checksums will differ from ours beginning at the line where you made the error.

Typing in Binary Programs

Binary programs are printed in two different formats, as source code and as object code in a hexadecimal dump. If you want to type in the source code, you will need an assembler. The S-C Macro Assembler is used to generate all the source code which we print. In our source code listings, the memory address of the each instruction is printed at the beginning of every line (instead of the line number).

Binary programs can also be entered directly with the use of the Apple monitor by typing in the bytes listed in the hexdump at the appropriate addresses. Be sure to enter the monitor with a CALL -151 before entering the hexdump. Don't type the checksums printed at the end of each line of the hexdump and don't forget to BSAVE binary programs with the proper address and length parameters listed in the article.

Checkbin

Like Checksoft, Checkbin also generates checksums, but was designed to check binary (machine language) programs.

Whenever Hardcore COMPUTIST prints a hexdump to type in, the associated Checkbin generated checksums are printed after every 8 bytes and at the end of every line.

Checksoft and Checkbin were printed in Hardcore COMPUTIST No. 1 and the Best Of Hardcore Computing and are sold on Program Library Disk No. 1 and the Best Of Hardcore Library Disk.

Let Us Hear Your Likes and Grips

New and longtime readers of Hardcore COMPUTIST are encouraged to let us know what they like and don't like about our magazine by writing letters to our INPUT column. Our staff will also try to answer questions submitted to the INPUT column, although we cannot guarantee a response due to the small size of our staff. Also, send your votes for the softkeys you would like to see printed to our "Most Wanted List."

How-To's Of Hardcore

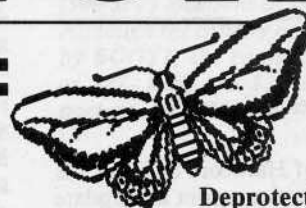
If you are reading our magazine for the first time, welcome to Hardcore COMPUTIST, a publication devoted to the serious user of Apple II and Apple II compatible computers. We believe our magazine contains information you are not likely to find in any of the other major journals dedicated to the Apple market.

Our editorial policy is that we do NOT condone software piracy, but we do believe that honest users are entitled to back up 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.

Hardcore

COMPUTIST

THIS ISSUE:



Publisher/Editor

Charles R. Haight

Technical Editors

Ray Darrah Gary Peterson

Production & Graphics

Lynn Campos-Johnson

Circulation

Valerie Robinson Michelle Frank

Business Manager

Ken Fields

Advertising

Attn: Valerie Robinson
Advertising Department
3710 100th Street SW
Tacoma, WA 98499

Printing

Grange Printing, Inc.
Seattle, WA

Publishing

Softkey Publishing
P.O. Box 45201
Tacoma, WA 98445
USA

Address all advertising inquiries to Hardcore COMPUTIST, Advertising Department, 3710 100th St. SW, Tacoma, WA 98499. Address all manuscripts and editorials to: Hardcore COMPUTIST, Editorial Department, P.O. Box 44549K, Tacoma, WA 98444.

Return postage must accompany all manuscripts, drawings, photos, disks, or tapes if they are to be returned. No responsibility can be assumed for unsolicited manuscripts. We suggest you send only copies.

Entire contents copyright 1984 by Soft-Key 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 Hardcore COMPUTIST magazine or Soft-Key Publishing.

SUBSCRIPTION INFO: Rates for one year are as follows: U.S. \$25, 1st Class, APO/FPO, and Canada \$34, Mexico \$39, Foreign Airmail \$60, Foreign Surface Mail \$40. Subscription inquiries should be directed to Hardcore COMPUTIST, Subscription Department, P.O. Box 44549-T, Tacoma, WA 98444.

DOMESTIC DEALER RATES sent upon request, or call (206) 581-6038.

Apple usually refers to the Apple II or II Plus computer and is a trademark of Apple Computers, Inc.

This month's cover: A hi-res graphic taken from The Lion's Share (copyright, Davka Corporation).

9 **Deprotecting Zoom Grafix**
By Michael Decker
Blow-up the protection of Zoom Grafix and add your own modifications.

11 **Deprotecting Flip Out**
By Clay Harrell

14 **Softkey For Lion's Share**
By Jan Eugenides
A quick softkey and a discussion of this semi-universal deprotection technique.

CORE Section:

16 **Psychedelic Symphony**
By Ray Darrah
Convert your favorite music into a dazzling geometric display.

19 **The CORE Disk Searcher**
By Bryce L. Fowler & Ray Darrah
Scan even protected disks for sequences of bytes.

23 **The Armonitor**
By Nick Galbreath
Know where your head is at during DOS 3.3 actions.

24 **Pseudo-ROMs On The Franklin ACE**
By Ken Stutzman
Fix those no ROM card blues, permanently.

27 **Backup And Modify Music Construction Set**
By Dan Rosenberg

DEPARTMENTS

4 **Input**

6 **Readers' Softkey And Copy Exchange**

Using Super IOB To Copy Hi-Res Computer Golf II
By Jeff Rivett

Deprotecting Suicide
By Clay Harrell

Backup For Sabotage
By Clay Harrell

Softkey For Millionaire
By Bill Wilson

Deprotecting Time Is Money
By Rod Wideman

Backing Up Type Attack
By Jerry Caldwell

13 **Adventure Tips**

INPUT INPUT INPUT

A Reader's Favorites

The How-To's of Hardcore on Page 2 is a super idea; I hope you reprint and update it occasionally.

Under Hardware Recommendations, I suggest you add the Track Star (as advertised in Hardcore). I just bought one and wouldn't be without it now. It's one neat tool for copying. Disk Search Utility...I suggest you add Nibbles Away...can be used to search a *protected* disk (see Appendix E of the C-1 manual) where Inspector won't. Under Disk Editor you suggested Tricky Dick from the CIA package. This program is very useful but, in my opinion, has one serious drawback: it does not display sync bytes in inverse video as do the editors from Copy II+ and Nibbles Away. To anyone who does as much hunting for address markers and insert marks as I do, sync bytes in inverse video are almost a requirement. I would suggest you add the latter two editors to your list. Under Software Recommendations, I suggest you drop Locksmith and add Nibble Away and Back It Up III. Finally, under Recommended Literature, I would suggest Call A.P.P.L.E.'s ALL ABOUT DOS; it has an associated disk available and includes many very useful programs and techniques that I have not seen anywhere else.

I've been enjoying Clay Harrell's softkeys very much. I like his style and in-depth explanations of why he does what he does. I was pleased to see his valuable softkey on the Sierra software written with Doni Grande.

Ray Darrah's Super IOB is really SUPER! As the man said, "it works"! I think this program, in wide use (as it surely will be) will serve as a milestone in the software backup game with the publishers. As you know by now, the program listing has a typesetting error- it's missing a blank in line 380 after the word COPY and before the "=" . This throws off the checksums. The controllers were set almost right in format, but the first line of the Standard Controller should be line number 1000 not 0000. On the subject of typesetting, I hope the typesetter can get some slashed zeroes to use to distinguish the zeroes from the oh's in the listings you give them. Also, they tend to customize the listings occasionally breaking the literals so we can't tell how many blanks to use between certain words or symbols. Also, it would have helped a lot if you had published the checksums for the three controllers; I spent a lot of time looking for controller. Finally, Ray suggested that the controllers be typed/added to Super IOB. Another way is to use the DOS Tool Kit (every Hardcore

reader should have a copy) to &HOLD the Super IOB and &MERGE the controller to it. This way the controllers could be saved separately and easily modified for the next application, then &MERGED with Super IOB when needed (saves disk space).

About Super IOB...I really got a lot of value out of it as I used it first on a standard disk, as Ray suggested, and then on Castle Wolfenstein and Visible Computer:6502, and they all worked!! It also was helpful for you to publish several other examples using Super IOB in the same issue. Three cheers for Ray!!

I finally found one good use for Locksmith 5.0 (Rev. D.) It won't backup any of my still-protected software, but Omega's 16 Sector FAST DISK BACKUP program sure works well. The way to beat the long load time problem is to use a copy card (like Wildcard 2) to make a BLOADable program of it. Then copy (with Copy II+) the program onto your favorite utility disk having a fast DOS (I prefer Pronto DOS and David DOS). This program is faster and more reliable than The Pirate's Friend copier (COPY II Version 3.0) and Disk Muncher and faster, too. But don't forget to type "V" to invoke the Verify option before running the program!

I missed Ray Darrah's column, Whiz Kid, and look forward to reading it in future issues.

Ken Burnell
Adh-Dhahran
Kingdom Of Saudi Arabia

Lowercase For Deadline

Could you pass this note to the editorial crew? In an earlier issue, a gentleman was looking for the lower case output for the older version of ZORK, and you replied that it was found in the same sector but different byte. I beg to differ. I found that one, but it didn't do the trick. I did, on the other hand, find the spots on my release on Deadline.

Step 3: The code to change to deactivate uppercase conversion is at bytes 0A to 14.

Step 7: The byte to change from 3F to FF is at byte 55.

Step 8: The byte to change is BF from 01 to 02.

Mark Provence
Sunnymead CA

Locksmith 5.0 Bug

The early version of Locksmith 5.0 had an

error in the setting of one of the default parameters (this includes the pirated WOMBAT version). If you change default value #151 from FF to 0D (and also #13E to 01), you will find Locksmith 5.0 to be a very good nibbler (with parameter set #151 set to FF, it can't even copy standard DOS!)

RAC
Detroit MI

Legacy Help

Here's a way to cheat in Legacy of Llylgamyn. Since certain levels can only be accessed by good and evil characters, I found it much easier to change the class of my characters than to have to create a good and evil party to explore the dungeon. To do this:

Backup your characters using the Character backup program.

Finding The Character

Run Diskedit 2.1, insert your backup disk and press space bar. Now press the A key. The display will change to letters and a character's name should appear at the top left corner of the screen. If this is the character that you want to edit, skip to Editing the Character Section. Otherwise, press the N key to read the next sector, until you find that character or until you reach Track 3. If you're at Track 3, start over.

Editing The Character

Press the H key once you have found the character you wish to change (this changes the letters back to numbers). Next, press the O key, then type "2A" (this moves the cursor to position 2A). Now the cursor will be pointing at one of three numbers: 01, 02, or 03 (good, neutral, evil). Now press the E key and enter the new alignment (01, 02, 03) and press the ESC key. Then press the W key followed by 3 RETURNs.

Now, simply recover the backup onto the scenario and you have your newly aligned characters on your Wizardry scenario.

The character alignment information was developed by my friend Brain N.

Here's another hint for Legacy of Llylgamyn: You can't kill Llylgamyn. You need the neutral crystal.

Hope some of this stuff is useful.

Keith Tripp
Attleboro MA

**More Parameters
For Copy II Plus 4.4c**

I am a five month subscriber to your magazine, and I enjoy it very much. I don't mean to bother you, but I am a little upset with your reviews of the copy programs (Issue No. 8, p. 27). Essential Data Duplicator is a good quality copy program, but to me, it's not the best. To tell you the truth, I don't think there is a "best" copy program.

I just received my updated version of Copy II + (4.4c), and it's terrific! It will copy Bank Street Writer (Broderbund) without parameter changes. Just copy Tracks 0-22. If the copy hangs up during the boot process, just re-copy Tracks 1A-22 for Apple II or II+, or Tracks 1B-1E for Apple //e. Easy, isn't it?

Here are some more parameters for Copy II Plus, Version 4.4c:

Face Maker-Copy II Plus 4.4c

Tracks 0-22 change parameters to: 3E=2, 34=1, 37=1B

Story Machine-Copy II Plus 4.4c-Bit Copy

Tracks 0-22, change parameters to: 3E=2, 34=1, 35=2A, 37=1B, 38=FC

Millionaire

Tracks 0-22

Rocky's Boots

Tracks 0-22

Flight Simulator II

Tracks 0-22: Synchronized (Keep track length)

Dollars & Sense

Boot disk, use copy from main menu, then:
Sector edit the System Disk Track 0, Sector 3.
Change starting at address 8C to:

```
EA A5 02 38 E9 40 85 04 A5 03 E9 00
85 05 A0 3F B1 04 91 02 88 10 F9
```

Use copy from main menu, and you will have a copy of Dollars and Sense

Master Type

Tracks 0-22: Retry Track 0 until backup works. The drive speed helps (it should be about 201.5ms)

Jeff Lucia

West Caldwell NJ

Boot Code Problem

With a subscription to your magazine I have begun to see some light at the end of that dark copy-protection tunnel.

However, could you please solve the following problem I am having with boot code tracing a normal unprotected DOS 3.3 disk? The problem occurs when I try to execute the BOOT1 code at \$801 after it has been loaded by the disk controller card in slot 6. From the monitor I typed in the following instructions. My intent was to move the code in the disk controller card down to \$9600 and modify it so that the disk drive motor is turned off and the Apple's monitor is entered at \$FF59 after track \$0, sector \$0 has been read in.

**9600<C600.C6FFM
96F8:8D E8 C0 4C 59 FF**

9600G

Everything worked fine until I tried to execute the code at \$801 by typing

801G

Instead of the disk coming on and proceeding to load in the rest of DOS, the only thing that happened was a crash at \$C061 in memory.

Reading 'Beneath Apple DOS' was not enough to help me with the problem. I thought location \$2B contained the disk slot #, but \$2B contains an \$07 instead of a \$06. What gives?

Michel Soulimovsky
Guadeloupe
French West Indies

Mr. Soulimovsky: At first glance it seems as if your technique should work successfully. However, there are a couple of details about the BOOT1 code that you have overlooked.

Like most subroutines, the BOOT1 subroutine at \$801-\$8FF has certain entry conditions which must be met when the subroutine is entered. Two of the conditions

for entry to the BOOT1 code are:

- 1) The disk drive motor must be revolving at the proper speed.
- 2) Proper values in certain zero page (\$00-\$FF) locations of the Apple's memory. Although the entirety of zero page is not used by BOOT1, the contents of locations such as \$27, \$2B, \$3E-\$3F and \$40-43 are important.

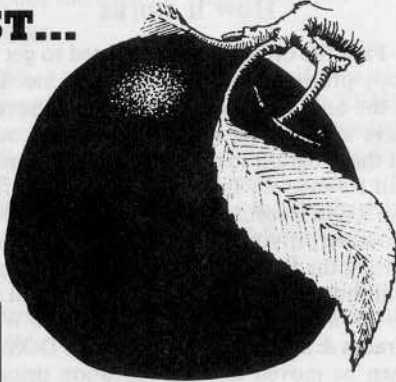
Continued on page 26

We would like to thank all those Hardcore COMPUTIST readers who took the time to send a postcard with the arrival date of their issue of HC No. 10. With this information, we will be better able to track the efficiency of our mailing system and hopefully, provide better service and shorter mailing times to subscribers.

Circulation Department

Hardcore COMPUTIST...

- Make backups more easily
- Move software from floppy to hard disk
- Add custom modifications such as fast-DOS to speedup LOADs and SAVEs



NOT PIRACY

Just Good Sense!

Annual Subscription Rates:

Please check one

<input type="checkbox"/> U.S.	\$25
<input type="checkbox"/> Canada, U.S. 1st Class, APO/FPO	\$34
<input type="checkbox"/> Mexico	\$39
<input type="checkbox"/> Foreign Airmail	\$60
<input type="checkbox"/> Foreign surface mail	\$40
<input type="checkbox"/> SAMPLE, U.S.	\$3.50
<input type="checkbox"/> SAMPLE, Foreign	\$4.50

() Yes, start my subscription now.
 () I would like to RENEW my subscription. (Please paste PRESENT MAILING LABEL below)

NAME _____

ADDRESS _____

CITY _____ ST _____ ZIP _____

COUNTRY _____

VISA/MC _____ EXP _____

SIGNATURE _____

Send order to: Hardcore COMPUTIST Subscription Department, P.O. Box 44549-T, Tacoma, WA 98444

READERS' SOFTKEY & COPY EXCHANGE

Using Super IOB To Copy Hi-Res Computer Golf II

By Jeff Rivett

High-Res Computer Golf II
Avante-Garde Creations
P.O. Box 30160
Eugene, OR 97403
\$34.95

Requirements:

Apple II+ or equivalent
Super IOB from Hardcore COMPUTIST 9
Super IOB Swap Controller
Two blank disks

Using the following method will help you to remove the copy protection from a lot of older software. Although simple, it's a great solution when your program is constantly accessing the disk and the DOS is reasonably close to DOS 3.3. It's well worth trying in many cases, because it is so easy.

How it works

First the original disk is booted to get the copy-protected DOS into the machine. This is the only tricky part because you have to have some means of stopping the execution of the program so that the Apple's monitor can be entered. In Avant-Garde's Golf II, the RESET vector is set to re-boot the disk. However, if you lift the drive door prior to hitting the RESET key twice, the disk drive will shut off and the protected DOS will still be in memory. The RWTS (Read/Write Tracks & Sectors) of the protected DOS can then be moved to a safe location prior to booting up with a normal DOS disk. The protected RWTS can then be saved and utilized by Super IOB, with the Swap Controller installed, to read the protected disk. For Golf, you only have to leave the disk drive door open to capture the RWTS, but for most other programs you will need an "old monitor" F8 ROM or one of the Non-Maskable Interrupt (NMI) copy cards to do the trick.

When Super IOB has the Swap Controller installed, it will load in a protected disk's RWTS at \$1900-\$2100. When it comes time to read the protected disk, this RWTS is moved into \$B800-\$BFFF. For writing, the normal RWTS is moved back into \$B800-\$BFFF. In other words, the disk is deprotected by reading it with a protected RWTS and writing the copy with a standard RWTS.

Let's try this with Golf II.

1) Boot the Golf II disk and when the "DO YOU WANT TO SEE THE INTRO?"

prompt appears, open the disk drive door and hit the RESET key. Hit the RESET key a second time to turn off the drive. The Golf DOS will still be in memory.

2) Enter the monitor and move the Golf RWTS down to \$1900

CALL -151
1900 < B800.BFFFM

3) Boot up with a slave DOS 3.3 disk which has a small or null HELLO program

C600G

4) Insert the disk which has Super IOB on it and BSAVE the Golf RWTS

BSAVE GOLF.RWTS, A\$1900, L\$800

5) Load in the Super IOB program

LOAD SUPER IOB

6) Type in or EXEC in the Swap Controller and make sure that the file BLOADED in line 10010 is called GOLF.RWTS.)

7) Run Super IOB and copy both sides of the Golf disk. The Golf DOS should NOT be copied.

Golf II has a strange bug that can be fixed by adding a line to the Applesoft file called "SWING.PRACTICE.FP". To fix this bug, LOAD the file and add this line:

2 POKE 16611,1

This brings up my favorite point. Once the protection has been removed from a commercial program, you can modify the program in any way you like. In addition, you can usually use a fast DOS, such as ProntoDOS or Diversi-DOS to speed up the program's disk I/O.

Using The Swap Controller On Other Disks

If you use the technique I have just outlined on another disk and it seems to work (except for the fact that the copied program will not run), you may be able to figure out what is going on by tracing the now readily accessible code and making modifications where necessary. This is something that cannot be taught. Sometimes the modifications will be quite simple, like adding a POKE statement to an Applesoft program or omitting portions of an Applesoft or machine language file.

If you happen to have a Replay II card, you can make a copy of the program at the title page and another copy with normal DOS 3.3 in the machine. You can then use the "compare" option on the replay II utility disk to compare the copy-protected DOS with normal DOS 3.3. This is often very helpful in making patches that allow the new copy to run properly.

Deprotecting Suicide

By Clay Harrell

Suicide
Piccadilly Software
89 Summit Avenue
Summit NJ 07901
\$29.95

Requirements:

Apple II or II+ with
old style F8 monitor ROM
A blank initialized DOS 3.3 disk
Suicide

Suicide is a rather gruesome little arcade game in which you attempt to save some creatures from killing themselves on the sidewalk. The game has a few twists too: you should not save mutant creatures, but let them fall to their deaths. Although this game has been around for quite awhile, the technique used to remove its protection is applicable to a great deal of other protected software.

Piccadilly uses a lot of disk protection to keep you from making a nibble copy of their disk. The primary protection is the use of half tracks. Upon booting the disk, the disk drive head seeks track 32 and then proceeds to read in the program alternating between whole and half tracks down to track 19. The program then JMP's to \$1000 to start the game.

The first step in removing the copy protection from a disk like this is to find what range of memory it occupies. The best way to do this is to use the monitor memory move command to fill most of memory with 00's, load in the game, and then to RESET into the monitor what has been loaded over by the program. To fill memory from \$800-\$9600 with 00's, from the monitor type

800:00 N 801 < 800.95FFM

After this is done, boot the game and RESET into the monitor and see what memory is used.

For Suicide, you should find that memory from \$800 to \$5FFF is used by the game. After booting the game and RESETTING into monitor, typing

7000 < 800.A00M

will clear the way for a slave disk boot. We can now BSAVE Suicide to a the DOS 3.3 slave disk.

If you try 1000G, you will find that the game starts up and everything is fine. When you RESET into the monitor you will see some screen garbage over text page one but fear not. This is the program loader and not part of the game. It does not need to be saved.

With Suicide, it is important to press RESET at the right time. Make sure you press RESET right when the red drive light goes out (when the initial title page is showing). Stopping the program here serves two purposes. First, when you save the game you will get the nice title page, and second, it stops the program before it initializes the lower pages of memory. So here's the method to deprotecting Suicide:

1) Boot suicide and, just after the drive motor stops (and when the initial title page is showing), reset into the monitor.

2) Next, move pages \$08 and \$09 where they will be safe during the boot of a slave disk

7000 < 800.9FFM

3) Boot a 48K slave disk with a short or null HELLO program

PR#6

4) Enter the monitor and move pages \$08 and \$09 back to their original locations

CALL -151

800 < 7000.71FFM

5) Next, enter some code which displays hi-res page 2 before the game starts

FF4:AD 50 C0 AD 54 C0 AD 57 C0 AD 52 C0

6) Install a patch to the above code at \$FF4

7FD:4C F4 0F

7) Finally, BSAVE the game to a DOS 3.3 disk by typing:

BSAVE SUICIDE,AS7FD,LS5803

Backup For Sabotage

By Clay Harrell

Sabotage

On-Line Systems
36575 Mudge Ranch Road
Coarsegold, CA 93614
\$24.95

Requirements:

Apple II, II+, or //e
A blank initialized DOS 3.3 disk
Sabotage from On-Line Systems

Sabotage was about the first game I ever bought for my Apple way back when DOS 3.3 had just been released. Even though it is a somewhat simple game, it held my attention for quite some time and even now I occasionally still play it.

The last time I tried to play Sabotage, I noticed that the original disk was having a hard time loading the hi-res title page. No doubt, the disk had seen its day and needed to be backed-up. I figured I could still save it before it was too late.

The protection used by On-Line was somewhat simple when this program was released (back in the good ol' days when Sierra On-Line was just On-Line Systems). The disk is

in a modified DOS 3.2 format which was another good reason for backing it up (this thing boots sloooooow!).

You can boot the disk and press ^{CTRL}C to interrupt the Hello program before it executes. You may now catalog the disk and examine the files if you wish.

The Hello program BLOAD's the banner picture and just BRUNs the game (file sabotage). What we want to do is BLOAD Sabotage, then check the locations it was loaded at and its length. To do this

BLOOD SABOTAGE

and then enter the monitor with

CALL -151

Then type

AA60.AA73

The last two bytes listed will be the location SABOTAGE was loaded at, and the first two bytes will be the length of the file, in backward order, of course. If you list memory at \$1D00, you will see that there are some memory moves to page \$01 and an RTS. If you type \$1D00G, the drive starts up and makes sure you're using the original disk. Obviously, we don't want this routine in our final production. Well, past that, the starting address is \$1D1F and the file continues up to \$5400.

So, to backup Sabotage we just BLOAD the file SABOTAGE, boot a 48K slave disk and BSAVE the file.

In cookbook fashion, here are the steps to deprotection:

1) Boot the SABOTAGE disk.

2) Press ^{CTRL}C immediately, and after the Hello program is loaded, your Apple will beep and you will be in APPLESOFT.

3) Type:

BLOAD SABOTAGE

4) Boot a 48K DOS 3.3 slave disk by typing:

PR #6

5) Save the program to your disk by typing:

BSAVE SABOTAGE,AS1D1F,LS36E2

Softkey For Millionaire

By Bill Wilson

Millionaire

Blue Chip Software
19537 Wells Drive
Tarzana, CA 91300
\$59.95

Requirements:

48K Apple II plus or equivalent
FID from DOS 3.3 System Master
A blank disk
Integer Card, modified F8 ROM or NMI card

Follow these simple steps to deprotect Millionaire:

1) Load in FID from the DOS 3.3 System Master

BLOOD FID

2) Enter the monitor

CALL-151

3) Make the following modifications to DOS

B925:18 60

B988:18 60

BE48:18

B8FB:29 00

4) Execute FID by typing

803G

5) Copy the following files from the original disk over to a normal DOS 3.3 disk

CHAIN

COMMON

INDUST

STOCKS

MESDATA

PLAY

DESCRIP

PLAYER

SAVE (If it's on your disk)

6) Boot the original Millionaire disk

PR#6

7) When the screen which asks if you wish to resume an old game appears, hit RESET (or use your NMI card) to enter the monitor.

8) Repeat step 3.

9) Place the DOS 3.3 disk which contains the Millionaire files (from step 5) into the drive and save the Millionaire Hello program

SAVE HELLO

(The above softkey was not verified by our staff.)

Deprotecting Time Is Money

By Rod Wideman

Time Is Money

Turning Point Software
11A Main Street
Watertown, MA 02172
\$100.00

Requirements:

48K Apple II plus or equivalent
COPYA from DOS 3.3 System Master
A blank disk
A sector editing program

Copy Time Is Money easily using the following instructions:

1) Make a copy of Time Is Money with COPYA from the DOS 3.3 System Master

RUN COPYA

DEAR AUTHOR:

Would you like to be published in **Hardcore COMPUTIST**? We would like to hear from you.

Hardcore COMPUTIST welcomes articles and submissions on a variety of subjects of interest to users of the Apple (or compatible) computers and would like to publish well-written material on the following:

- * Softkeys
- * Hardware Modifications
- * Advanced Playing Techniques
- * DOS modifications
- * Utilities
- * Product reviews
- * Adventure Tips
- * Original programs of interest
- * Do-it-yourself hardware projects
- * General interest articles
- * Bit-Copy Parameters

We prefer to see your submission on a DOS 3.3 disk using an Apple (or compatible) editing program. Please enclose a double-spaced hardcopy (paper) manuscript using a dot-matrix or letter-quality printer (or typewriter). Submissions will be mailed back if adequate return packaging is included.

Hardcore COMPUTIST pays on publication. Rate of payment depends on the amount of editing necessary and the length of the article. Payment ranges between \$10 for a short softkey, and \$50 per typeset page for a full-length article. We pay more for softkeys if the original commercial disk is enclosed for verification. We guarantee the disk's return.

Softkey Publishing buys all rights as well as one-time reprint rights (for upcoming BEST OF Hardcore) on general articles, and exclusive rights on programs. We may make alternate arrangements with individual authors, depending on the merit of the contribution.

At present we are not accepting fiction or poetry submissions, but **Hardcore COMPUTIST** may make an exception for an outstanding computer-related short story or poem.

For a copy of our WRITER'S GUIDE send a business-sized (20-cent) SASE (self-addressed, stamped envelope) to:

Hardcore COMPUTIST
WRITER'S GUIDE
P. O. BOX 44549-T
TACOMA WA 98444

Note: COPYA will probably stop when it tries to read track \$22. This is alright, as track \$22 is only used for the nibble count which will be eliminated with a sector edit.

2) Use a sector editor to make the following modifications to the copy of Time Is Money

Trk	Sector	Byte	From	To
\$02	\$0F	\$19	\$8D	\$60
\$02	\$0F	\$74	\$38	\$18
\$05	\$0F	\$19	\$8D	\$60
\$05	\$0F	\$74	\$38	\$18

3) Don't forget to write the modified sectors back out to the disk.

(The above softkey was not verified by our staff.)

Backing Up Type Attack By Jerry Caldwell

Type Attack
Sirius Software
10364 Rockingham Drive
Sacramento, CA 95827
\$39.95

Requirements:

- 48K Apple II plus or equivalent
- A nibble copy program
- A sector editor
- One blank disk

Type Attack from Sirius is one of those rare pieces of software that is both educational and fun to use. Unfortunately, like so many home and educational programs, Type Attack is copy protected to the extent that none of the bit copiers I own would make a working backup copy. Not wanting to hand my original Type Attack disk over to my young children, I set out to find a way to make a copy. Happily, I was successful. My report follows.

I found that on the original Type Attack disk, only tracks \$0-\$10 have any useful data on them, although track \$22 is used to verify the presence of the original program disk. I assumed that if I could find and circumvent the routine which checks track \$22, I would have a working bit copy of Type Attack.

The majority of the Type Attack disk is 4&4 encoded (rather than 6&2 encoded) and cannot be read by a normal sector editor. This makes it very difficult to find the location on the disk that the verification routine is called from. However, track \$0, sector \$0 can be read by any sector editor and I found that it was possible to make some modifications there to prevent the disk verification routine from being called. The changes I made on track \$0, sector \$0 modified the instruction at \$A68 so that the disk verification routine is bypassed and also changed the reset vector so that the disk will not do a total reboot if the RESET key is hit.

On an original copy of Type Attack the final instruction of the code on track \$0, sec-

tor \$0 reads

```
JMP $9F0
```

The modification which I made to the copy of Type Attack changed this code to

```
LDA #$4C
STA $A68
LDA #$6E
STA $A69
LDA #$0A
STA $A6A
LDA #$00
STA $A01
LDA #$40
STA $A0F
JMP $9F0
```

Luckily, there is enough free space on the sector to fit in the extra code.

Thus, the entire procedure for making a backup of Type Attack involves copying tracks \$0-\$10 with a bit copier followed by the sector edit to track \$0, sector \$0 of the copy. The procedure is detailed below.

Attack On Type Attack

1) Use a bit copy program to copy tracks \$0-\$10 of Type Attack. Set the address header parameters to AD DA DD (for COPY II+, E=AD, F=DA, 10=DD).

2) Use your sector editor to make the following changes to track \$0, sector \$0.

Byte	From	To	Byte	From	To
\$93	\$4C	\$A9	\$A1	\$00	\$09
\$94	\$F0	\$4C	\$A2	\$00	\$A9
\$95	\$09	\$8D	\$A3	\$00	\$00
\$96	\$00	\$68	\$A4	\$00	\$8D
\$97	\$00	\$0A	\$A5	\$00	\$01
\$98	\$00	\$A9	\$A6	\$00	\$0A
\$99	\$00	\$6E	\$A7	\$00	\$A9
\$9A	\$00	\$8D	\$A8	\$00	\$40
\$9B	\$00	\$69	\$A9	\$00	\$8D
\$9C	\$00	\$0A	\$AA	\$00	\$0F
\$9D	\$00	\$A9	\$AB	\$00	\$0A
\$9E	\$00	\$0A	\$AC	\$00	\$4C
\$9F	\$00	\$8D	\$AD	\$00	\$F0
\$A0	\$00	\$6A	\$AE	\$00	\$09

3) Write the sector back to the disk.

```
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
WORD ATTACK !!
```



Sometimes a BASIC program can be deceiving. Often, it isn't clear (in a print-out) exactly how many spaces follow a PRINT or DATA statement. Yet if you want your checksums to be correct and the program to function as the author intended, you must key in the program (spaces between quotes and after DATA statements, especially) exactly.

For this reason, Hardcore COMPUTIST will be printing BASIC listings with delta characters (^) in all the places where you must type a space. ALL other spaces are merely inserted into the program for easy reading.

When keying in DATA statements, DON'T type any spaces after the word DATA (even if there is one printed there). If you should find a delta character after the word DATA, type a space. Otherwise, DON'T! This is to ensure that your Checksoft generated checksums will match up with the ones we print for the program.

Zoom Grafix
Phoenix Software
64 Lake Zurich Drive
Lake Zurich, IL 60047
\$49.95

Requirements:

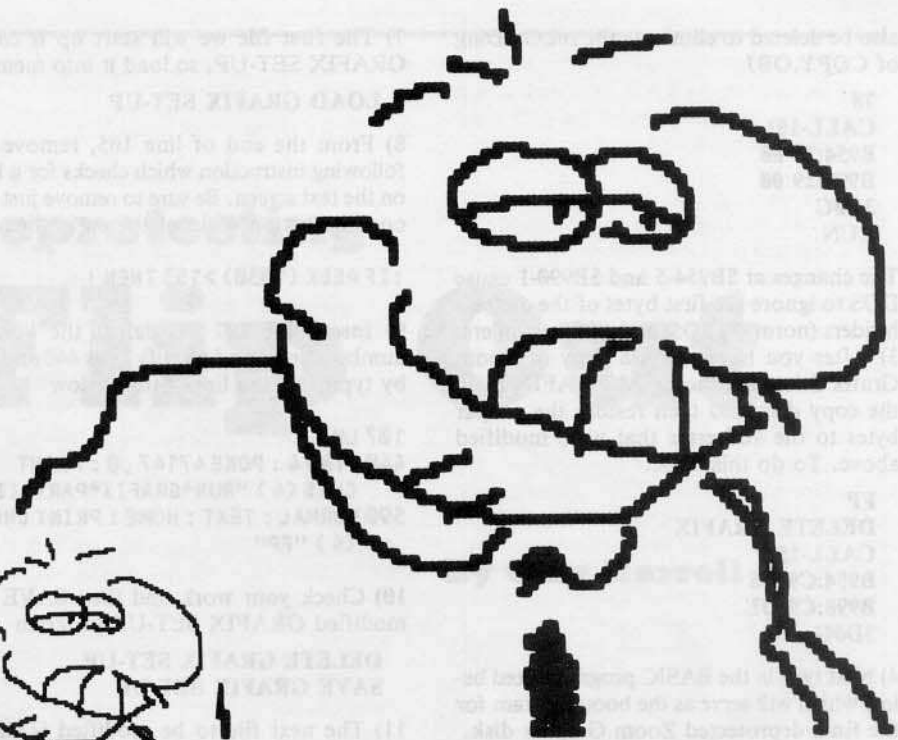
Apple II plus or equivalent
COPYA and FID from 3.3 System Master
Two blank disks

Zoom Grafix is a superb piece of programming and a delight to use, and has been a workhorse in my library for years. The recent acquisition of a hard disk drive prompted me to deprotect it.

After investigating the Zoom Grafix disk a bit, I was surprised to find that the bulk of the programs on the disk are written in Applesoft. Zoom Grafix is protected against standard DOS 3.3 copy programs by the use of non-standard address marks. This can be easily circumvented by making the appropriate POKEs into DOS before COPYA is run. However, the Zoom Grafix programs also check the disk for an illegal volume number in addition to the usual techniques of setting the autorun flag (\$D6) and messing up the DOS warmstart routine. Since the programs were written in Applesoft, it was fairly easy for me to remove these traps.

The two major steps to deprotecting Zoom Grafix involve:

- 1) Copying Zoom Grafix onto a disk which has normal DOS 3.3
- 2) Studying the listable Applesoft programs and removing the instructions which serve to copy protect the disk.



Deprotecting

Zoom Grafix

By Michael Decker

What To Do

This procedure is based upon the version of Zoom Grafix which is dated 9APR82. The procedure will also work on the earlier version, but the programs are numbered differently and you will need to find the appropriate lines to modify on your own.

We will use a modified COPYA to copy the original disk to a temporary disk which retains the abnormal DOS. Then FID will be used to transfer all the files from the temporary disk to a normal 3.3 disk or one that has been initialized with a "fast DOS". We also have to write a new boot program for the final deprotected Zoom Grafix and remove copy protection traps that some of the Applesoft programs contain.

The overall procedure I use may seem rather roundabout, but there are no special requirements such as a non-autostart ROM or NMI card. All owners of Zoom Grafix

"Zoom Grafix...a superb piece of programming and a delight to use...has been a workhorse in my library for years."

should be able to perform this softkey. Let's get started.

- 1) Boot with a normal DOS 3.3 disk and then RUN the COPYA program

**PR#6
RUN COPYA**

- 2) After COPYA is loaded and running, we need to halt it so that some modifications to DOS can be made. Line 70 of COPYA will

also be deleted to eliminate the reLOADing of COPY.OBJ

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

The changes at \$B954-5 and \$B990-1 cause DOS to ignore the first bytes of the address headers (normally \$D5) and address trailers. 3) After you have made a copy of Zoom Grafix delete the file called GRAFIX from the copy disk and then restore the proper bytes to the addresses that were modified above. To do this, type:

```
FP
DELETE GRAFIX
CALL-151
B954:C9 D5
B990:C9 DE
3D0G
```

4) Next type in the BASIC program listed below which will serve as the boot program for the final deprotected Zoom Graphix disk.

```
10 TEXT : HOME : D$ = CHR$ (13) + CHR$
(4)
20 PRINT D$ "MAXFILES^1" : PRINT D$
"BLOAD^GRAFIX.OBJ"
30 HOME : PRINT "1)^^CONFIGURE^
ZOOM^GRAFIX" : PRINT : PRINT
"2)^^RUN^ZOOM^GRAFIX" : PRINT :
PRINT "^^^ENTER^YOUR^CHOICE^"
; : GET A$ : PRINT A$
40 IF VAL (A$) < 1 OR VAL (A$) > 2 THEN
GOTO 30
50 A = VAL (A$) : ON A GOTO 60 , 80
60 POKE 103 , 1 : POKE 104 , 96 : POKE
24576 , 0
70 PRINT CHR$ (4) "RUN^GRAFIX^SET-
UP" : END
80 POKE 103 , 1 : POKE 104 , 96 : POKE
24576 , 0
90 PRINT CHR$ (4) "RUN^
GRAFIX^PART^II"
```

5) Now, initialize a blank disk with this program in memory. If you like, a "fast DOS" can be used.

INIT ZOOM LOADER

6) We will use FID from the DOS 3.3 System Master to transfer the files from the temporary copy of Zoom Grafix to the disk we just initialized. Get FID up and running and use the wildcard character ("=") to transfer all the files

BRUN FID

You are now done with the temporary disk. When you CATALOG the final copy disk, you'll notice that the program files all seem to be 000 sectors long. This is wrong and we will correct their sizes by loading, deleting, and resaving them as we deprotect them. If you have an Applesoft program editor, like GPLE, load it in now because it will make your job much easier.

7) The first file we will start up is called GRAFIX SET-UP, so load it into memory

LOAD GRAFIX SET-UP

8) From the end of line 105, remove the following instruction which checks for a logo on the text screen. Be sure to remove just this one instruction and not the entire line

```
: IF PEEK (1030) > 153 THEN !
```

9) Insert line 107 (to defeat the volume number check) and modify lines 440 and 590 by typing in the lines listed below

```
107 LM = 0
440 VTAB 4 : POKE 47147 , 0 : PRINT
CHR$ (4) "RUN^GRAFIX^PART^II"
590 NORMAL : TEXT : HOME : PRINT CHR$
(4) "FP"
```

10) Check your work and then SAVE the modified GRAFIX SET-UP program

DELETE GRAFIX SET-UP SAVE GRAFIX SET-UP

11) The next file to be modified is called GRAFIX PART II, so LOAD it in. You may have to CATALOG the disk and trace over the file name if you can't type the left bracket ([) from your keyboard

LOAD GRAFIX PART II

GRAFIX PART II is the main program which contains a number of excellent routines.

12) Make the following modification to GRAFIX PART II before resaving it: Enter a line to kill the volume number check

```
195 LM = 0
```

Remove from the beginning of lines 990 and 1750 a command which will trash the DOS warm-start routine

```
: POKE -25150, 18:
```

Replace line 880 with the following line

```
880 NORMAL:TEXT:HOME:PRINT
CHR$(4);"FP"
```

13) Check your work and then resave the program. The brackets in the file name will be replaced with I's. Again, you may have to CATALOG and trace over the file name

DELETE GRAFIX PART II SAVE GRAFIX PART II

14) The remainder of the files do not need to be modified, but we will load, delete and resave them all so that a CATALOG will show the proper file sizes

```
BLOAD GRAFIX.INFO
DELETE GRAFIX.INFO
BSAVE GRAFIX.INFO, A$800, L$4D5
```

The next binary file overwrites its own buffer at the default maxfiles of 3.

```
MAXFILES 1
```

```
BLOAD GRAFIX.OBJ
DELETE GRAFIX.OBJ
BSAVE GRAFIX.OBJ, A$9000, L$94D
FP
```

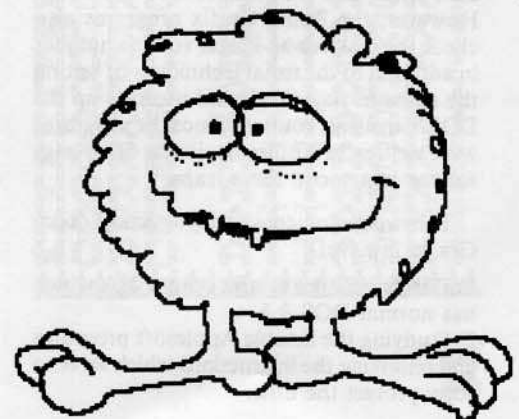
Final Comments

You now have a fully functioning, deprotected ZOOM GRAFIX which you can list, examine, and modify. GRAFIX PART II is the main program; it pokes in two short machine language routines from 926-935 and 936-973 (the latter switches HI-RES screens.) GRAFIX.INFO contains set-up parameters as used by GRAFIX SET-UP. These parameters are ultimately passed to GRAFIX.OBJ. The latter performs the actual printing tasks, as well as screen flipping and other duties. The deleted program GRAFIX is a loader which is so full of traps it's best replaced by the program which we called ZOOM LOADER. Note that the loader reset the Applesoft start-of-program pointers to \$6000, so that GRAFIX SET-UP and GRAFIX PART II load above HI-RES page 2 (from 24577 to 34016). We closed our exit routines with FP so that these pointers and maxfiles would be normalized on exit from the program; if you reset out of either program, remember that these are left abnormal.

If your deprotected program seems to run but won't actually print, it is probably because you haven't properly run the set-up routines. As I mentioned before, those of you who have a different version than mine (9APR82) will have to list the programs and find the appropriate changes to make.

A final enhancement: ZOOM GRAFIX has an undocumented feature. If you enter a question mark (?) when the initial screen is displayed, ("MAKE SURE PRINTER IS READY THEN PRESS RETURN TO GO ON"), a date will appear on the screen, presumably the date of manufacture. You can change this to whatever date you like with the following method:

```
MAXFILES 1
BLOAD GRAFIX.OBJ
POKE 36926, first number (month)
POKE 36927, second number (day)
POKE 36928, third number (last two digits
of year)
BSAVE GRAFIX.OBJ, A$9000, L$94D
FP
```



Requirements:

48K Apple II or II+ with old style

F8 monitor ROM

One blank initialized DOS 3.3 disk

Flip Out from Sirius Software

Flip Out is a hi-res strategy game from Sirius Software (R.I.P.) which requires you to send all ten of your marbles through the playing course before your opponent does. Each player starts with ten of his opponent's marbles and then takes turns dropping these marbles into the Flip Out playing field (of which there are many variations). Your first goal is to trap your opponent's marbles in a spot where they will be difficult to recover. Each marble dropped may cause a chain reaction, so some strategy is required. After the players have dropped all ten of their opponent's marbles, they begin to drop their own marbles through the course. This continues until one of players wins by getting all his marbles through the course.

Who Stole the ROMS?

Flip Out is well done and challenging! But after I bought it and was done playing a game, I hit the RESET key and did not see the usual monitor prompt (I have an old style F8 monitor ROM on the motherboard). My computer rebooted as if I had a new style F8 monitor ROM! This intrigued me into investigating this strange phenomenon.

What I discovered was that the main program (not the boot code) will copy an image of the new style F8 monitor ROM into a slot 0 RAM card if one is found. Of course, I had a RAM card and they had it turned on instead of the motherboard ROMs. It is easy to understand what they are doing if you just remember what memory the RAM card occupies.

The RAM card occupies memory from \$D000 to \$FFFF. This may seem strange since the motherboard ROMs (APPLESOFT and the monitor) also occupy \$D000 to \$FFFF. However, there are a set of soft switches that can turn on motherboard ROMs or turn on the RAM card. An example of this occurs when you boot your 48K DOS 3.3 system master and it loads INTEGER basic into the RAM card. Now you have two languages available that occupy the same logical memory space, \$D000 to \$FFFF. You can switch between the two with INT and FP. When the INT command is typed in, the soft switches are thrown so that the language card is read enabled. Likewise, the FP command read enables the motherboard ROMs.

Deprotecting Flip Out

By Clay Harrell

If you read your RAM card manual you can see that the softswitch at memory location \$C080 (assuming your RAM card is in slot zero) will allow you to look at your RAM card's memory. But have you ever tried typing \$C080 from the monitor? It will lock your Apple up requiring you to power off and then back on again to recover.

Recovering the ROMS

This phenomenon results because you have switched to the RAM card's memory \$D000-\$FFFF and turned off the mother-

board's ROMs. When you do this, the computer loses control since there is no longer a monitor ROM available from \$F800-\$FFFF to overlook your Apple's operations! You cannot recover from this condition, even with a RESET.

To get around this problem you must first type

```
C081 C081 N F800 < F800.FFFFFM
```

from the monitor. This reads the motherboard's ROM, but allows you to write to bank 2 of the RAM card. The memory move, F800 < F800.FFFFFM, moves the F8



motherboard ROM into \$F800-FFFF in bank 2 of the RAM card! Now you may type C080 to turn on the RAM card and look at its memory, since a copy of the \$F8 monitor ROM is in the RAM card from \$F800 to \$FFFF.

This is what Sirius and other software publishers will sometimes do to prevent people from utilizing their RAM cards for deprotection purposes. With a RAM card in slot 0, no matter what \$F8 ROM you have in the motherboard, your Apple will only look at the new style \$F8 ROM image in your RAM card. Thus, the Apple clears memory and reboots when RESET is pushed! This is an easy problem to fix, now that we have identified it. Just take your RAM card out of your computer and you may RESET into the monitor as usual. As demonstrated in the Softkey for Sensible Speller, you can sometimes get away with moving your RAM card to a slot other than 0 where the program does not expect to find a RAM card.

We may now move onward in the quest for deprotection of Flip Out.

Flip Out is a single load program. To deprotect single load programs, there are 3 basic things which we need to determine:

- 1) What memory is used by the program.
- 2) The starting address of the program.
- 3) How to get the memory saved to a normal DOS 3.3 disk and reloaded back into memory in the proper place(s).

Keep these three items in mind as we snoop through Flip Out or any other single load game.

Sirius has changed its protection schemes a lot in the last few years. The height of their protection mania was demonstrated in games like "Bandits" and "Fly-wars". The problem with hi-tech protection is that a program might not boot on a Rana drive, or on a //e or some other flavor of Apple. Sophisticated protection schemes are also costly and drive up the retail price of software.

In light of this, Sirius chose a much simpler, but still effective, copy protection scheme for Flip Out. Just try and copy it with Nibbles Away or another bit copier and you'll see just how effective it is!

A Bit of Boot Tracing

Keeping in mind the three things we must figure out to deprotect a single load game, the first thing I generally do is to find what memory is required to run the program. To do this we can flip through memory (remember that shape tables, etc. don't disassemble into meaningful code), or we can trace the boot and see where the program gets loaded to. I prefer to trace the boot when it is fairly simple, which it is on Flip Out.

So, boot up normal DOS 3.3 (so we can save a piece of code for later examination, if you wish) and enter the monitor with CALL-151. Now we must copy the code in

the disk controller ROM down to RAM so we can modify it to our liking. Do this by typing

8600 < C600.C6FFM

from the monitor. Now we have the disk controller ROM code where we can modify it, and start to trace the boot.

If you did not know already, the disk controller ROM reads track zero, sector zero into memory from \$800 to \$8FF. Then it JMPs to \$801 and starts executing the code (which continues by loading in a little more code, which then loads in more code, which then.....well, you get the idea).

At the end of the code at \$8600 we see a JMP \$0801. We must change this to JMP \$FF59, which will exit us in the monitor after it is done loading track zero, sector zero into \$800 to \$8FF. Put Flip Out in drive one and type

86F9:59 FF N 8600G

The drive will recalibrate and, a second later, beep into the monitor just like we told it. To turn off the drive motor, type

C0E8

from the monitor prompt. If you want to, you can save this hunk of code to your normal DOS 3.3 disk with BSAVE BOOT0,A\$800,L\$100 (since this process did not disturb DOS which lives from \$9600 to \$BFFF).

Now type 801L to flip through the code just loaded. This code below is what you should find

```

801- A5 2B LDA $2B
803- AA TAX
804- 85 FB STA $FB
806- 4A LSR
807- 4A LSR
808- 4A LSR
809- 4A LSR
80A- 09 C0 ORA #$C0
80C- 8D 00 30 STA $3000
80F- A0 00 LDY #$00 -----
811- 84 00 STY $00
813- A9 D0 LDA #$D0
815- 85 01 STA $01 Put the
817- A2 30 LDX #$30 code from
819- AD 81 C0 LDA $C081 ROM into
81C- AD 81 C0 LDA $C081 the slot
81F- B1 00 LDA ($00),Y zero RAM
821- 91 00 STA ($00),Y card.
823- C8 INY
824- D0 F9 BNE $081F
826- E6 01 INC $01
828- CA DEX
829- D0 F4 BNE $081F -----
82B- A6 FB LDX $FB Load code
82D- 84 F7 STY $F7 into RAM
82F- A9 04 LDA #$04 starting
831- 85 F8 STA $F8 at page
833- 85 FA STA $FA $04
835- BD 8C C0 LDA $C08C,X -----
838- 10 FB BPL $0835
83A- C9 AD CMP #$AD Look for
83C- D0 F7 BNE $0835 a data
83E- BD 8C C0 LDA $C08C,X field
841- 10 FB BPL $083E epilogue
843- C9 DA CMP #$DA of AD,
845- D0 F3 BNE $083A DA, DD
847- BD 8C C0 LDA $C08C,X
84A- 10 FB BPL $0847
84C- C9 DA CMP #$DD
84E- D0 EA BNE $083A -----

```

88A- 4C 29 04 JMP \$0429 JMP BOOT2

This code loads in the final loader (boot2) over the text screen memory (\$400 to \$7FF) and JMPs to \$429. Now we need to examine boot2 (the game loader) to see where it actually loads in the game.

You might notice that this is slightly difficult since boot2 gets loaded over the text page, and when we hit RESET this memory pretty much hits the bit-bucket. But boot1 (the code which we are now looking at in \$801-\$88C) can be changed to load boot2 somewhere else and gracefully RESET into monitor. To do this we can change the load byte from page \$04 to page \$14, and change the JMP \$429 to jump into the monitor. Then we can examine the boot2 loader. To do this enter

830:14 N 88B:59 FF

from the monitor.

The next thing we must change is the disk controller ROM code at \$8600. We need it to execute, but not write over, the modified code at \$801. To do this, we can tell it to load track zero sector zero at \$6000 (instead of \$800) and jump to our modified code at \$801. Of course our code will load boot2 into \$1400 so we can look at it. Are you confused yet? Go back if you are, and don't come back until you understand what is going on.

OK, put Flip Out in drive one and type

8659:60 N 86F9:01 08 N 8600G

The drive will recalibrate and boot zero will read track zero, sector zero into \$6000 (thus, not overwriting our code at \$801). It will then jump to \$801 (boot1) and load boot2 into \$1400 to \$17FF. Unfortunately, it will keep reading boot2 into \$1400 to \$17FF because we haven't changed enough code. So, after a few seconds, hit RESET.

Now if you want to, you can put your normal DOS 3.3 disk in a drive and save boot2 with BSAVE BOOT2,A\$1400,L\$400.

Next type 1429L and examine boot2. You will notice that memory from \$800 to \$BFF gets wiped clean and that a reset error routine gets moved to \$8F00-\$8F80. This is a good indication that Flip-Out lives from \$C00 to \$8F80! I'll let you sort through the boot2 code to find out for sure, or you can take my word for it.

The last tidbit of information that the boot2 loader reveals is the starting location of Flip Out. Look at the code at \$17CC to \$17E4 and you'll see how it wipes out memory from \$800 to \$BFF and then JMPs to \$7800 to start the game.

Now we have filled requirements one and two. All that is left is to save the memory from \$C00 to \$8F80 on a normal DOS 3.3 disk. This is easy since a 48K slave disk does not destroy memory from \$900 to \$96FF. So just boot Flip Out and, when the drive stops, RESET into the monitor. Now boot your 48K slave disk and save Flip Out to disk!

The Steps

In cook-book fashion, here are the steps necessary to get a BRUNable version of Flip Out.

0) Turn your Apple *OFF* and remove your RAM card

1) Boot Flip Out

PR#6

2) After the drive stops and Flip Out is loaded into memory, hit *RESET* to enter the monitor.

3) Boot a 48K slave disk.

PR#6

4) Enter the monitor by typing

CALL-151

5) So that the program will execute when BRUN, enter some code which JMP's to \$7800 to start up the game

BFD:4C 00 78

6) BSAVE Flip Out by typing

A964:FF

BSAVE FLIP-OUT,ASBFD,LS8383

If you want the title page displayed, you will also have to perform the following steps:

1) Reboot the Flip Out disk and *RESET* into the monitor when you see the title page.

2) Boot your 48K slave disk and BSAVE the picture by typing

PR#6

BSAVE PIC,AS2000,LS1FFB

3) BLOAD the Flip Out file

BLOAD FLIP-OUT

4) BLOAD the picture file

BLOAD PIC,AS2000

5) Enter the monitor with

CALL-151

6) Enter this code which will display hi-res page 1 and wait for a key to be pressed before it JMP's to the start of the game

BE0:AD 10 C0 AD 50 C0 AD 54

BE8:C0 AD 57 C0 AD 52 C0 AD

BF0:00 C0 10 FB 4C 00 78

7) BSAVE the file by typing

A964:FF

BSAVE FLIP-OUT,ASBE0,LS83A0



Moving Soon?

Let us know your new address at least 30 days in advance so you won't miss a single issue of Hardcore COMPUTIST.

Fill out the form below, paste your present mailing label in the space provided, and send it to us.

That's all there is to it!

My new address is:

Address _____

City _____ St _____ Zip _____

Paste present mailing label here or fill in old address:

Name _____

Address _____

City _____ St _____ Zip _____

Country _____

Send to: Hardcore COMPUTIST, Subscription Dept., P.O. Box 44549-T, Tacoma, WA 98444

ADVENTURE TIPS

ADVENTURE TIPS

Cranston Manor Sierra On-Line

Trying to get into the vault? Money talks. Water and computers don't mix. Look for the route to the tower in the library's reference section.

Enchanter Infocom, Inc.

Read each sign on the Long Road. You can get the rezrov spell from the witch, but not much more. The Adventurer will like you only with the help of the vaxum spell. Look for it in the bedroom.

Kabul Spy Sirius Software

Aren't there always old newspapers under the bed? The little boy can give you a clue to get to the bar, for a price. "Go to ----" to get into the bar. The priest and his church are suspiciously well off. You might be in danger of being waylaid on the road.

Colossal Caves Adventure International

Don't know how to catch the bird? Ask for "help". Drop the nugget and the stairs will magically reappear. Only dwarfs can read magazines. That plant looks big enough to climb.

The Witness Infocom, Inc.

A good detective is alert and aware of his surroundings. The butler may know where the man of the house is. Everyone knows a little something about everyone else. Ask around. What time is it?

Pirate Adventure Adventure International

If you're walking on window ledges, it's best to not wear heels. There's some good reading in the alcove. Rum is good as gold to a pirate. You'll need to light up that crack before you go in.

Exodus: Ultima III Origin Systems, Inc.

Make sure to wear your armor and carry a weapon, or your adventure will be shortlived. Try a direct attack to acquire a boat. Enter the whirlpool to reach Ambrosia. Be sure to have plenty of keys handy when you want to leave the continent.

Adventureland Adventure International

* The ax is sharp enough to cut a "cypress tree".
* After you say "Bunyon", the ax will disappear. Then find "Paul's place".
* Contributed by The Wizard

Softkey For Lion's Share

By Jan Eugenides



Requirements:

Apple II+ or equivalent
One blank disk
Super IOB from Issue No. 9

This is a softkey for the adventure game, "The Lion's Share", by Davka Corporation.

First, for those of you who just want to backup the game, I will provide a softkey. Second, for those who are interested, I will also explain how I figured it out.

The Softkey

Here's all you do:

1) Boot your system master and clear any program in memory

FP

2) Enter the monitor

CALL -151

3) Make some modifications to DOS

BA69:60

BA6A < BA69.BA91M

BCDF:60

BCE0 < BCDF.BCF8M

4) Put in a blank disk to put this modified DOS on to

INIT A

5) Turn the disk over and put the DOS on the other side, too

INIT A

Note: You will have to cut a notch in the side of the disk to allow you to format this side.

6) Reboot your system master

PR#6

7) Type in the Super IOB controller at the end of this article and save it.

8) Put side one of "The Lion's Share" in the drive and start the drive with the drive door open

PR#6

9) Before closing the door, press the break key

CTRL C

10) Close the door, wait for the break message and then clear the program in memory.

FP

11) Enter the monitor

CALL- 151

12) Move the RWTS to a safe place for booting

2000 < B800.BFFFM

13) Put your Super IOB disk (or some other disk with a very short greeting program) in the drive and boot it.

6 CTRL P

14) Save the RWTS

BSAVE RWTS.LIONS,AS2000,LS800

15) Load Super IOB and install the controller at the end of this article and then execute Super IOB

RUN

16) Copy both sides of the original Lion's Share disk to both sides of the disks you formatted at the start of the procedure.

17) You're done! Boot your copy and enjoy!

Beneath The Softkey

"The Lion's Share" is a pretty good adventure, with some nice graphics. I enjoyed it, but even more, I enjoyed figuring out the protection scheme. It was much more of a challenge than getting the sword out of the snake pit!

I am totally against piracy, but I am totally for learning and I learned a lot about DOS, and about my Apple, doing this crack.

Here's How I Did It

When I booted the game, I noticed a prompt at the bottom of the screen. This usually means that some form of DOS is being used. Going on this assumption, I decided to have a look at tracks 0-2 of the disk, to see if I could ferret out the system.

No dice. I couldn't read the disk at all, nor would it CATALOG. Using a program called "Diskview 1.1A," I managed to peek at the raw nibble dump, and it was definitely nonstandard. So, I resorted to a little trickery.

First I removed the top 8 RAM chips from my machine. These are the ones nearest to the left rear of the Apple, inside the white box labeled "RAM." (Scary, isn't it?) Then I booted my system master. (It has to be the master, and not a slave disk.) This had the effect of loading DOS 16K lower in memory than usual. (The higher memory was in my left hand!)

You see, the master disk checks to see how much memory is available and loads DOS in at the top of this available memory. A slave disk, on the other hand, always loads DOS into the same location, namely the location it was in when the slave was made.

Can You See What's Next?

I then initialized a slave diskette, using my now 32K Apple. I'll call this the 32K slave. Next, I replaced the RAM chips, and booted "The Lion's Share."

Pressing RESET causes the disk to re-boot, since the reset vector has been changed, so I took advantage of this fact. I now removed the game disk, and put in my 32K slave. Pressing RESET caused the disk to boot, and regular old DOS 3.3 was loaded in at its 32K location. Because of this, the Davka DOS was still intact in the upper memory! Now it was a simple matter to save this Davka DOS with an address of \$9D00 and a length of \$22FF.

I now had Davka DOS on my disk as a binary file. You may be able to use this method on other protected disks. Just keep the 32K slave around for future use.

The next step was to boot up the system master again, thus placing DOS 3.3 back in

its normal location at \$9D00. I now BLOADED Davka DOS at \$2D00. (It can be anywhere, really, as long as it doesn't overwrite DOS 3.3)

Using the monitor verify command I made a print-out of all the locations which were different between the two DOSs. This is a nifty command which compares two ranges in memory and tells you if they are the same or not, and if not, what the differences are. Check the Apple reference manual for more information.

In this case, the command was 2D00<9D00.BFFFV. (Turn on your printer first with a PR#1, and set it to skip the perforations if you can.) This gave me a complete list of all the changes!

With my trusty copy of "Beneath Apple DOS" in hand, I proceeded to analyze the changes. I saw that the SAVE and INIT command word spelling had been altered but the rest of the commands were the same.

Most importantly, I discovered that both the read and write routines had been changed, as well as their translate tables. No wonder I couldn't read the disk; it was scrambled!

Then it was no problem. I just patched the DOS 3.3 read routine to match that used by Davka. Sure enough, I could now CATALOG the game disk. Of course, the DOS 3.3 write routine was still standard, so all I had to do was FID all the files over to a normal disk, right? Not quite, but close!

Once I had copied all the files to a normal disk, I booted it and tried to run the game. It worked perfectly until I tried the SAVE GAME option, and then it bombed.

The Apple obligingly informed me that it had stopped at \$BA78. Hmm. Isn't that unused space in DOS? Yep, it sure is. But Davka put a routine there which changed the address and data markers back to standard, so that their game could read and write to standard disks. (Pretty sneaky, huh?)

I replaced this whole section with 60's (RTS), and also the section at \$BCDF, which was part of the same routine.

After I had initially deprotected The Lion's Share, I discovered something which made the overall softkey procedure a little easier. What I found was that The Lion's Share boot program would break (stop execution) if ^{CTRL}C was typed as the disk booted. After the break occurred, typing

FP
CALL -151
2000<B800.BFFFM

allowed me to enter the monitor and

move The Lion's Share RWTS to a safe location. After booting with a slave disk, I just BSAVED the RWTS.

To automate the copy process a little, I modified the SWAP Controller of Super IOB so that it would utilize the Lion's Share RWTS. All these changes were incorporated into the softkey procedure which I detailed at the beginning of this article.

Ta Daaa!

I have played the whole game through on my copy, and it works perfectly. It works exactly like the original, only now I don't have to worry about damaging it because I can make backups. Also, I can peek at the files if I feel too frustrated, although this game isn't that hard. Not like Zork III, anyway! I can also check out the program to see how it works and learn some more. That's what it's all about, right?

Lion's Share Controller

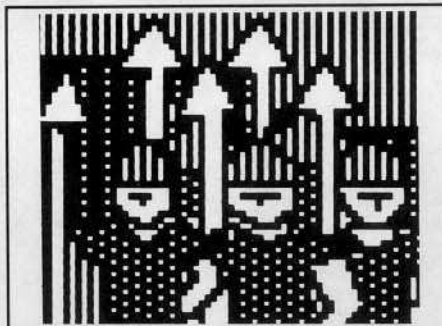
```

1000 REM SWAP CONTROLLER
      ( LIONS SHARE )
1010 TK=3 : ST=0 : LT=35 : CD=WR
1020 T1=TK : GOSUB 490 : GOSUB 360 :
      ONERR GOTO 550
1030 GOSUB 430 : GOSUB 100 : ST=ST+1
      : IF ST < DOS THEN 1030
1040 IF BF THEN 1060
1050 ST=0 : TK=TK+1 : IF TK < LT
      THEN 1030
1060 GOSUB 490 : TK=T1 : ST=0 : GOSUB
      360
1070 GOSUB 430 : GOSUB 100 : ST=ST+1
      : IF ST < DOS THEN 1070
1080 ST=0 : TK=TK+1 : IF BF=0 AND
      TK < LT THEN 1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "REMEMBER TO A
      COPY BOTH SIDES" : END
10010 PRINT CHR$(4) "BLOAD A
      RWTS.LIONS,AS1900"

```

Controller Checksums

1000 - \$356B	1060 - \$20C0
1010 - \$3565	1070 - \$28C5
1020 - \$6170	1080 - \$6CA2
1030 - \$7771	1090 - \$9DCA
1040 - \$6342	1100 - \$CED6
1050 - \$ABA3	10010 - \$0A76



Most Wanted List

If you have been trying to backup a program, and have only ended up pulling your hair out as a result of the ordeal, let us know about it.

We have received softkeys for a number of programs previously in our list and these will be published as soon as each has been evaluated and edited by our staff.

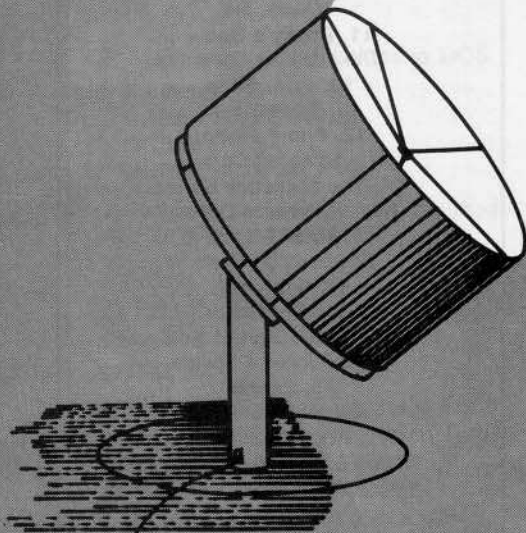
**Hardcore COMPUTIST
Wanted List
P.O. Box 44549K
Tacoma, WA 98444**

If you know how to de-protect, unlock or modify any of the programs below, we encourage you to help other Hardcore COMPUTIST readers and earn some extra money at the same time. Send the information to us in article form on a DOS 3.3 diskette.

1. **Apple Business Graphics**
Apple Computer
2. **Flight Simulator II**
Sub Logic
3. **DB Master 4.0**
Stoneware, Inc.
4. **Bookends**
Sensible Software
5. **Crossword MAGIC**
L & S Computerware
6. **Visiblend**
Micro Lab
7. **BPI General Ledger**
Apple Computer
8. **Dollars And Sense**
Monogram
9. **Word Juggler**
Quark, Inc.
10. **Catalyst**
Quark, Inc.
11. **Rocky's Boots**
The Learning Company
12. **Print Shop**
Broderbund
13. **Prime Plotter**
Primesoft Corp.
14. **The Statistics Series**
Human Systems Dynamics
15. **MASTER TYPE**
Scarborough Systems
16. **Sargon III**
Hayden
17. **Dow Jones Market Analyzer**
Dow Jones Software
18. **Zardax**
Computer Solutions
19. **Stickybear Series**
Xerox Education Publications

Psychedelic Symphony

By Ray Darrah



Requirements:
48K Apple][with
cassette jacks

While enjoying a laser show choreographed by Steven Chandler, I had a brainstorm. What if I were to use the cassette input jack of an Apple][to convert sound into a dazzling geometric display? A few short days later and presto, Psychedelic Symphony was born.

What Exactly Does It Do?

The work horse of the Psychedelic Symphony pair (Symphony.OBJ) analyzes the frequency at the cassette jack and then converts the resulting numbers into a lo-res rectangle. It does this only once. When called over and over again, the square placed on the screen will contort in rhythm to the sound being played through the jack. This produces the effect of animation similar to a laser show.

Symphony.OBJ can animate up to 28 squares. Each square has its own specifications. Subjects covered in the specifications include coordinates of the square, maximum size and color parameters.

Specifications are passed to Symphony.OBJ via memory locations. Up to 256 bytes may define the squares to be animated. Location \$E0 and \$E1 must point to the first byte of this string of parameters.

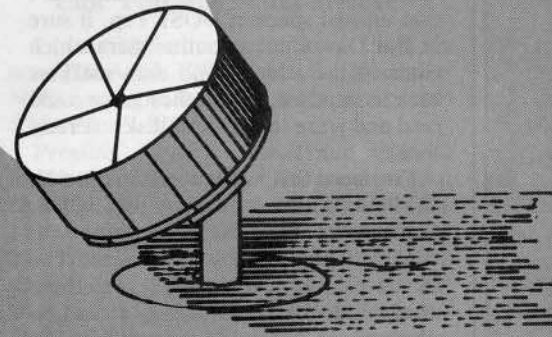
The string of parameters is arranged as follows:

Byte(s)	Function
0	This byte holds the number of squares to place on the screen after each call.
1 - 9	Definition of square number 1.
10 - 18	Definition of square number 2.
⋮	⋮
⋮	⋮
(n-1)*9 + 1 through (n-1)*9 + 9	Definition of square number n.

The Definitions of the individual squares each occupy nine (9) bytes. Only seven (7) of them are used by Symphony.OBJ to define the square. The other two bytes (bytes four (4) and seven (7)) are used by Symphony.OBJ as workspace.

Here is how the individual definitions are arranged:

Byte(s)	Name	Function
0	Sample	If bit 7 is set, then a new sample will be taken prior to plotting the square.
1	Stdcolor	If bit 7 is set, then this square is a steady color and the next byte applies. Otherwise a random color will be chosen each time the square is updated and byte two (2) does NOT apply.
2	Color	If bit 7 of byte one (1) is set, then this is the color to use for this square.
3	Dividy	The divisor that the cassette sample will be divided by to calculate the height of the square.
5	Centy	The y coordinate of the center of the square.
6	Dividx	The divisor that the cassette sample will be divided by to calculate the width of the square.
8	Centx	The x coordinate of the center of the square.



The BASIC driver program (Psychedelic) provides an easy way of defining the squares (or rectangles). It also allows you to load or save your favorite definitions of squares.

The program interfaces to Symphony.OBJ through a machine language routine that 1. Calls Symphony.OBJ, 2. Waits for a while, 3. Tests the keyboard for a key, 4. Goes to step one if none is pressed (1). Otherwise, it returns.

: Convert Your Favorite Music Into A Dazzling Geometric Display

Typing It In

First of all, type in the hexdump at the end of this article and save it.

BSAVE SYMPHONY.OBJ,AS300,LSC4

Next, type in the BASIC program at the end of this article and save it.

SAVE PSYCHEDELIC

It is important not to **R**UN the program before saving it. This is because the first few lines relocate the BASIC program.

Using Psychedelic

The main menu of Psychedelic has five options:

```

PSYCHEDELIC SYMPHONY FORMATTER
1) EDIT THE FORMAT
2) VIEW THE FORMAT IN ACTION
3) LOAD A FORMAT
4) SAVE THE FORMAT
WHICH ?
    
```

Pressing a number from one (1) through four (4) will invoke the corresponding routine. These options are explained below. The fifth option, which exits the program, is not displayed on the screen. To exit press the ESCape key.

Edit The Format

In Psychedelic, the combined definitions of squares is referred to as the "format."

If you should press 1, the screen will turn to GGraphics and all the maximum sizes of the squares in the format will be displayed. The square you are editing will blink. At this time, a number of commands are available to you. They are:

I J K M - Moves the current square in the usual ESCape directions.

CTRL I - Shrinks the maximum size of the square vertically.

CTRL J - Shrinks the maximum size of the square horizontally.

CTRL M - Enlarges the maximum size of the square vertically.

CTRL K - Enlarges the maximum size of the

square horizontally.

CTRL U - Also known as the right arrow key, drops the current square where it lies and grab the NEXT one.

CTRL H - Also known as the left arrow key, drops the current square where it lies and grab the LAST one.

CTRL D - Deletes the square you're working with from the list of definitions.

CTRL N - Adds another square to the table of definitions exactly like the last one formed (same size and position).

ESC - Exits the edit mode with all the squares placed exactly where they are being displayed.

CTRL S - Allows the editing of the special features of the current square which are:

```

STATUS
NEW SAMPLE=>Y
STEADY COLOR=>Y
COLOR=>7
    
```

New sample is used to link squares together. When it is set to No, this square will use the same set of samples as the square before it. This means that if it were the same maximum size, the squares would perform identically.

When steady color is set to Y, the square will always be drawn in the same color (the color specified by color). If steady color is set to N, then every time the square is drawn, it will appear in a random color.

Color is only valid if steady color is set to Y. In this case, color should be set to the lores (0-15) "COLOR=" number corresponding to the color you wish.

View The Format In Action

After pressing "2" you will be prompted for a delay constant between 1 and 255. This is the amount of time that will expire between each call of Symphony.OBJ. The greater the number, the slower the squares will jump. Smaller numbers produce faster more nervous displays. When you are through viewing the format in action, press any key. You must press a key while the squares are being animated. If you press a key with no sound being played into the cassette jack, nothing will happen.

Load A Format

This option can be used to get a CATA-

LOG of the disk in the drive. After you press "3", the directory of the current disk is displayed and then a filename is asked for. If you supply no filename, Psychedelic will go to the main menu with everything unchanged. If you supply a filename, then Psychedelic will attempt to BLOAD that filename. If you should get some error and don't wish to erase the format in memory, then type "GOTO 920".

Save The Format

Pressing "4" is identical to pressing "3" except the filename will be saved from memory instead of being loaded.

Closing Notes

This program can be very fun and I'm sure you will enjoy it. Those of you who come up with some outstanding formats should send them to us here at Hardcore COMPUTIST.

Psychedelic

```

10 REM PSYCHEDELIC*SYMPHONY*P
20 REMS DELIC SYMP ONY PS
30 REM Y HED LIC S MPH NY* SY
40 REM C EDE IC SYM HON * SYC
50 REM H IC* YMP ONY* SYCH
60 REM E IC* SY NY*P YCHE
70 REM D LI *SY PHO Y*PS CHED
80 REM E IC* YM HON *PSY HEDE
90 REM LIC*SYMPHONY*PSYCHEDEL
100 REM
110 REM RELOCATE BASIC
120 IF PEEK (104) <> 13 THEN POKE 104
,13 : POKE 3328 ,Z : PRINT CHR$(4
)"RUN^PSYCHEDELIC"
130 GOTO 830
140 REM SWITCH TO GR
150 REM AND PLOT ALL SQUARES
160 POKE - 16304 ,Z : POKE - 16302 ,Z :
CALL - 1998
170 FOR A = 3073 TO PEEK (SF) * 9 + SF
STEP 9 : IF A = SQ THEN NEXT : GOTO
190
180 GOSUB 730 : NEXT
190 GOSUB 1180
200 REM FLASH CURRENT SQUARE
210 A = SQ : GOSUB 730 : A1 = I : FOR A =
I TO 25 : IF PEEK (- 16384) > 127
THEN 240
220 NEXT : GOSUB 1200 : A1 = Z : FOR A =
I TO 25 : IF PEEK (- 16384) > 127
THEN 240
230 NEXT : GOTO 210
240 IF A1 THEN GOSUB 1200
250 REM GO TO CORRECT ROUTINE
260 GET AS : FOR A = I TO LEN (K$) : IF
AS <> MID$ (K$ , A , I) THEN NEXT :
GOTO 210
270 ON A GOTO 290 ,340 ,350 ,300 ,390
,460 ,540 ,530 ,450 ,610 ,570
,650 ,420 ,920
280 REM MOVE SQUARE UP/DOWN
290 YS = - I : GOTO 310
300 YS = I
310 IF CY - Y1 + YS < Z OR CY + Y1 + YS >
47 THEN 210
    
```

```

320 POKE SQ+5,CY+YS:GOTO 210
330 REM MOVE SQUARE LEFT/RIGHT
340 XS=-I:GOTO 360
350 XS=I
360 IF CX-X1+XS<Z OR CX+X1+XS>
39 THEN 210
370 POKE SQ+8,CX+XS:GOTO 210
380 REM GRAB NEXT SQUARE
390 IF SQ=(PEEK(SF)-I)*9+3073
THEN SQ=3073:GOTO 160
400 SQ=SQ+9:GOTO 160
410 REM GRAB LAST SQUARE
420 IF SQ=3073 THEN SQ=(PEEK(SF)
-I)*9+3073:GOTO 160
430 SQ=SQ-9:GOTO 160
440 REM ENLARGE/SHRINK VERTICAL
450 YS=-I:GOTO 470
460 YS=I
470 Y1=47:X1=3
480 XS=PEEK(SQ+X1):FOR A=XS TO
252*(YS=I)+I STEP YS:IF INT
(255/A)=INT(255/XS) THEN
NEXT
490 XS=A:FOR A=XS TO 252*(YS=I)
+I STEP YS:IF INT(255/A)=
INT(255/XS) THEN NEXT
500 XS=A-YS:IF CY-INT(255/XS)
<0 OR CY+INT(255/XS)>Y1 OR
XS>255 THEN 210
510 POKE SQ+X1,XS:GOTO 210
520 REM ENLARGE/SHRINK HORIZONTAL
530 YS=-I:GOTO 550
540 YS=I
550 CY=CX:Y1=39:X1=6:GOTO 480
560 REM DELETE CURRENT SQUARE
570 IF PEEK(SF)=I THEN 210
580 POKE SF,PEEK(SF)-I:A1=SQ
:A2=SQ+9
590 A3=3327:GOSUB 1220:SQ=3073:
GOTO 160
600 REM ADD ANOTHER SQUARE
610 IF PEEK(SF)=25 THEN 210
620 SQ=9*PEEK(SF)+3073:POKE SF
,PEEK(SF)+I
630 A1=SQ:A2=SQ-9:A3=A2+8:
GOSUB 1220:GOTO 160
640 REM EDIT THE STATUS
650 TEXT:HOME:PRINT SPC(17)
"STATUS":PRINT:PRINT
660 PRINT "NEW SAMPLE=>Y" CHR$(8);
:IF PEEK(SQ)<128 THEN PRINT
"N";
670 PRINT:PRINT:PRINT "STEADY^
COLOR=>Y" CHR$(8);:IF PEEK
(SQ+I)<128 THEN PRINT "N";
680 PRINT:PRINT:PRINT "COLOR=>"
INT(PEEK(SQ+2)/16):VTAB 4
:HTAB 13
690 A=Z:GOSUB 780:VTAB 6:HTAB 15
:A=I:GOSUB 780
700 VTAB 8:HTAB 8:INPUT "";$:IF
VAL($)<1 OR VAL($)>15 THEN
160
710 POKE SQ+2,VAL($)*16+VAL
($):GOTO 160
720 REM LIGHT A SQUARE
730 COLOR=RND(I)*15+I:IF PEEK
(A+I)>127 THEN COLOR=PEEK(A
+2)/16
740 CY=PEEK(A+5):CX=PEEK(A+8

```

```

):Y1=INT(255/PEEK(A+3))
:X1=INT(255/PEEK(A+6))
750 VLIN CY-Y1,CY+Y1 AT CX-X1:
VLIN CY-Y1,CY+Y1 AT CX+X1
760 HLIN CX-X1,CX+X1 AT CY-Y1:
HLIN CX-X1,CX+X1 AT CY+Y1:
RETURN
770 REM GET A Y OR N
780 GET $:IF $<>"Y" AND $<>"N"
AND $<>CHR$(13) AND $<>
CHR$(27) THEN 780
790 IF $=CHR$(13) THEN RETURN
800 IF $=CHR$(27) THEN POP:GOTO
160
810 PRINT $:POKE SQ+A,255*(A$=
"Y"):RETURN
820 REM INITIAL SETUP
830 FOR A=749 TO 767:READ PK:POKE
A,PK:NEXT
840 SPEED=255:NORMAL:K$="IJKM"+
CHR$(21)+CHR$(9)+CHR$(10)
+CHR$(11)
850 K$=K$+CHR$(13)+CHR$(14)+
CHR$(4)+CHR$(19)+CHR$(8)+
CHR$(27)
860 POKE 2048,Z:A1=2049:A2=2048
:I=1:SF=3072:Z=0
870 A2=2048:A3=3327:GOSUB 1220
880 IF PEEK(768)<>160 THEN PRINT
CHR$(4)"BLOOD^
SYMPHONY.OBJ,A$300"
890 SQ=3073:POKE SF,I:POKE SQ
,128:POKE SQ+I,Z
900 POKE SQ+3,128:POKE SQ+5,23:
POKE SQ+6,128:POKE SQ+8,19
910 REM MAIN MENU
920 TEXT:HOME:PRINT SPC(5)
"PSYCHEDELIC^SYMPHONY^
FORMATTER"
930 VTAB 4:PRINT "1)^EDIT^THE^
FORMAT":PRINT:PRINT "2)^
VIEW^THE^FORMAT^IN^ACTION"
940 PRINT:PRINT "3)^LOAD^A^
FORMAT":PRINT:PRINT "4)^
SAVE^THE^FORMAT"
950 PRINT:PRINT:PRINT "WHICH^?"
CHR$(8);
960 GET $:IF ($<"1" OR $>"4")
AND $<>CHR$(27) THEN 960
970 REM ESC = EXIT
980 IF $=CHR$(27) THEN HOME:END
990 REM GO TO CORRECT SUBROUTINE
1000 ON VAL($) GOTO 1020,1030
,1080,1130
1010 REM VIEW IT IN ACTION
1020 SQ=3073:GOTO 160
1030 IF PEEK(SF)=Z THEN 920
1040 HOME:VTAB 12:INPUT "DELAY^
CONSTANT^(1-255)^=";$:A:IF A<
I OR A>255 THEN 1030
1050 POKE 755,A:POKE 224,Z:POKE
225,12:GR:POKE-16302,Z
1060 CALL-1998:CALL 754:GET $:
GOTO 920
1070 REM LOAD A FORMAT
1080 HOME:PRINT SPC(14)"LOAD^
FORMAT":POKE 34,I
1090 PRINT CHR$(4)"CATALOG":
PRINT:INPUT "FILE=>";$
1100 IF $="" THEN 920

```

```

1110 PRINT CHR$(4)"BLOOD" $
",A$C00":GOTO 920
1120 REM SAVE THE FORMAT
1130 HOME:PRINT SPC(14)"SAVE^
FORMAT":POKE 34,I
1140 PRINT CHR$(4)"CATALOG":
PRINT:INPUT "FILE=>";$
1150 IF $="" THEN 920
1160 PRINT CHR$(4)"BSAVE" $
",A$C00,L$100":GOTO 920
1170 REM MOVE GR1 TO GR2
1180 A1=2048:A2=1024:A3=2047:
GOTO 1220
1190 REM MOVE GR2 TO GR1
1200 A1=1024:A2=2048:A3=3071
1210 REM
MOVE A2 THROUGH A3 INTO A1
1220 POKE 61,A2/256:POKE 60,A2-
PEEK(61)*256:POKE 63,A3/
256
1230 POKE 62,A3-PEEK(63)*256:
POKE 67,A1/256:POKE 66,A1-
PEEK(67)*256
1240 CALL 749:RETURN
1250 DATA 160,0,76,44,254,169,0
,32,168
1260 DATA 252,32,0,3,44,0,192
,16,243,96
1270 REM HARRAD YAR YB

```

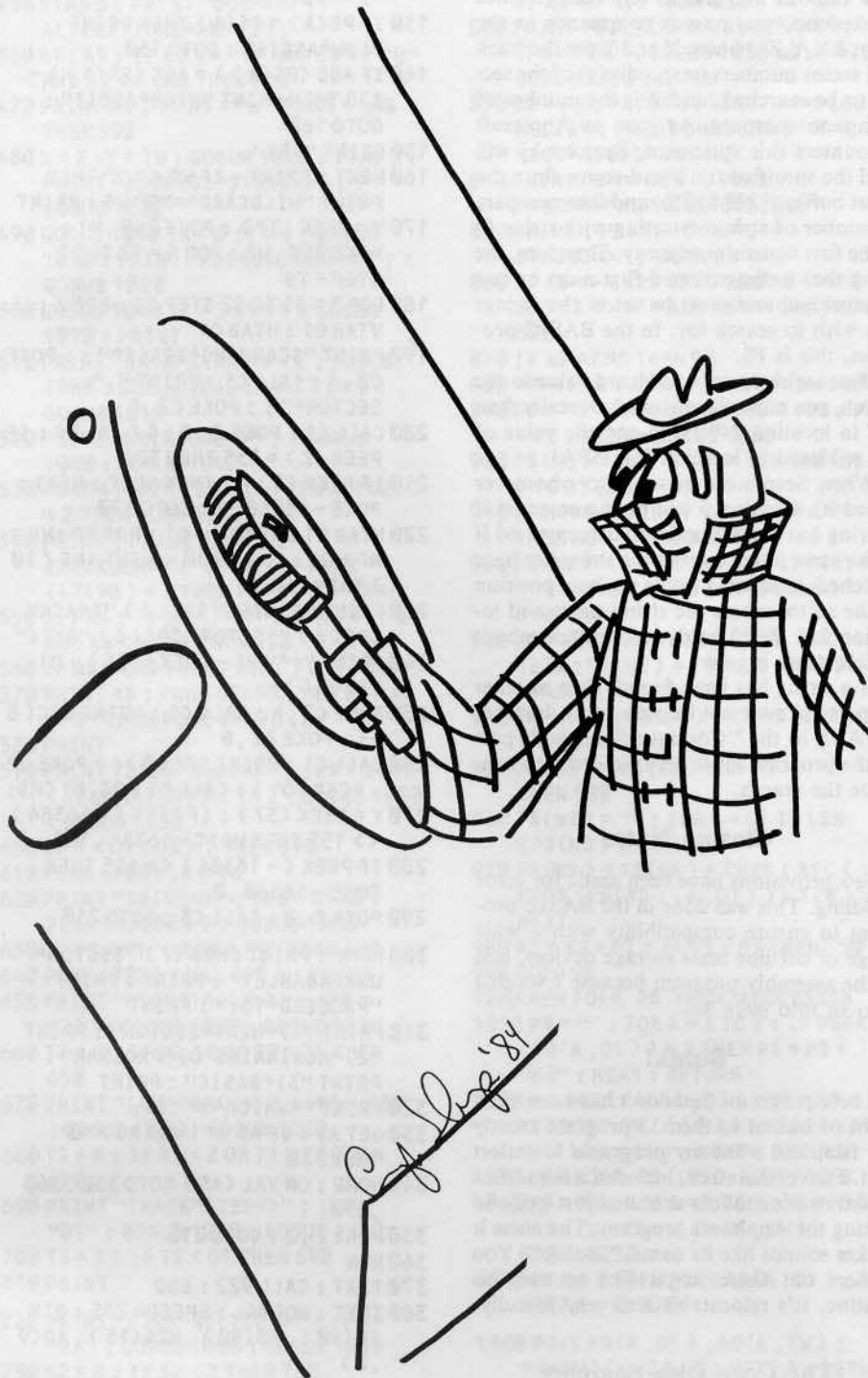
Symphony.OBJ

0300:	A0 00 B1 E0 85 FB C8 C6	\$77D3
0308:	FB 10 01 60 84 E2 C8 C8	\$8B16
0310:	C8 C8 20 95 03 85 FC A5	\$88D3
0318:	2C 85 2D C8 C8 A9 00 85	\$27CA
0320:	30 20 95 03 20 A7 03 A4	\$1E6A
0328:	E2 B1 E0 10 22 A9 00 85	\$0BB4
0330:	FE 85 FF 2C 60 C0 10 FB	\$C71C
0338:	2C 60 C0 30 FB E6 FE F0	\$6880
0340:	05 2C 60 C0 10 F7 E6 FF	\$FC6B
0348:	F0 05 2C 60 C0 30 F7 C8	\$FE6C
0350:	B1 E0 30 11 84 E2 A2 01	\$AD1A
0358:	20 B4 EF A5 9F 20 64 F8	\$30DC
0360:	A4 E2 C8 D0 05 C8 B1 E0	\$6E1D
0368:	85 30 A5 FF 20 85 03 85	\$F047
0370:	FC A5 2C 85 2D A5 FE 20	\$08EA
0378:	85 03 84 E2 20 A7 03 A4	\$7545
0380:	E2 C8 4C 07 03 C8 A2 00	\$5991
0388:	F0 03 F1 E0 E8 D1 E0 B0	\$8BB5
0390:	F9 8A C8 91 E0 B1 E0 85	\$1691
0398:	9F C8 B1 E0 AA 18 65 9F	\$C59F
03A0:	85 2C 8A 38 E5 9F 60 85	\$40C8
03A8:	FD A8 A5 FC 20 19 F8 A5	\$2E8E
03B0:	2D A4 FD 20 19 F8 A5 FC	\$D406
03B8:	A4 FD 20 28 F8 A5 FC A4	\$74A8
03C0:	2C 4C 28 F8	\$06F2

Symphony.OBJ Source Code

0060	CASS.IN .EQ \$C060	CASSETTE INPUT
	LOCATION	
00FE	NUM.X .EQ \$FE	HOW LONG CASS.I
	N STAYED POSITIVE	
00FF	NUM.Y .EQ \$FF	HOW LONG STAYED
	NEGATIVE	
00E0	SQ.PTR .EQ \$E0	POINTS TO SQ
	UARE DEFINITIONS	

Continued on page 29



The CORE Disk Searcher

By Bryce L. Fowler & Ray Darrah

Requirements:

48K Apple with Applesoft
DOS 3.3

Hardcore COMPUTIST publishes some softkeys that involve finding one or more specific bytes on a disk. Often, for one reason or another, the location of these bytes is not known. An example would be the recent softkey for Sierra On-line programs where a disk search program must be used. Since a lot of you out there probably don't have a program with this capability, I wish to present The CORE Disk Searcher.

The CORE Disk Searcher (CDS) will search an entire disk in just over a minute and a half for one string of input. CDS will display the track, sector and starting byte where each search string is found. Provisions have been made to allow for searching less than the entire disk, searching hard disks, scanning protected disks and skipping tracks or sectors.

Typing It In

Type in the Basic program listing at the end of this article using the instructions on page 2 of this magazine and

SAVE CORE DISK SEARCHER

Next, type in the hexdump at the end of this article and

BSAVE SEARCH.OBJ,A\$2F0,L\$B9

Using The Program

When CDS is RUN, you will first be prompted whether to make any **DOS alterations** or not. If you type a "Y", then you may change the address marks, data marks and the option to ignore the checksum or not. If you place a "00" in any of the input string, then that byte is considered ignored. An example would be if you wanted to ignore the second byte of the address start marker you would change the string to "D50096". This feature even allows you to search some protected diskettes.

Next, you will be prompted for the **disk slot** and **drive** numbers where the disk to be searched will be placed. To enter the default values of the last accessed disk, you merely press RETURN.

Third, you will be prompted for the **high track**, **low track** and **track step**. The **high track** is the highest track you wish to be searched. The **low track** is the lowest track

you wish searched. The **track step** is a whole number that determines whether or not to skip any tracks. A track step of one will even-ly search every track. But a track step of two will skip every other track.

A Word About The Inputs

The input values in CDS aren't checked very thoroughly (so as to work on drives with more tracks or sectors etc.) so be sure that you enter them correctly. All values in this program are hexadecimal unless otherwise noted. When two digits are displayed as the default and you must type a preceding zero if you wish to change it to a one digit number (ex. type "0E" for a high track of \$E). Pressing RETURN in the middle of a hexadecimal string will not truncate it at the cursor position. However pressing RETURN in the middle of an ASCII string will.

Next you are asked for the **high sector, low sector and sector step**. These are similar to the track prompts preceding them except these deal with sectors instead of tracks.

This is followed by the **slot for printout** prompt. A zero will print the search results to the screen. Any other number will attempt to print the results to a printer in the corresponding slot. If you select a slot other than zero, be sure your printer is on before continuing.

The Wildcard

Next, you will be asked if you wish a wildcard or not. If you answer "Y", then the hexadecimal value of this wildcard must be input. If this wildcard is contained in any of the search strings, that byte in the string will match any byte on the disk.

Entering The Search Strings

Finally, you will be asked to enter the strings to search for. There are three types of search strings (displayed at the top of the screen). They are **low ASCII, high ASCII and hexadecimal**. CDS will first ask you for the type of string that is to follow. You may press a quote for high ASCII, an apostrophe for low ASCII or a dollar sign for hexadecimal. After the type indicator, you may enter the string to search for. Up to eighty search strings may be entered. This should be more than enough for your searching needs. When you finish entering all the search strings, press RETURN when asked for the type of string.

Pausing

CDS will automatically pause every time it finds a search string. To get it going again, press any key. To stop CDS from pausing, press the ESC key.

The Assembly Language Subroutine

The machine language portion of The CORE Disk Searcher is rather unique. When it is BRUN, Search.obj hooks itself up to the

now famous ampersand (&) vector. Once hooked up, you pass it commands in the form &X,Y,Z where X and Y are the track and sector numbers (respectively) of the sector to be searched, and Z is the number of strings to compare. As soon as Applesoft encounters this statement, Search.obj will read the specified track and sector (into the input buffer (\$200-\$2FF)) and then compare Z number of strings (starting with string 0) in the first dimensioned array. Therefore, the array that is dimensioned first must be one dimensional and must be set to the strings you wish to search for. In the BASIC program, this is F\$.

If you wish to use a wildcard value in the search, you must place a number greater than 127 in location 249 (\$F9) and the value of the wildcard in location 250 (\$FA).

When Search.obj returns (to whomever called it), location 0 will be incremented if a string has been found and decremented if a disk error has occurred. If a string has been matched, location 1 holds the byte position in the sector where the string starts and location 255 (\$FF) holds information about which string it was.

If a string has been found, then another ampersand must not be performed. Instead, a CALL to the "Continue Scanning" part of the program must occur in order to continue the search.

Closing Notes

Few provisions have been made for error handling. This was done in the BASIC program to ensure compatibility with a wide range of off-line mass storage devices, and in the assembly program because I wanted it to fit into page 3.

Boink!

I hate programs that don't have some element of humor in them. I program mostly for fun, and want my programs to reflect that. I have, therefore, included a humorous sound routine that is executed just prior to exiting the Applesoft program. The noise it makes sounds like its name, "Boink". You gamers out there might like to use this routine. It's relocatable and very friendly.

The Core Disk Searcher

```

10 REM -----
20 REM \_/ THE CORE \_/
30 REM \_/ DISK SEARCHER \_/
40 REM \_/ \_/
50 REM \_/ BY RAY DARRAH \_/
60 REM \_/ \_/
70 REM
80 GOTO 380
90 NF = -1 : IF NS = Z THEN 800
100 HOME : POKE 34 , TW : PR# PR : PRINT
: VTAB 3
110 PRINT "CORE^DISK^SEARCHER" :
PRINT : PRINT "STRINGS:"
120 FOR A = Z TONS - 01 : PRINT A + 01

```

```

")^" P$(A) "A-^" ;
130 IF P$(A) = F$(A) THEN PRINT
"LOW^ASCII" : GOTO 160
140 IF ASC (P$(A) ) = ASC (F$(A) ) -
128 THEN PRINT "HIGH^ASCII" :
GOTO 160
150 PRINT "HEX"
160 NEXT : PRINT : IF W$ <> "" THEN
PRINT "WILDCARD^=>$" W$ : PRINT
170 Y = PEEK (37) : POKE 249 , W1 :
POKE 250 , W2 : FOR A = T2 TO T1
STEP - TS
180 FOR B = S1 TO S2 STEP SS : PR# Z :
VTAB 01 : HTAB 01
190 PRINT "SCANNING^TRACK^" : POKE
C2 , A : CALL C1 : PRINT " , ^
SECTOR^" : POKE C2 , B
200 CALL C1 : POKE Z , Z : & A , B , NS : IF
PEEK (Z) = 255 THEN 300
210 IF PEEK (Z) = Z THEN NEXT : NEXT :
POKE - 16368 , Z : GOTO 370
220 HTAB 01 : VTAB Y + 01 : PR# PR : NF =
NF + 01 : IF NF / 10 <> INT (NF / 10
) THEN 240
230 PRINT "STRING" SPC (6) "TRACK"
SPC (6) "SECTOR" SPC (6) "BYTE"
240 PRINT "^^" NS - PEEK (255) + 01
TAB (14) ;
250 POKE C2 , A : CALL C1 : PRINT SPC (8
) : POKE C2 , B
260 CALL C1 : PRINT SPC (8) : POKE C2
, PEEK (01) : CALL C1 : PRINT CM$ ;
270 Y = PEEK (37) : IF PEEK (-16384)
<> 155 THEN WAIT - 16384 , 128
280 IF PEEK (-16384) <> 155 THEN
POKE - 16368 , Z
290 POKE Z , Z : CALL C3 : GOTO 210
300 HOME : PRINT CHR$ (7) "SECTOR^
UNREADABLE!" : PRINT : PRINT
"PROCEED^TO:" : PRINT
310 PRINT "1)^NEXT^SECTOR" : PRINT
"2)^BEGINNING^OF^PROGRAM" :
PRINT "3)^BASIC" : PRINT
320 PRINT "A^WHICH^?" CH$ ;
330 GET AS : IF AS < "1" OR AS > "3"
THEN 330
340 HOME : ON VAL (AS) GOTO 350 , 360
, 370
350 POKE Z , Z : GOTO 210
360 RUN
370 TEXT : CALL 922 : END
380 TEXT : NORMAL : SPEED = 255 : DIM
F$(80) , P$(80) , HX$(15) , AD(9
, 2)
390 FOR A = 0 TO 9 : READ AD (A , 0)
, AD (A , 2) : HX$(A) = STR$(A) :
NEXT : FOR A = 0 TO 5
400 HX$(A + 10) = CHR$(65 + A) : NEXT
: IF PEEK (768) + PEEK (769) <>
155 THEN PRINT CHR$(4) "BRUN^
SEARCH.OBJ"
410 FOR A = Z TO 8 : AD (A , 1) = AD (A , 0)
+ 2 : NEXT : AD (A , 1) = AD (A , 0)
420 CH$ = CHR$(8) : CU$ = CHR$(21)
: CM$ = CHR$(13)
430 01 = 1 : Z = 0 : TW = 2 : C1 = 912 : C2 =
918 : C3 = 905 : PR# Z : IN# Z : CALL
1002
440 HOME : PRINT TAB (9) "THE^CORE^
DISK^SEARCHER"

```

```

450 VTAB 5 : PRINT "DOS^
ALTERATIONS=>N" CH$;
460 GET AS : IF AS <> "Y" AND AS <> "N"
AND AS <> CM$ THEN 460
470 PRINT AS : IF AS = "N" OR AS = CM$
THEN 590
480 X = Z : Y = TW : GOSUB 1010 : VTAB 5 :
PRINT "ADDRESS^START=>"; :
GOSUB 1080
490 GOSUB 1040 : X = 3 : Y = 4 : GOSUB
1010 : PRINT "ADDRESS^END=>"; :
GOSUB 1080
500 GOSUB 1040 : X = 5 : Y = 7 : GOSUB
1010 : PRINT
510 PRINT "DATA^START=>"; : GOSUB
1080 : GOSUB 1040 : X = 8 : Y = 9 :
GOSUB 1010
520 PRINT "DATA^END=>"; : GOSUB
1080 : GOSUB 1040
530 POKE 47422 , 201 : IF PEEK (47423 )
= Z THEN POKE 47422 , 41
540 PRINT : PRINT "IGNORE^
CHECKSUM=>N" CH$; : IF PEEK
(47498 ) = Z THEN PRINT "Y" CH$;
550 GET AS : IF AS <> "Y" AND AS <> "N"
AND AS <> CM$ THEN 550
560 IF AS = CM$ THEN PRINT : GOTO 580
570 PRINT AS : POKE 47498 , 183 : IF AS
= "Y" THEN POKE 47498 , Z
580 PRINT
590 PRINT "DISK^SLOT=>"; : PS = STR$
( PEEK (47081 ) / 16 ) : GOSUB 1080
: GOSUB 1230
600 IF R < 01 OR R > 7 THEN 590
610 POKE 47081 , R * 16
620 PRINT "DRIVE=>"; : PS = STR$ (
PEEK (47082 ) ) : GOSUB 1080
630 IF PS < "1" OR PS > "9" THEN 620
640 POKE 47082 , VAL ( PS ) : PRINT
650 PRINT "HIGH^TRACK=>"; : PS =
"22" : GOSUB 1080 : GOSUB 1230
660 T2 = R : IF T2 < Z OR T2 > 35 THEN
650
670 PRINT "LOW^TRACK=>"; : PS = "00"
: GOSUB 1080 : GOSUB 1230
680 T1 = R : IF T1 < Z OR T1 > T2 THEN
670
690 PRINT "TRACK^STEP=>"; : PS =
"01" : GOSUB 1080 : GOSUB 1230
700 TS = R : IF TS < 01 THEN 690
710 PRINT
720 PRINT "HIGH^SECTOR=>"; : PS =
"0F" : GOSUB 1080 : GOSUB 1230
730 S2 = R : IF S2 < Z THEN 720
740 PRINT "LOW^SECTOR=>"; : PS =
"00" : GOSUB 1080 : GOSUB 1230
750 S1 = R : IF S1 < Z OR S1 > S2 THEN
740
760 PRINT "SECTOR^STEP=>"; : PS =
"01" : GOSUB 1080 : GOSUB 1230
: SS = R
770 PRINT
780 PRINT "SLOT^FOR^PRINTOUT=>";
: PS = "0" : GOSUB 1080 : GOSUB
1230
790 PR = R : IF PR > 7 THEN 780
800 HOME : PRINT SPC ( 13 ) "SEARCH^
STRINGS" CM$ : PRINT "USE^A^
WILDCARD?=>N" CH$;
810 PS = "" : W1 = Z : GET AS : PRINT AS :

```

```

IF AS <> "Y" THEN 830
820 PRINT "WILDCARD^VALUE=>$" ;
: PS = "00" : GOSUB 1080 : W1 = 128 :
GOSUB 1230 : W2 = R
830 WS = PS : PRINT : PRINT "'^A^LOW^
ASCII" : PRINT CHR$ ( 34 ) "'^A^
HIGH^ASCII" : NS = Z
840 PRINT "$^A^HEX" : PRINT
"<<CR>^A^NO^MORE^STRINGS"
850 PRINT : PRINT "TYPE^OF^A
STRING=>"; : GET AS
860 IF AS <> CHR$ ( 34 ) AND AS <> "$"
AND AS <> "" AND AS <> CM$ THEN
850
870 IF AS = CM$ THEN 900
880 PRINT AS : IF AS <> "$" THEN 940
890 PS = "" : FOR A = 01 TO 4 : PS = PS
+ $ : NEXT : GOSUB 1080
900 PS ( NS ) = PS : FOR A = LEN ( PS ( NS )
) TO 01 STEP - 01
910 IF MID$ ( PS ( NS ) , A , 01 ) = ""
THEN NEXT : GOTO 850
920 PS ( NS ) = LEFT$ ( PS , A ) : F$ ( NS ) =
"" : FOR A = 01 TO LEN ( PS ( NS ) )
STEP TW
930 PS = MID$ ( PS ( NS ) , A , TW ) : GOSUB
1230 : F$ ( NS ) = F$ ( NS ) + CHR$ ( R )
: NEXT A : GOTO 980
940 INPUT "" ; PS ( NS ) : IF PS ( NS ) = ""
THEN THEN 850
950 F$ ( NS ) = PS ( NS ) : IF AS = ""
THEN 980
960 F$ ( NS ) = "" : FOR A = 01 TO LEN
( PS ( NS ) )
970 F$ ( NS ) = F$ ( NS ) + CHR$ ( ASC (
MID$ ( PS ( NS ) , A , 01 ) ) + 128 ) :
NEXT
980 NS = NS + 01 : IF NS < 81 THEN 850
990 GOTO 900
1000 REM FORM PS FROM ADDRESSES
1010 PS = "" : FOR A = X TO Y : IF PEEK
( AD ( A , 01 ) ) = Z THEN PS = PS +
"00" : NEXT : RETURN
1020 R = PEEK ( AD ( A , Z ) ) : PS = PS +
HX$ ( INT ( R / 16 ) ) + HX$ ( R - INT
( R / 16 ) * 16 ) : NEXT : RETURN
1030 REM POKE PS INTO ADDRESSES
1040 AS = PS : FOR A = X TO Y : PS = MID$
( AS , TW * ( A - X ) + 01 , TW ) :
GOSUB 1230
1050 IF R = Z THEN POKE AD ( A , 01 ) , Z :
NEXT A : RETURN
1060 POKE AD ( A , 01 ) , AD ( A , TW ) :
POKE AD ( A , Z ) , R : NEXT A : RETURN
1070 REM INPUT HEX NUMBER
1080 A = PEEK ( 36 ) : PRINT PS ; : POKE
36 , A : B = 01 : A = LEN ( PS )
1090 GET AS : IF AS = CM$ OR AS = CH$ OR
AS = CU$ THEN 1170
1100 IF B > ( A ) THEN 1090
1110 IF ( AS < "0" OR AS > "9" ) AND ( AS
< "A" OR AS > "F" ) THEN 1090
1120 IF B = 01 AND LEN ( PS ) = 01 THEN
PS = AS : GOTO 1160
1130 IF B = 01 THEN PS = AS + RIGHT$
( PS , A - 01 ) : GOTO 1160
1140 IF B = ( A ) THEN PS = LEFT$ ( PS , A
- 01 ) + AS : GOTO 1160
1150 PS = LEFT$ ( PS , B - 01 ) + AS +
RIGHT$ ( PS , A - B )

```

```

1160 PRINT AS ; : B = B + 01 : GOTO 1090
1170 IF AS = CM$ THEN PRINT : RETURN
1180 IF AS = CH$ AND B = 01 THEN 1090
1190 IF AS = CH$ THEN PRINT AS ; : B = B
- 01 : GOTO 1090
1200 IF B > ( A ) THEN 1090
1210 AS = MID$ ( PS , B , 01 ) : GOTO 1110
1220 REM CONVERT PS TO DECIMAL
1230 R = Z : FOR B = LEN ( PS ) - 01 TO Z
STEP - 01 : FOR C = Z TO 15
1240 IF MID$ ( PS , LEN ( PS ) - B , 01 ) <
> HX$ ( C ) THEN NEXT
1250 R = R + INT ( 16 # B * C ) : NEXT B :
RETURN
1260 REM DATA FOR ALTERED MARKS
1270 DATA 47445 , 240 , 47455 , 242
, 47466 , 231
1280 DATA 47505 , 174 , 47515 , 164
1290 DATA 47335 , 244 , 47345 , 242
, 47356 , 231
1300 DATA 47413 , 10 , 47423 , 170

```

Search.obj

02F0:	A9 00 8D F6 03 A9 03 8D	\$44C3
02F8:	F7 03 A9 4C 8D F5 03 60	\$E0E4
0300:	20 7B DD 20 F2 EB A5 A1	\$34EC
0308:	8D EC B7 20 BE DE 20 7B	\$FB4E
0310:	DD 20 F2 EB A5 A1 8D ED	\$F5B8
0318:	B7 20 BE DE 20 7B DD 20	\$5682
0320:	F2 EB A5 A1 85 FF A9 01	\$3331
0328:	8D F4 B7 A9 00 8D F0 B7	\$4B92
0330:	8D EB B7 A9 02 8D F1 B7	\$F5D5
0338:	20 E3 03 20 D9 03 90 03	\$85D2
0340:	C6 00 60 A9 07 85 FE A4	\$0EA7
0348:	FE B1 6B 85 FD C8 B1 6B	\$3955
0350:	85 FB C8 B1 6B 85 FC C8	\$B9DE
0358:	84 FE A2 00 A0 00 B1 FB	\$D8BF
0360:	24 F9 10 04 C5 FA F0 0A	\$8CC2
0368:	DD 00 02 F0 05 E8 D0 EC	\$F51E
0370:	F0 08 C8 C4 FD B0 08 E8	\$EF89
0378:	D0 E4 C6 FF D0 C9 60 E6	\$5473
0380:	00 E8 CA 88 D0 FC 86 01	\$1F89
0388:	60 A6 01 E8 D0 CE F0 EA	\$57B5
0390:	A9 A4 20 ED FD A9 00 4C	\$5DB5
0398:	DA FD A0 6E AD 30 C0 98	\$E1E3
03A0:	38 E9 01 D0 FB 88 D0 F4	\$03CB
03A8:	60	\$ADB3

Search.obj Source Code

DD7B	FRM.EVAL .EQ \$DD7B	EVALUATES THE B ASIC EXPR
03F5	AMP.VEC .EQ \$3F5	BASIC JSRs TO H ERE WHEN IT GETS AN &
EBF2	INT.CONV .EQ \$EBF2	CONVERTS FAC IN TO AN INTEGRAL NUMBER AT \$A0 AND \$A1
DEBE	COM.CHK .EQ \$DEBE	MAKES SURE COMM AS SEPARATE PARAMETERS
03D9	RWTS .EQ \$3D9	HOOKEED TO THE R WTS
B7EC	IOB.TRK .EQ \$B7EC	RWTS TRACK PARA METER
B7ED	IOB.SECT .EQ \$B7ED	SECTOR NUMBER
03E3	FIND.IOB .EQ \$3E3	LOADS A AND Y W ITH THE ADDRESS OF THE IOB
B7F0	IOB.BUF .EQ \$B7F0	POINTER TO USER DATA BUFFER
B7F4	IOB.CMD .EQ \$B7F4	IOB COMMAND
00A1	VAL .EQ \$A1	WHERE INT.CONV STORES IT'S ANSWER
B7EB	IOB.VOL .EQ \$B7EB	RWTS VOLUME XP ECTED

```

00FF NUM.STRINGS .EQ $FF NUMBER OF STRIN
GS TO MATCH
00FE YSAVE .EQ $FE STRING CURRENTL
Y ON
006B ARRAY.PTR .EQ $6B POINTS TO START
OF ARRAYS
00FD LEN .EQ $FD LENGTH OF CURRE
NT STRING
00FB STR.PTR .EQ $FB AND $FC POINT T
O CURRENT STRING
00FA WILD.CARD .EQ $FA WILDCARD CHARAC
TER
00F9 WILD.ON .EQ $F9 IF B7=1 THEN WI
LDCARD CHARACTER VALID
0200 BUFF .EQ $200 SECTOR BUFFER
0000 FOUND.FLG .EQ $0 IF FOUND, +1
0001 POS .EQ $1 POSITION THAT I
T WAS FOUND IN
FD0D COUT .EQ $FD0D PRINT A AS ASCI
I
FD0A PRBYTE .EQ $FD0A PRINT A AS HEX
C030 SPEAKER .EQ $C030 TOGGLE SPEAKER
LOCATION
.OR $2F0 $2F0-$2FF IS EX
PENDABLE
.TF SEARCH.OBJ

```

* HOOK UP TO AMPERSAND

```

02F0 LDA #MAIN.PRG LSB
02F2 STA AMP.VEC+1
02F5 LDA /MAIN.PRG MSB
02F7 STA AMP.VEC+2
02FA LDA #$4C JMP OPCODE
02FC STA AMP.VEC
02FF RTS HOOKUP COMPLETE

```

* GET PARAMETERS FOR SCAN
* AND READ SECTOR

```

0300 MAIN.PRG JSR FRM.EVAL GET TRACK#
0303 JSR INT.CONV INTEGER PLEASE
0306 LDA VAL GET ANSWER
0308 STA IOB.TRK
030B JSR COM.CHK COMMA
030E JSR FRM.EVAL
0311 JSR INT.CONV
0314 LDA VAL
0316 STA IOB.SECT
0319 JSR COM.CHK
031C JSR FRM.EVAL PUT NUMBER OF
031F JSR INT.CONV STRINGS IN VAL
0322 LDA VAL
0324 STA NUM.STRINGS
0326 LDA #1 READ COMMAND
0328 STA IOB.CMD
032B LDA #0
032D STA IOB.BUF BUF=$200
0330 STA IOB.VOL ANY VOL
0333 LDA #2
0335 STA IOB.BUF+1

```

* TRY TO GET THE SECTOR

```

0338 JSR FIND.IOB GET IOB.PTR
033B JSR RWTS GET SECTOR
033E BCC NO.ERR DOS COULD READ
IT
0340 DEC FOUND.FLG TELL BASIC
0342 RTS

```

* SETUP STRINGS

```

0343 NO.ERR LDA #7 FIRST
0345 STA YSAVE OFFSETT=7

```

* GET THE NEXT STRING

```

0347 GET.STNG LDY YSAVE CURRENT $
0349 LDA (ARRAY.PTR),Y GET
034B STA LEN LENGTH
034D INY
034E LDA (ARRAY.PTR),Y ADDR
0350 STA STR.PTR
0352 INY
0353 LDA (ARRAY.PTR),Y
0355 STA STR.PTR+1
0357 INY
0358 STY YSAVE 4 NEXT $

```

* SCAN FOR A STRING

```

035A TRY.BUF LDX #0 BUFF=0
035C TRY.CHAR LDY #0 STRING=0
035E NXT.CHAR LDA (STR.PTR),Y GET CHAR
0360 BIT WILD.ON ACTIVE?
0362 BPL NO.WILD
0364 CMP WILD.CARD MATCH?
0366 BEQ MATCHED1
0368 NO.WILD CMP BUFF,X MATCH?
036B BEQ MATCHED1
036D INX
036E BNE TRY.CHAR
0370 BEQ NXT.STRING

```

```

0372 MATCHED1 INY STRING DONE?
0373 CPY LEN
0375 BCS TELL.B YES!
0377 INX NEXT BUFFR
0378 BNE NXT.CHAR NOT EOB

```

* TRY FOR ANOTHER STRING

NXT.STRING

```

037A DEC NUM.STRINGS DONE?
037C BNE GET.STNG
037E RTS

```

* TELL BASIC ABOUT MATCH

```

037F TELL.B INC FOUND.FLG FOUND!
0381 INX FOR CONTINUE
0382 BACKUP DEX BACKUP TO
0383 DEY START OF $
0384 BNE BACKUP
0386 STX POS
0388 RTS

```

* CONTINUE SCANNING

```

0389 LDX POS GET OLD POS
038B INX
038C BNE TRY.CHAR GO!
038E BEQ NXT.STRING ..ALWAYS

```

* PRINT A HEX NUMBER

```

0390 LDA #$A4 PRINT A "$"
0392 JSR COUT
0395 LDA #0 DUMMY NUMBER
0397 JMP PRBYTE PRINT IT

```

* BOINK!

```

039A LDY #$6E SET LOOP
039C TOGGLE LDA SPEAKER CLICK
039F TYA
03A0 DELAY SEC DELAY LOOP GETS
03A1 SBC #1 SHORTER EACH
03A3 BNE DELAY TIME THROUGH
03A5 DEY
03A6 BNE TOGGLE
03A8 RTS

```

Disk Searcher Checksums

10	- \$BADD	660	- \$475E
20	- \$9B13	670	- \$D9A7
30	- \$4D3B	680	- \$5EEA
40	- \$AD92	690	- \$14FF
50	- \$C899	700	- \$D82F
60	- \$FF65	710	- \$F4D6
70	- \$A3BF	720	- \$C70B
80	- \$AD37	730	- \$A40A
90	- \$DF7C	740	- \$5A6C
100	- \$50D0	750	- \$DAE2
110	- \$3311	760	- \$92A0
120	- \$8B12	770	- \$0541
130	- \$CBEC	780	- \$5DBA
140	- \$F989	790	- \$8CDD
150	- \$82C7	800	- \$AFCE
160	- \$1ACF	810	- \$11E1
170	- \$2115	820	- \$FEC3
180	- \$4884	830	- \$F5B8
190	- \$92CF	840	- \$9C92
200	- \$039D	850	- \$28EF
210	- \$B90E	860	- \$3244
220	- \$C087	870	- \$29A4
230	- \$0778	880	- \$08A0
240	- \$D909	890	- \$6AEF
250	- \$373C	900	- \$D04D
260	- \$CFBE	910	- \$70C8
270	- \$C8EB	920	- \$7420
280	- \$EB29	930	- \$35B6
290	- \$B12C	940	- \$8167
300	- \$DD73	950	- \$08EE
310	- \$69D2	960	- \$608F
320	- \$AC28	970	- \$A9A0
330	- \$A786	980	- \$74E6
340	- \$401E	990	- \$A6D8
350	- \$F4DE	1000	- \$9DF0
360	- \$082B	1010	- \$8FC5
370	- \$4162	1020	- \$7FF0
380	- \$023C	1030	- \$68B1
390	- \$AFA9	1040	- \$1A63
400	- \$3B8A	1050	- \$F448
410	- \$B9BA	1060	- \$9268
420	- \$EFB4	1070	- \$589C
430	- \$0726	1080	- \$85D7
440	- \$6093	1090	- \$A6C6
450	- \$4763	1100	- \$2230
460	- \$5EAB	1110	- \$16B0
470	- \$0D1F	1120	- \$FF5C
480	- \$AFA0	1130	- \$2728
490	- \$D1F8	1140	- \$A2EE
500	- \$EBE3	1150	- \$E96C
510	- \$34FE	1160	- \$A36F
520	- \$A50D	1170	- \$0DAE
530	- \$73A2	1180	- \$B51F
540	- \$7C77	1190	- \$49A8
550	- \$0123	1200	- \$D695
560	- \$ED12	1210	- \$8853
570	- \$376F	1220	- \$F739
580	- \$C440	1230	- \$7413
590	- \$01CD	1240	- \$C091
600	- \$153A	1250	- \$D3ED
610	- \$0879	1260	- \$C936
620	- \$35CD	1270	- \$FF85
630	- \$A8E5	1280	- \$FFF2
640	- \$AC8E	1290	- \$47A7
650	- \$2434	1300	- \$568D



The Armonitor

By Nick Galbreath

Requirements:
48K Apple with DOS 3.3

Wouldn't it be nice to be able to find the location of that darned "I/O ERROR" on your disk? You could get out your disk editing program and fix it right then and there! Or, how about reconstructing track/sector lists from blown files? What about an easy way of learning what the Disk Operating System does? Dream no more. Your wish has been answered. A small program, *The Armonitor*, does all of the above, and more.

In a nutshell, *The Armonitor* monitors the disk head, recording and printing all it does in a simple, logical way. The format of the recording is "X Y Z" where "X" is the operation the disk head is performing. The possible operations are "S" for a seek, "R" for a read, and "W" for a write operation. The two-digit hexadecimal numbers which follow that, "Y" and "Z", specify the track and sector on which the former operation is in effect. For example, "R 04 0A" would mean that DOS is attempting to read from track \$4, sector \$A.

Typing it In

First enter the monitor.

CALL -151

Next, type in the hexdump at the end of this article using the instructions on page 2 (The How-To's Of Hardcore) of this magazine and save it.

BSAVE ARMONITOR,AS300,LS6C

How to Use It

Using this utility is quite simple: BRUN the program and then type in a "&" to install *The Armonitor*. Unfortunately, *The Armonitor* really slows down disk access time and can be a nuisance for long files. However, removing it is as easy as installing it -- just retype in another "&" and it will 'unhook' itself, waiting for another "&" to revive it again.

The Inner Workings

How *The Armonitor* works is also quite simple. Part one of the program points the ampersand vector to a checking routine which sees if the Read/Write Track/Sector part of DOS has been hooked up to *The Armonitor*. If *The Armonitor* has not been installed, it installs it. If it has been installed, it removes itself and restores DOS.

The next subroutine, the "meat" of the program, first saves the A and Y registers on the stack. Then it loads the operation that is being processed, converts it to the appropriate letter, and prints it. After that, it loads up the track and sector and prints them as hex bytes. The fourth part restores the A and Y registers from the stack and returns control back to the RWTS routine. In addition to these remarks on how

The Armonitor works, the assembly also contains helpful remarks on the internal workings of *The Armonitor*.

Possible Modifications

Modification possibilities are endless: adding a CTRL-Y function in place of or instead of the ampersand, changing the format of the recording, having an auto-printout on a printer -- the list goes on and on. One modification I have developed, which may be used in place of or added to the main routine (part three), creates a small audible click every time a sector is read. Now you can hear the difference made by a program that speeds up DOS!

```
LDX #S03 ;NUMBER OF CLICKS
LOOP LDA #SX ;PITCH FACTOR
JSR $FCA8 ;DELAY SUBROUTINE
STA $C030 ;TOGGLE THE SPEAKER
DEX ;SUBTRACT
BNE LOOP ;KEEP GOING UNTIL ZERO
```

The Armonitor shall hopefully join your bag of useful utilities. I have seen advertised in many magazines (in Hardcore COMPUTIST, too) a drive hook-up which just monitors the track that the disk head is on for \$99! *The Armonitor* gives you that feature, plus it also tells which sector the disk head is on as well as the operation the disk head is doing -- all this for the cover price of the magazine!

Armonitor Hexdump

```
0300: A9 4C 8D F5 03 A9 10 8D $2C5A
0308: F6 03 A9 03 8D F7 03 60 $BB3A
0310: AD 00 BD CD 68 03 F0 0C $C565
0318: A2 03 BD 68 03 9D 00 BD $A4EC
0320: CA 10 F7 60 A9 20 8D 00 $482F
0328: BD A9 39 8D 01 BD A9 03 $F86A
0330: 8D 02 BD A9 EA 8D 03 BD $81C6
0338: 60 48 98 48 20 8B FD AE $F257
0340: F4 B7 BD 65 03 20 ED FD $B89B
0348: A9 A0 20 ED FD AD EC B7 $C742
```

```
0350: 20 DA FD A9 A0 20 ED FD $27B9
0358: AD ED B7 20 DA FD 68 85 $C459
0360: 48 68 85 49 60 D3 D2 D7 $DA26
0368: 84 48 85 49 $696B
```

Armonitor Source Code

```
*****
*
* THE DISK ARMONITOR *
*
* BY NICK GALBREATH *
* 1984 *
*****
.OR $300
.TF ARMONITOR

03F5 AMP.VEC .EQ $3F5 AMPERSAND LOCAT
ION TO PUT A JMP
BD00 RWTS.ENTRY .EQ $BD00 THIS IS A MAIN
POINT OF ENTRANCE FOR THE RWTS
B7EC IOB.TRK .EQ $B7EC NORMAL IOB TRAC
K LOCATION
B7ED IOB.SCT .EQ $B7ED SECTOR LOCATION
B7F4 IOCMD .EQ $B7F4 COMMAND LOCATIO
N
FD8B CROUT .EQ $FD8B ROUTINE THAT PR
INTS RETURN
```

```
FDDA PRBYTE .EQ $FDDA PRINTS A AS HEX
FDED FDED .EQ $FDED PRINTS A AS A-
SCII
```

```
*-----*
* SET UP THE AMPERSAND VECTOR
*-----*
```

```
0300 LDA #S4C JMP OPCODE
0302 STA AMP.VEC MAKE AMP.VEC
0305 LDA #TOGGLE JMP TO
0307 STA AMP.VEC+1 TOGGLE
030A LDA /TOGGLE
030C STA AMP.VEC+2
030F RTS
```

```
*-----*
* HERE IS WHERE THE AMPERSAND
* GOES TO
*-----*
```

```
0310 TOGGLE LDA RWTS.ENTRY
0313 CMP RWTS.OLD NOT HOOKED?
0316 BEQ HOOKUP YES, HOOKUP
```

```
*-----*
* WE ARE HOOKED UP SO UNHOOK
*-----*
```

```
0318 LDX #S03
031A RESTORE LDA RWTS.OLD,X
031D STA RWTS.ENTRY,X
0320 DEX
0321 BPL RESTORE
0323 RTS
```

```
*-----*
* ARMONITOR ISN'T HOOKED UP SO
* CONNECT IT TO DOS
*-----*
```

```
0324 HOOKUP LDA #S20 JSR OPCODE
0326 STA RWTS.ENTRY MAKE RWTS.ENT
RY
```

```
0329 LDA #PRINT.ACT JSR TO
032B STA RWTS.ENTRY+1 PRINT.ACT
032E LDA /PRINT.ACT
0330 STA RWTS.ENTRY+2
0333 LDA #SEA NOP OPCODE
0335 STA RWTS.ENTRY+3
0338 RTS
```

```
*-----*
* THE RWTS JSRs TO HERE BEFORE
* PERFORMING ANY ACTION IF
* ARMONITOR IS HOOKED UP
*-----*
```

```
0339 PRINT.ACT PHA SAVE A & Y
033A TYA IN THE
033B PHA STACK
033C JSR CROUT <CR>
033F LDX IOCMD
0342 LDA CMDS,X GET COMMAND
0345 JSR CROUT PRINT IT
0348 LDA #S40 SPACE
034A JSR CROUT
034D LDA IOB.TRK GET TRACK#
0350 JSR PRBYTE PRINT IT
0353 LDA #S40 SPACE
0355 JSR CROUT
0358 LDA IOB.SCT GET SECTOR#
035B JSR PRBYTE PRINT IT
```

```
035E PLA RESTORE A,Y
035F STA $48 AND STORE
0361 PLA THEM IN $48
0362 STA $49 AND $49
0364 RTS RETURN TO RWTS
```

```
0365 CMDS .AS -'SRW' SEEK, READ, WRITE
0368 RWTS.OLD .HS 84488549 RWTS RESTORE DA
TA
```

Pseudo-ROMS On The Franklin ACE

By Ken Stutzman

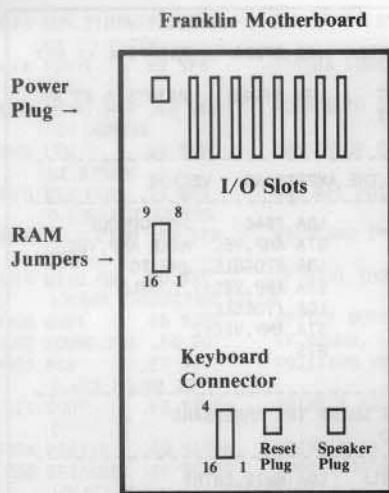


ILLUSTRATION 1

NOTE: This article is designed for readers who have some experience with soldering and electronics. Soft-key Publishing assumes no responsibility for damage done to the computer of any reader who follows the procedure described in this article.

Requirements:

- Franklin Ace 1000/1200
- 1 blank disk
- A DPDT Switch (Radio Shack #275-626)
- A soldering iron
- 4-24" pieces of 22 gauge hook up wire

Have you ever been disappointed by seeing a "flip INTEGER switch and hit RESET" message in the middle of a softkey or had no way of breaking out of a protected program? Well, if you own a Franklin ACE, the minor hardware/software modifications described in this article will allow you to perform these softkeys with ease (in addition to some other things you may not have thought of such as customizing Applesoft). The best part is that you don't have to mess around with burning EPROMs or trying to find an INTEGER card for sale. The total cost of the project is less than the price of a single floppy disk (it cost me \$1.89 in parts).

A Barely Documented Feature

While paging through my Franklin user reference manual, I noticed a list of memory configuration jumpers in Appendix C. A thorough examination of this appendix described the use of jumpers to modify the RAM card to your liking. Two especially useful jumpers can write protect the card until the power is turned off and enable the RAM card while disabling the ROMs.

By connecting pins 4 and 13 of the jumper pad, the built-in language can be written to until it is write inhibited and then it cannot be written to again until the computer is powered off and back on again. If pins 6 and 11 are jumpered in conjunction with the jumpering of pins 4 and 13, then the RAM is permanently read enabled. Thus, when these two jumpers are installed and the RAM

is write inhibited, no software can ever write anything into the \$D000-\$FFFF RAM! I decided to install a switch in place of these jumpers so I could enable these Franklin "Pseudo-ROMS" after some appropriate code had been written into them. So read on, Hardware Hackers. This is where the fun begins!

Hardware Modifications

The first thing to do is to find a small DPDT (Double Pole, Double Throw) Switch such as Radio Shack #275-626, 4 pieces of hook up wire (Gauge 22 or so), a Phillips screwdriver and a soldering iron (*Do not even think of using a soldering gun*). If you have little or no experience with soldering and electronics, DO NOT attempt to make this modification on your own! Get someone who knows what they are doing to help you.

Follow these steps carefully and you should have no problems. It is advisable to work in an uncluttered and well-lit area.

- 1) Unplug your computer from the wall socket and remove the lid and all cards from the motherboard.
- 2) Turn your computer over so the bottom

is facing up and remove the 11 Phillip-type screws on the outer edge of the bottom panel.

3) Carefully lift the bottom panel up and unplug the keyboard connector from the motherboard.

4) Set the bottom panel with the chips up and set the keyboard half in a safe place.

5) Remove the power, reset and speaker connectors. (See ILLUSTRATION 1).

6) Remove the 10 Phillip-type screws from the motherboard and put them in a safe place away from the other 11 screws. Set the bottom panel with the keyboard half.

The procedure so far has been the long-winded way of saying "Get the motherboard out!"

7) Now position the motherboard so the chips are up and the 8 expansion slots are facing away from you.

8) Find the row of chips on the Franklin's motherboard that is marked F'. At the left edge of this row, you should find a dashed white box with the word "OPTION" written beneath it. We will wire our switch to the appropriate solder pads inside of this box. Note: On newer revision motherboards,

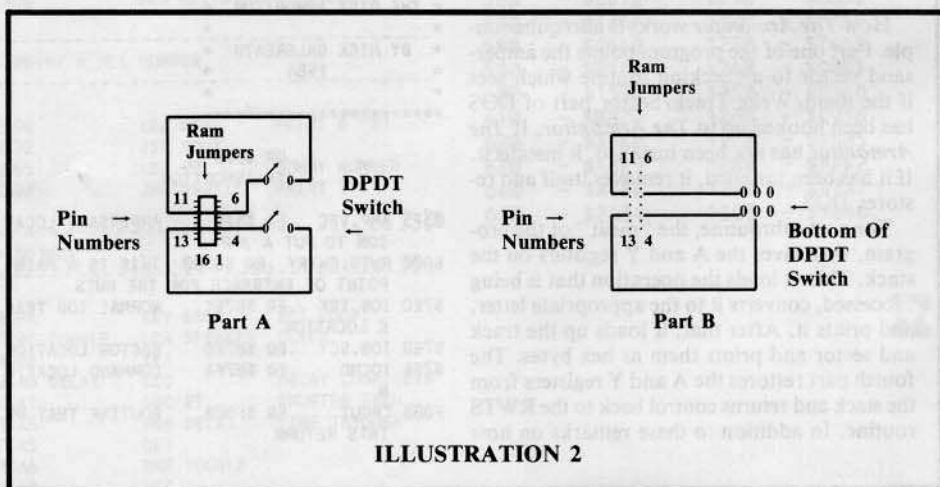


ILLUSTRATION 2

resistors were used for the factory jumpers but the switch will still work the same way.

9) Solder a wire to pin 4, pin 6, pin 11, and pin 13. (See ILLUSTRATION 2).

10) Connect and solder these four wires to the DPDT switch as shown in ILLUSTRATION 2. Part A shows a general schematic and Part B shows how to hook up the wires to the switch.

We are now ready to put the computer back together. The hard part is over!

11) Carefully reposition the motherboard over the bottom panel and reinsert the 10 Phillip-type screws.

12) Plug the power, reset and speaker (don't get the latter two confused) connectors back on the motherboard. (ILLUSTRATION 1).

13) Get the keyboard half and carefully plug the keyboard connector into the motherboard. Make sure pin 1 of the plug goes to pin 1 of the socket.

14) Place the keyboard half and bottom panels together and place the computer so the bottom is facing up.

15) Reinsert the 11 Phillip-type screws in the outer edge of the bottom panel.

16) Turn your computer over and replace your cards, powercord and lid. If the wires on your switch are long enough, run them out the back slots so you don't have to remove the cover when you want to flip the switch. Or, you may want to drill a hole in the case of your computer and mount the switch in it instead.

Congratulations! You're finished (with the hardware part, that is)! You should now be able to turn on your computer and run programs normally. If you can't, try checking the slot your disk drive is in. It should be in slot 6. Be sure your power connector is also plugged in if you can't get anything to work.

Software Modifications

Now that the hard (or hardware) part is finished, let's get to the software part. All you need is a disk with normal DOS 3.3 on it (a System Master will work fine) and one blank disk. Follow these simple instructions and you will have an official Franklin Pseudo-ROM Disk! Look out protected programs!

1) Insert a disk with normal DOS 3.3 and turn your computer on. (A disk with a fast DOS such as Diversi-DOS will also work, as long as it still has an INIT command).

2) Wait until the drive stops and an Applesoft prompt appears. Then clear BASIC program memory by typing

FP

3) Type in the BASIC program found in Listing 1. This will become the HELLO program for our Pseudo-ROMS disk.

4) Take out the disk containing DOS and replace with the blank disk.

5) INITIALize the disk by typing

INIT HELLO

6) Enter the monitor with

CALL -151

7) Move an image of the ROMS into RAM at \$2000-\$4FFF by typing

2000 < D000.FFFF

This allows us to modify an image of ROM.

8) Change the monitor prompt from a normal asterisk to an inverse asterisk by typing

4F6A:2A

This will indicate to us when the Pseudo-ROMs are enabled.

9) To change the power up title from "ACE 1000 v2.2" to "PSEUDO-ROM OK", type

4CFE:D0 D3

4D00:C5 D5 C4 CF AD D2 CF CD

4D08:A0 CF CB

10) To change the RESET and NMI vectors so they will jump to FECD where our small program will reside, type

4FFA:CD FE CD FE

11) Type in the hexdump found in Listing 2. This hexdump was written by Mr. Ernie Young and originally appeared in Hardcore COMPUTIST #6, pg. 14. (You may wish to refer to this article if you want to further examine this program.) It was then modified to include more options and appeared in Hardcore COMPUTIST #8, pg. 26. It has now been included for the sake of any new readers. This program allows the

user to select between a normal autostart reset, an old monitor style reset or one that saves memory locations \$0000-\$900 and puts them in locations \$2000-\$2900. This option will also save the stack pointer, the accumulator, and the X and Y registers at locations \$2091, \$2092, \$2093, \$2094 respectively. 14) Save this Pseudo-ROM code to disk by typing

**BSAVE PSEUDO-ROMS,
AS2000,LS2FFF**

Now all there is left to do is to put our hardware/software modifications to work!

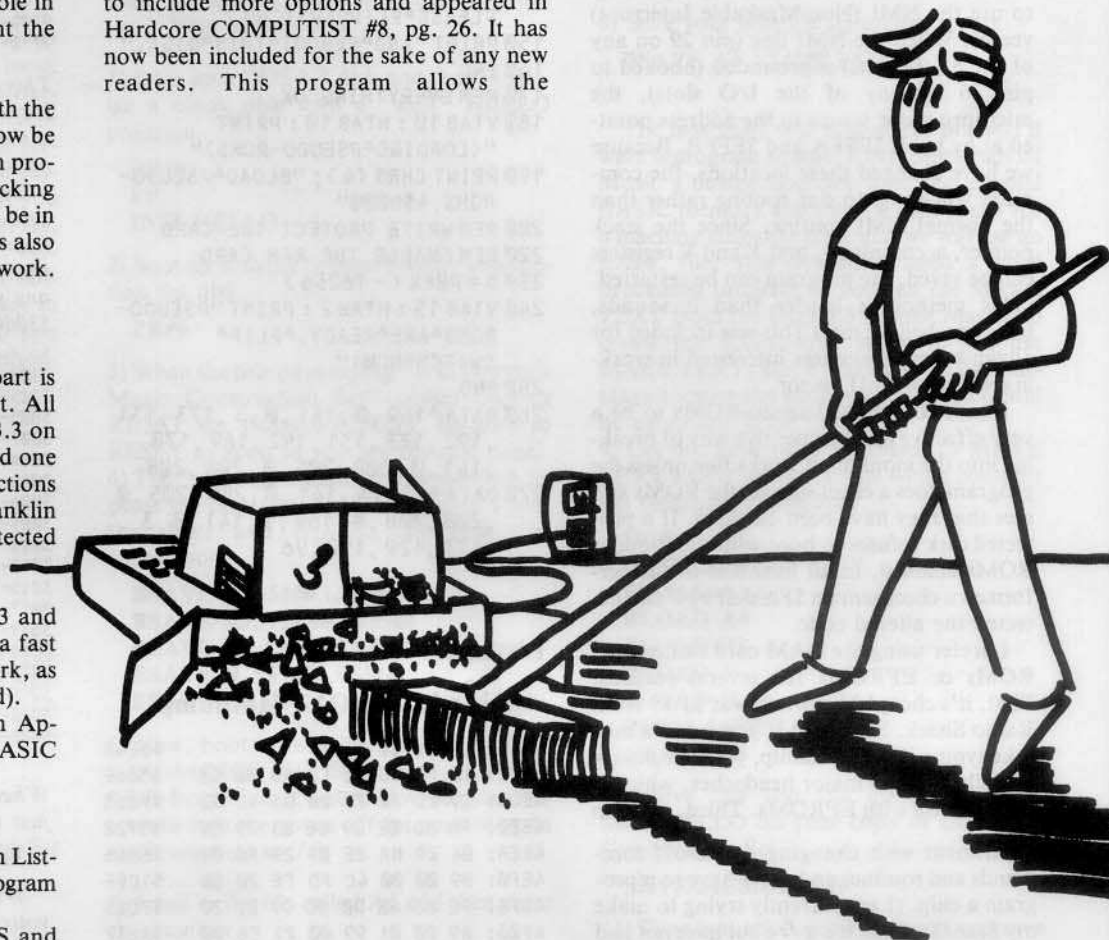
Applications

You now have a disk which contains the Pseudo-ROMs to be loaded in your RAM card. Type

RUN HELLO

and you should get the "FLIP SWITCH NOW" message. If you don't, flip your RAM card switch and try running it again. When you do get the message "FLIP SWITCH NOW", do so and you have Pseudo-ROMS installed in your computer! To test them, type

CALL -1184



The screen should clear and you should get a "PSEUDO-ROM OK" message. You should also get an inverse monitor prompt instead of the normal prompt when you type

CALL -151

If these don't work, type

BLOAD PSEUDO-ROMS

and check Listing 2 and the other modifications. Type

**CALL -151
4ECD.4F26**

Listing 2 should appear exactly as it is in the printed hexdump. If it doesn't, go back and retype it. Be sure to resave the PSEUDO-ROMS file after any modifications.

Now for the true test. Boot your favorite protected program, wait for the boot process to end and hit RESET. Because we have changed the RESET vector address, the computer will jump to our routine instead of the normal RESET routine. Nothing should happen until you hit Return (for normal RESET), a minus sign (for old stye RESET), or a colon for a Super Saver RESET. If none of these work, check your file PSEUDO-ROMS as explained above. Also, check to be sure your switch is properly connected and the solder joints are good if your switch is not working properly.

Another way to break out of programs is to use the NMI (Non-Maskable Interrupt) vector. When the NMI line (pin 29 on any of the 8 I/O slots) is grounded (hooked to pin 26 of any of the I/O slots), the microprocessor jumps to the address pointed at by bytes \$FFFA and \$FFFB. Because we have changed these locations, the computer will jump to our routine rather than the normal NMI routine. Since the stack pointer, accumulator, and X and Y registers can be saved, the program can be restarted. (This method is harder than it sounds, however, believe me!) This was included for advanced programmers interested in working with the NMI vector.

I have found the Pseudo-ROMs to be a very effective and inexpensive way of breaking into the monitor. It works fine unless the program does a checksum on the ROMs and sees that they have been changed. If a protected disk refuses to boot with the Pseudo-ROMs enabled, in all likelihood it has performed a checksum on \$F800-\$FFFF and detected the altered code.

I prefer using the RAM card rather than ROMs or EPROMs for several reasons. First, it's cheap! My switch was \$1.89 from Radio Shack. Secondly, if you make a mistake typing in the hexdump, you can change it easily with no major headaches, which is not the case with EPROMs. Third, you can experiment with changing Applesoft commands and routines and never have to reprogram a chip. (I am currently trying to make my Franklin think it's a //e but have not had much luck. Anybody got any ideas?) Finally, once the switch is in, you never have to

touch the motherboard again!

I hope you found this article to be as informative and educational for you to read as it was for me to prepare. Hopefully, this article will encourage readers to write more programs designed specifically for us Franklin Ace users.

I assume there is some way to write protect the APPLE //e's RAM card, but someone else will have to do the honors. I'm having too much fun with my Franklin!

Listing 1

Pseudo ROMS Hello

```

10 REM PSEUDO-ROMS LOADER
20 REM BY KEN STUTZMAN
30 REM JUNE 1984
40 HOME
50 VTAB 3 : HTAB 11 : PRINT
  "FRANKLIN^PSEUDO-ROMS"
60 VTAB 5 : HTAB 8 : PRINT
  "DEVELOPED^BY^KEN^STUTZMAN"
70 FOR AD = 769 TO 807 : READ BY : POKE
  AD, BY : NEXT
80 CALL 769
90 IF PEEK (768) = 1 THEN 170
100 REM CARD WRITE PROTECTED
110 REM OR DISCONNECTED
120 PRINT : PRINT
130 PRINT "RAM^CARD^IS^EITHER^
  WRITE^PROTECTED"
140 PRINT "OR^DISCONNECTED.^
  PLEASE^FLIP^SWITCH"
150 PRINT "AND^REBOOT^THIS^DISK."
160 END
170 REM EVERYTHING OK
180 VTAB 10 : HTAB 10 : PRINT
  "(LOADING^PSEUDO-ROMS)"
190 PRINT CHR$(4) ; "BLOAD^PSEUDO-
  ROMS, A$D000"
200 REM WRITE PROTECT THE CARD
220 REM ENABLE THE RAM CARD
230 D = PEEK (-16256)
240 VTAB 15 : HTAB 2 : PRINT "PSEUDO-
  ROMS^ARE^READY.^FLIP^
  SWITCH^NOW!"
250 END
260 DATA ^169, ^0, ^141, ^0, ^3, ^173, ^131
  ^, ^192, ^173, ^131, ^192, ^169, ^170
  ^, ^141, ^0, ^208, ^205, ^0, ^208, ^208
270 DATA ^17, ^74, ^141, ^0, ^208, ^205, ^0
  ^, ^208, ^208, ^8, ^169, ^1, ^141, ^0, ^3
  ^, ^173, ^129, ^192, ^96

```

Listing 2

Modified ROMS Hexdump

```

4ECD: 2C 00 C0          $EE4B
4ED0: 10 FB 8D 10 C0 AD 00 C0 $5669
4ED8: C9 2D F0 7D B0 03 4C 62 $FB58
4EE0: FA 8D 02 29 8E 03 29 8C $3722
4EE8: 04 29 BA 8E 01 29 A0 00 $6AA6
4EF0: B9 00 00 4C FD FE 20 00 $1C9F
4EF8: FE 68 68 D0 6C 99 00 20 $7C25
4F00: B9 00 01 99 00 21 C8 D0 $AE37
4F08: E7 84 3C 84 42 84 3E A9 $980A
4F10: 09 85 3F A9 02 85 3D A9 $7015

```

4F18: 22 85 43 20 2C FE 20 2F \$3A66
4F20: FB 20 58 FC 4C 59 FF \$28CD



Continued from page 5

These two conditions are not met when you try to execute BOOT1 because:

- A) The disk drive motor has been turned off with the instruction STA \$C0E8 (8D E8 C0).
- B) Some of the necessary information stored on zero page is lost when the Apple's monitor is entered at \$FF59 (4C 59 FF). This is why location \$2B contains a \$07 instead of the slot number * 16 (\$60 for slot 6).

To get around this problem, put the following code at \$96F8 after moving the disk controller code down to page \$96.

```

96F8- A0 43      LDY  #43
96FA- A9 00      LDA  #00
96FC- 85 FC      STA  $FC
96FE- 85 FD      STA  $FD
9700- 85 FE      STA  $FE
9702- A9 60      LDA  #60
9704- 85 FF      STA  $FF
9706- B1 FC      LDA  ($FC),Y
9708- 91 FE      STA  ($FE),Y
970A- 88        DEY
970B- 10 F9     BPL  $9706
970D- 8D E8 C0 STA  $C0E8
9710- 4C 59 FF  JMP  $FF59

```

This code will move the necessary zero page locations up to page \$60 before the disk drive motor is turned off and the monitor is entered.

Before BOOT1 at \$801 is executed, the zero page locations must be restored from page \$60 and the disk drive motor should have been spinning for at least one second. The following code placed at \$3000 will do this.

```

3000- 8D E9 C0  STA  $C0E9
3003- A0 09      LDY  #09
3005- A9 C0      LDA  $C0
3007- 20 A8 FC  JSR  $FCA8
300A- 88        DEY
300B- D0 F8      BNE  $3005
300D- A0 43      LDY  #43
300F- A9 00      LDA  #00
3011- 85 FC      STA  $FC
3013- 85 FD      STA  $FD
3015- 85 FE      STA  $FE
3017- A9 60      LDA  #60
3019- 85 FF      STA  $FF
301B- B1 FE      LDA  ($FE),Y
301D- 91 FC      STA  ($FC),Y
301F- 88        DEY
3020- 10 F9     BPL  $301B
3022- 4C 01 08  JMP  $0801

```

When you are ready to execute BOOT1 just type

3000G

We hope this information helps solve your problem.



Backup And Modify Music Construction Set

By Dan Rosenberg

Many readers probably own a copy of Music Construction Set which, in my opinion, is the best music program currently available for the Apple. However, I suspect that many owners of MCS are dismayed, like I was, by their inability to back up or modify the program. Because Music Construction Set is protected, it has an annoying habit of checking for an original program disk from time to time. For the user, this copy protection measure means increased wear on the original MCS disk in addition to slowing down the overall process of composing music.

The Music Construction Set disk uses a version of DOS 3.3 called DOS 3.3P (the P stands for Protected.) Most copy programs will copy the Music Construction Set disk without errors, but the copy will not work because the program(s) check the disk for its originality.

Luckily it is not too difficult to defeat 3.3P. Once the protection has been defeated it is possible to make some modifications to Music Construction Set.

The Copy

The first thing to do is determine which version of Music Construction Set you have. The different versions require entirely different copy methods. I know of two different versions, which I will call version 1 and version 2. Version 1 does not support the cassette output function (it won't ask you if you would like cassette output), and the files A3, A4, P3, N and O appear in the catalog. Version 2 allows you to use cassette port output, and A3, A4 and P3 do not appear in the catalog. Check for which version you have, and then use the appropriate method.

MCS Version 1 Copy

Requirements:

48K Apple II plus or equivalent
COPYA or bit copy program
A blank disk

1) Boot up with any DOS 3.3 disk

PR#6

2) Copy your Music Construction Set disk with COPYA onto the blank disk. If COPYA has any trouble making a copy, get out your bit copy program and copy tracks \$0-\$22 with the standard parameters.

RUN COPYA

3) Insert the copy of Music Construction Set and load the binary file called A4

BLOAD A4

4) Enter the monitor by typing

CALL -151

5) Make the following modifications to the image of A4 in memory

**9131:60
913A:EA EA
4C00:60**

6) Resave the file A4 by typing

BSAVE A4,AS4A00,LSB60

7) Boot up your copy of the Music Construction Set disk and start constructing (musically, that is).

MCS Version 2 Copy

Requirements:

Apple II plus or equivalent
A blank disk
Text Editing program or MAKETEXT from the System Master disk

1) Boot up a DOS 3.3 disk and then initialize a blank disk with a "null" HELLO program

**PR#6
FP
INIT HELLO**

2) Boot up with the original Music Construction Set disk

PR#6

3) When the title page saying "Will Harvey's Music Construction Set" comes up, hit RESET (or Control-RESET). Be sure to RESET as soon as you see the title page!

4) The ^ prompt should now be showing, so place the disk you initialized in step 1 into the drive and type in the following commands

**BSAVE H,AS400,LS600
BSAVE A3,ASA00,LS4000
BSAVE A4,AS4A00,LS4B60
BSAVE N,AS7400,LS120
BSAVE P3,AS300,LSD0**

5) Now, boot up with the disk which contains the MCS files. Since you can't do a PR#6 from Music Construction Set, you will have to turn the power off (or control-open apple-reset on a //e).

6) Load in the file called A4 and then enter the monitor

**BLOAD A4
CALL -151**

7) Next, type

86D9.86DB

If you get 4C 00 C6 in response, then type

**86D9:EA EA EA
7F39:60**

Note: If your new copy doesn't work, type in

910D:60

If you didn't get 4C 00 C6, then type

**9131:60
913A:EA EA**

8) Save your changes by typing

BSAVE A4,AS4000,LS4B60

The new copy will now work, but you'll want a program to start it off. Since part of Music Construction Set uses the normal BASIC memory and there is little room for a machine language program, we will use an EXEC file. You will either need a word processing program or you can use MAKE TEXT from the DOS 3.3 System Master to create the EXEC file. Note: if you use MAKE TEXT, don't make any typing mistakes because the backspace characters will be saved into your file.

9) Get out your word processor or MAKE TEXT and create a text file which contains the following commands

**HGR
POKE-16301,0
BLOAD A3
BLOAD A4
BLOAD P3
BLOAD N
BLOAD H
POKE -16368,13
CALL 2156**

10) Save this text file under the name MCS.HELLO on your copy of the Music Construction Set.

SAVE MCS.HELLO

11) Type in the following program and save it on the disk as the HELLO program

FP

10 PRINT : PRINT CHR\$(4) "EXEC"
MCS.HELLO"

SAVE HELLO

12) To copy any of the music files from the original MCS disk to the copy, a file transfer program like FID can be used. Both the music file and its .OBJ file have to be transferred. For instance, if you want the song Dixie on your copy of MCS, then the files DIXIE and DIXIE.OBJ will both have to be transferred.

13) You can now boot up your new Music Construction Set disk and use it normally. If you are using a DOS other than DOS 3.3 (Pronto-DOS, Diversi-DOS, etc.), you'll find the program will run much faster. I recommend you get one of these fast DOS's as they are well worth the money.

Alternate Method For MCS Version 2

Requirements:

- 48K Apple II plus or equivalent
- COPYA or a bit copy program
- Disk Editing program
- A blank disk

1) Make a copy of the original Music Construction Set disk with COPYA (if COPYA won't copy it, use a bit copier)

RUN COPYA

2) Use your disk editor to make the following changes to the copy of the Music Construction Set

Trk	Sect	Byte	From	To
\$B	\$2	\$09	\$4C	\$EA
\$B	\$2	\$0A	\$00	\$EA
\$B	\$2	\$0B	\$C6	\$EA
\$B	\$D	\$29	\$20	\$EA
\$B	\$D	\$2A	\$00	\$EA
\$B	\$D	\$2B	\$4C	\$18

Don't forget to write the modified sectors back to the copy of MCS.

MCS Modifications

It is also possible to customize the Music Construction Set. For example, say that you have your Mockingboard in a slot other than 4. The slot that MCS expects to find the Mockingboard in can be easily changed with the program below. Note that there are two different data statements and you should use one or the other depending upon which version of MCS you own. Just type the program in (using the appropriate DATA statement) and save it as MB SLOT CHANGER by typing

SAVE MB SLOT CHANGER

Just RUN the program when you want to have MCS utilize a Mockingboard in another slot. The program can also be used if MCS will not boot because you have a card in slot 4 that is not a Mockingboard. Just run the MB SLOT CHANGER program and then

change the Mockingboard slot to one of the slots in your computer that is empty.

```
0 REM DATA FOR VERSION 1
5 DATA ^35946,35951,35956,35962,
  ,35967,35972,35982,35987,
  ,35992,35998,36003,36008,
  ,36019,36022,36027,36030
6 REM DATA FOR VERSION 2
10 DATA ^35932,35937,35942,35948,
  ,35953,35958,35968,35973,
  ,35978,35984,35989,35994,
  ,36008,36005,36013,36016
15 REM
  TYPE IN ONLY 1 SET OF DATA!!
20 DIM A(16) : TEXT : HOME
30 FOR I = 1 TO 16 : READ A(I) : NEXT
40 INVERSE : PRINT "MCS^
  MOCKINGBOARD^SLOT^CHANGER"
50 NORMAL : V TAB 5 : PRINT "WHAT^
  SLOT^WOULD^YOU^LIKE^TO^PUT^
  YOUR^MOCKINGBOARD^IN,^OR^
  HAVE^MUSIC^CONSTRUCTIO^SET^
  LOOK^FOR^A^MOCKINGBOARD?^
  (1-7):" ; : GET Z$
60 Z = VAL(Z$) : IF Z < 1 OR Z > 7 THEN
  HTAB 1 : GOTO 50
70 PRINT "MAKE^SURE^YOUR^MUSIC^
  CONSTRUCTION^SET^DISK^IS^IN^
  THE^DRIVE" ; : GET Z$
80 PRINT : PRINT CHR$(4) "BLOAD^A4"
90 FOR I = 1 TO 16 : POKE A(I), Z + 192
  : NEXT
100 PRINT : PRINT CHR$(4)
  "BSAVE^A4, A$4A00, L$4B60"
110 PRINT : PRINT "DONE." : END
```

If you performed the alternate copy method (sector edit method) on version 2, then the above program will not work for you because your disk does not have a file called A4 on it. However, you can still change the Mockingboard slot if you have a sector editor. The bytes to modify are stored on track \$B, sector \$8.

Get your disk editor running and read in track \$B, sector \$8 from the copy of MCS you made. The table below shows the bytes to modify. You will need to substitute the slot number you want your Mockingboard in for the n in the "To" column. For instance, if you want to use slot 2 for the Mockingboard, change the bytes listed in the table from \$C4's to \$C2's.

Byte	From	To	Byte	From	To
\$5C	\$C4	\$Cn	\$8A	\$C4	\$Cn
\$61	\$C4	\$Cn	\$90	\$C4	\$Cn
\$66	\$C4	\$Cn	\$95	\$C4	\$Cn
\$6C	\$C4	\$Cn	\$9A	\$C4	\$Cn
\$71	\$C4	\$Cn	\$A5	\$C4	\$Cn
\$76	\$C4	\$Cn	\$A8	\$C4	\$Cn
\$80	\$C4	\$Cn	\$AD	\$C4	\$Cn
\$85	\$C4	\$Cn	\$B0	\$C4	\$Cn

After changing the bytes, don't forget to write the sector back to your disk.

Cassette Port Output

Version 1 of Music Construction Set does

not allow you to use the cassette port for output as does version 2. Using the cassette port for output allows you to play the music through an external amplifier and speaker for improved sound quality (especially if you turn the treble all the way down on your amplifier). Adding cassette port output to version 1 of MCS is really quite simple if you have a little knowledge of the Apple's built-in I/O.

Sound can be output to the Apple's speaker by referencing address \$C030 or to the cassette output by referencing address \$C020. The binary file called A4 controls the output of MCS, and the changes necessary for cassette output need only be applied to this one file. The program listed below will allow you to pick the output path (speaker or cassette) and will make the necessary modifications to A4. Type in this program and SAVE it as CASSETTE OUTPUT.

```
10 DATA ^23,43,46,66
20 TEXT : HOME
30 FOR I = 1 TO 4 : READ A(I) : NEXT
40 INVERSE : PRINT "MCS^
  CASSETTE^APPLE^SPEAKER^
  OUTPUT^TOGGLE"
50 NORMAL : V TAB 5 : PRINT "DO^YOU^
  WANT^<C>ASSETTE^OR^<A>PPLE^
  SPEAKER^OUTPUT^(C^OR^A):" ; :
  GET Z$
60 IF Z$ <> "C" AND Z$ <> "A" THEN
  HTAB 1 : GOTO 50
70 PRINT : PRINT CHR$(4) "BLOAD^A4"
80 N = 33 : IF Z$ = "A" THEN N = 49
90 FOR I = 1 TO 4 : POKE A(I) + 21800, N
  : NEXT
100 PRINT : PRINT CHR$(4)
  "BSAVE^A4, A$4A00, L$4B60"
110 PRINT "DONE." : END
```

When you want to switch from speaker to cassette output (or vice-versa), just

RUN CASSETTE OUTPUT

Final Words

Since your new Music Construction Set is now on a normal DOS 3.3 disk, if a DOS error is generated (like trying to load a non-existent file), the program will leave you in BASIC (version 2 only). To return to MCS just type

CALL 2156

Anything that you were working on at the time of the error will still be intact.

That about does it for my modifications to Music Construction Set. I am sure that if you poke around a bit, you can come up with some more enhancements for MCS. A good place to start investigating is the area around \$7F00. Happy constructing!



Continued from page 18

```

EFB4 RANDOM .EQ $EFB4 GENERATES A RAN
DOM NUMBER
009F RND .EQ $9F RANDOM RETURNS
HERE
F864 SETCOL .EQ $F864 MAKES THE CO
LOR BYTE EQUAL TO A
0030 COLOR .EQ $30 COLOR MASK BYTE
00E2 YSAVE .EQ $E2 AREA FOR SAVING
CURRENT INDEX INTO DEFINITIONS
00FD H1 .EQ $FD LEFT EDGE FOR S
QUARE
00FC V1 .EQ $FC TOP COORDINATE
FOR SQUARE
002C H2 .EQ $2C MONITOR RIGHT E
DGE FOR HLINE
002D V2 .EQ $2D MONITOR TOP EDG
E FOR VLINE
00FB SQ.CONT .EQ $FB COUNTS DOWN THE
SQUARE AS THEY ARE PLOTTED

F819 HLINE .EQ $F819 DRAW HORIZ LINE
AT (A) FROM (Y) THROUGH H2
F828 VLINE .EQ $F828 DRAW VERTICAL L
INE AT (Y) FROM A THROUGH V2

.OR $300 FIT INTO PAGE T
HREE BARELY
.TF SYMPHONY.OBJ

0300 START LDY #0 GET GET NUMBE
R SQRS
0302 LDA (SQ.PTR),Y
0304 STA SQ.CONT
0306 INY

0307 NXT.SQRE DEC SQ.CONT DONE WITH ALL?
0309 BPL DO.SQRE NOPE!
030B RTS

030C DO.SQRE STY YSAVE SAVE POSITION
030E INY MOVE IT TO
030F INY
0310 INY
0311 INY Y ANSWER
0312 JSR SQ.EDGE CALCULATE EDGE
0315 STA V1 SAVE V1
0317 LDA H2 MOVE H2 TO V2
0319 STA V2
031B INY SKIP A BYTE
031C INY
031D LDA #0 COLOR=
031F STA COLOR
0321 JSR SQ.EDGE DO HORIZONTALS
0324 JSR PLOT.SQUARE

0327 LDY YSAVE
0329 LDA (SQ.PTR),Y NEW SAMPLE?
032B BPL SAME.SAMPLE NO

032D NEW.SAMP LDA #0 ZERO THE
032F STA NUM.X PLUS BYTE
0331 STA NUM.Y AND MINUS BYTE

0333 WAIT1 BIT CASS.IN WAIT ONE
0336 BPL WAIT1 COMPLETE CYCLE

0338 WAIT2 BIT CASS.IN
033B BMI WAIT2

PLUS.COUNTER
033D INC NUM.X COUNT THEM
033F BEQ NEG.COUNTER IF MAXIMUM
0341 BIT CASS.IN OR UNTIL MINUS
0344 BPL PLUS.COUNTER

NEG.COUNTER
0346 INC NUM.Y COUNT THEM
0348 BEQ SAME.SAMPLE IF MAXIMUM
034A BIT CASS.IN OR UNTIL PLUS
034D BMI NEG.COUNTER

SAME.SAMPLE
034F INY NEXT BYTE
0350 LDA (SQ.PTR),Y STEADY COLOR
0352 BMI COPY.COLOR YES!
0354 STY YSAVE SAVE Y

```

```

0356 LDY #1 LIKE RND(1)
0358 JSR RANDOM GET RANDOM ONE
035B LDA RND COLOR FOR NEXT
035D JSR SETCOL SQUARE
0360 LDY YSAVE GET INDEX BACK
0362 INY SKIP COLOR
0363 BNE DIV.Y ..ALWAYS
COPY.COLOR
0365 INY COLOR BYTE
0366 LDA (SQ.PTR),Y GET IT
0368 STA COLOR DONE!

036A DIV.Y LDA NUM.Y DIVIDE Y
036C JSR DIVIDE BY Y DIVISOR
036F STA V1 SAVE TOP EDGE
0371 LDA H2 MOVE BOTTOM
0373 STA V2 EDGE TO V
0375 LDA NUM.X
0377 JSR DIVIDE DO THE X COOR
037A STY YSAVE BECAUSE OF PLOT
037C JSR PLOT.SQUARE MAKE A SQUAR
E

* -----
* DO THE NEXT SQUARE
* -----
037F LDY YSAVE RESTORE INDEX
0381 INY NEXT SQUARE
0382 JMP NXT.SQRE

* -----
* DIVIDE A BY DIVISOR IN PARM LIST
* -----
0385 DIVIDE INY NEXT BYTE
0386 LDY #0 DIVIDE A BY
0388 BEQ DIV.CMP (SQ.PTR),Y

038A SUBTR SBC (SQ.PTR),Y C=1
038C INX YET ANOTHER 1
038D DIV.CMP CMP (SQ.PTR),Y STILL BIGGER
038F BCS SUBTR YES, SUBTRACT
0391 TXA A=ANSWER
0392 INY STORE IT
0393 STA (SQ.PTR),Y

* -----
* CALCULATE EDGES OF A SQUARE
* -----
0395 SQ.EDGE LDA (SQ.PTR),Y GET ANSWER
0397 STA RND SAVE IT
0399 INY
039A LDA (SQ.PTR),Y GET CENTER
039C TAX SAVE IT
039D CLC
039E ADC RND FAR EDGE
03A0 STA H2 FOR XS ONLY
03A2 TXA
03A3 SEC
03A4 SBC RND NEAR EDGE
03A6 RTS

* -----
* PLOT A SQUARE
* -----
03A7 PLOT.SQUARE STA H1 SAVE LEFT EDGE
03A9 TAY DO HORIZ LINE
03AA LDA V1
03AC JSR HLINE DO TOP
03AF LDA V2
03B1 LDY H1
03B3 JSR HLINE DO BOTTOM
03B6 LDA V1 DO VERTIC LINES
03B8 LDY H1
03BA JSR VLINE DO LEFT EDGE
03BD LDA V1
03BF LDY H2 DO RIGHT EDGE
03C1 JMP VLINE

```

```

90 - $924D 690 - $7B94
100 - $CB63 700 - $D092
110 - $BD13 710 - $AD14
120 - $BF30 720 - $DB22
130 - $0826 730 - $1500
140 - $7CE3 740 - $D39E
150 - $65F4 750 - $4E3B
160 - $540A 760 - $CEE2
170 - $1556 770 - $A7ED
180 - $2A87 780 - $F766
190 - $6EAC 790 - $1BA0
200 - $FEA3 800 - $6B6E
210 - $8A3A 810 - $1F42
220 - $55D4 820 - $D74F
230 - $9676 830 - $9260
240 - $F0EB 840 - $A329
250 - $9DB6 850 - $8B77
260 - $BAD2 860 - $814B
270 - $1B62 870 - $5639
280 - $E257 880 - $03BB
290 - $F428 890 - $AB41
300 - $1F9A 900 - $ECF2
310 - $2280 910 - $D739
320 - $8FD2 920 - $667C
330 - $7A7E 930 - $8898
340 - $BC7B 940 - $ECD9
350 - $F072 950 - $6A77
360 - $862F 960 - $5F5D
370 - $8E80 970 - $4CB2
380 - $B0A8 980 - $E1F6
390 - $D8CE 990 - $EB80
400 - $A518 1000 - $9B21
410 - $88A0 1010 - $00F2
420 - $FC73 1020 - $252B
430 - $B505 1030 - $2FAB
440 - $E99F 1040 - $E65F
450 - $3193 1050 - $BA78
460 - $ACB7 1060 - $DC02
470 - $3083 1070 - $EA3E
480 - $6219 1080 - $FE00
490 - $6A4D 1090 - $ED1B
500 - $599B 1100 - $6153
510 - $5FCB 1110 - $2105
520 - $887A 1120 - $2182
530 - $29CD 1130 - $4187
540 - $978C 1140 - $6AB1
550 - $8ADE 1150 - $D701
560 - $81BD 1160 - $9527
570 - $82BB 1170 - $E33F
580 - $7226 1180 - $35A0
590 - $1586 1190 - $A863
600 - $E10A 1200 - $171D
610 - $9A09 1210 - $8185
620 - $71E0 1220 - $2B50
630 - $80D5 1230 - $67F5
640 - $36B1 1240 - $F198
650 - $F55E 1250 - $A72C
660 - $B282 1260 - $DC23
670 - $482C 1270 - $2FA0
680 - $B3F2

```

Psychedelic Checksums

```

10 - $BADD 50 - $C899
20 - $9B13 60 - $FF65
30 - $4D3B 70 - $A3BF
40 - $AD92 80 - $A900

```

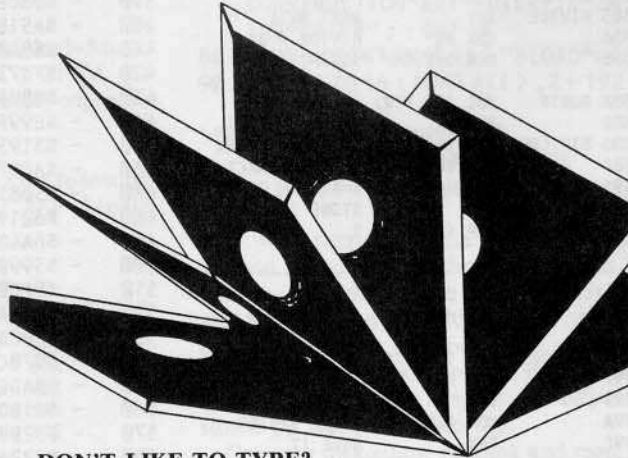
If you took all the old Hardcores...
 Tore off the fancy covers...
 Deleted all the editorial material,
 out-of-date interviews and letters...
 Updated the remaining material, and THEN
 included the MOST RECENT and MOST COMPLETE LIST of
 parameters
 for the major bit-copy programs...
 And packed it all into a single volume...
 you'd have the core of Hardcore Computing.

We Call It:

THE BEST OF HARDCORE COMPUTING

- Not just a reprint of old data
- Not a collection of unrelated articles

But an updated, rewritten, improved consolidation of "The Best" of Hardcore Computing. From the first article and program to the last, discover the heart and soul of your Apple with DiskEdit, DiskView, Super IOB (Hardcore's OWN deprotecting and copy program), Parameters, games and more...



DON'T LIKE TO TYPE?

Order now, and receive *The Best Of Hardcore Computing PLUS the Program Disk for \$19.95.* (Washington State residents add 7.8% sales tax. Foreign orders add 20% shipping & handling. U.S. funds only).

Please send me:

- The Best Of Hardcore Computing AND Program Disk.....\$19.95
- The Best Of Hardcore Computing..... \$14.95
- Program disk only.....\$9.50

Name _____

Address _____

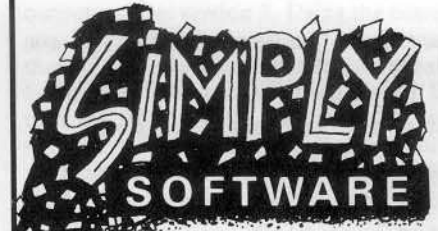
City _____ St _____ Zip _____

Country _____

Visa/MC _____ Exp _____

Signature _____

Send check or money order to:
 Hardcore COMPUTIST, Box 44549B, Tacoma, WA 98444



**DISCOUNT
 SPECIALS ON
 APPLE SOFTWARE
 ALL PRICES 30% OFF**

LIST \$25.00 SPECIAL \$17.50

Comp. Stocks & Bonds
 Computer Acquire
 Guns of Fort Defiance
 Multi Disk Catalog II

LIST \$39.95 SPECIAL \$27.97

Adven. to Atlantis Starcross
 Comp. Baseball (SSI) Strange Odyssey
 Copy II Plus Three Mile Island
 Legacy of Voodoo Castle
 Lylgamyn

LIST \$49.95 SPECIAL \$41.97

Appleworld Invoicing (AST)
 CDEX Train. Visicalc Report Card

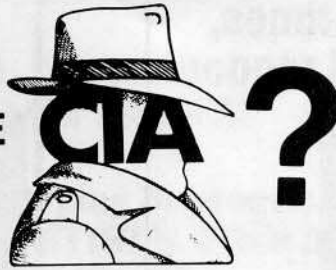
	LIST	SPECIAL
Applesoft Compiler +	\$ 99.95	\$ 69.97
Aplwrtr 80col. Boot	\$ 19.00	\$ 13.30
The Artist	\$ 79.95	\$ 55.97
Bank Street Speller	\$ 69.95	\$ 48.97
Beneath Apple Manor	\$ 29.95	\$ 20.97
Bez-Off	\$ 32.95	\$ 23.07
Bug Byter	\$ 47.50	\$ 33.25
Context Connector	\$180.00	\$126.00
Crush, Crum. & Chomp	\$ 30.00	\$ 21.00
Data Capture 4.0	\$ 65.00	\$ 45.50
Data Factory 5.0	\$300.00	\$210.00
dBase II	\$700.00	\$490.00
Empire I	\$ 32.95	\$ 23.07
Friday!	\$295.00	\$206.50
Hi-Res Secrets	\$129.95	\$ 90.97
Hist. Plot (Andent)	\$ 29.00	\$ 20.30
Knoware	\$ 95.00	\$ 66.50
Letter Perfect	\$149.95	\$104.97
Locksmith 5.0	\$ 99.95	\$ 69.97
Merlin	\$ 64.95	\$ 45.47
Micropainter	\$ 34.95	\$ 24.47
Personal Investor	\$145.00	\$101.50
Quant. Comparisons	\$ 27.95	\$ 19.57
Rastor Blastor	\$ 29.95	\$ 20.97
Real Estate Ana.	\$195.00	\$136.50
Robotron	\$ 34.95	\$ 24.47
Screenwriter II	\$129.95	\$ 90.97
Sol. Checkbook Reg.	\$ 20.00	\$ 14.00
Sorcerer	\$ 49.95	\$ 34.97
Spy Strikes Back	\$ 19.95	\$ 13.97
Study Prog. for SAT	\$ 89.95	\$ 62.97
Supr' Terminal (M&R)	\$349.95	\$244.97
Target Planner Calc	\$ 99.00	\$ 69.30
Terrapin Logo	\$149.95	\$104.97
Constitution Tutor	\$ 20.00	\$ 21.00
Visifile	\$250.00	\$175.00
Visischedule	\$300.00	\$210.00
Wordstar	\$395.00	\$276.00

PLEASE make check or M.O. payable to:

Simply Software Inc.
 P.O. Box 36068
 Kansas City, Missouri 64111

Add \$3.00 shipping, Missouri residents add 5 5/8% sales tax. Allow 4-6 weeks for delivery.

**WHO
LIKES
THE**



**HERE ARE JUST A
FEW OF THE MANY**

- "an essential part of the Apple-user's repertoire" - APPLE USER
- "a valuable buy ... manual is practically worth having on its own" - WASHINGTON APPLE PI"
- "the folks at Golden Delicious should be commended ... worth waiting for" - HARDCORE COMPUTIST
- "multifaceted" - NIBBLE
- "the most comprehensive disk accessor I have ever come across" - A.B., VERNON, CANADA
- "its ability to unlock other programs will greatly help me" - DR. B.P., SAN FRANCISCO, CALIFORNIA
- "an excellent set of programs ... just great - and good value too" - E.A.S., MILTON KEYNES, ENGLAND
- "very, very educational ... great manual ... it is FANTASTIC!!!!!!!" - J.C., TUCSON, ARIZONA
- "a very enlightening piece of software/book ... top of my list for good buys" - H.S., BLAINE, MINNESOTA
- "I like yours the BEST" - R.R., CHICAGO

Why all the excitement about the CIA (confidential information advisors)? Probably because it is the ONLY set of utilities (5 in all) which enable even a beginner to investigate, edit, locate, list, trace, rescue, translate, patch, repair, verify, examine, protect, unprotect, analyse, encrypt, and decrypt programs on normal AND protected disks. You also get the "CIA Files", a 65000+ word book which contains detailed instructions for using the C.I.A. plus easy-to-follow, hand-holding tutorials about patching, repair, formatting, encoding, protection, and numerous other disk topics. You'll find plenty of material here which has never before appeared in print. PROGRAMS NOT COPY PROTECTED

To put the 5 C.I.A. utilities, plus book, on the trail of your Apple II+/IIe (R) disks, send \$65.00 by check or money order to:

GOLDEN DELICIOUS SOFTWARE LTD.
350 Fifth Avenue, Suite 3308, Dept H, New York, New York 10118

**Apple II, II+, IIe,
Franklin users:**

- Do you have problems backing-up your copy-protected programs?
- Do you lack parameters for your copy programs?
- Are you looking for programs that you can AFFORD?
- Are you hesitating to upgrade your equipment due to expensive prices quoted in other ads?

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

Send \$1.00 U.S. funds to:

**Reliant
P.O. Box 33610
Sheungwan, Hong Kong**

IMPORTANT: We have over 600 PC name-brand programs and various hardware offers. Programs @ U.S. \$8.00/PC includes the disk and registered airmail handling.

*By Hackers
For Hackers*

- ELITE BOARD DOWNLOADS
- CRACKING TIPS
- PHREAKING SECTION
- GAME CHEATS
- PARMS
- PROGRAMS
- INTERVIEWS

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

The BOOT-LEGGER MAGAZINE

Subscribe Now!

Send 25 Bucks for a 1-Year Subscription to:
**THE BOOT LEGGER, 3310 Holland Loop Road,
Cave Junction, Oregon 97523**

**Want
To subscribe?**

See Page 5 for subscription information and order form.

Know where your head is, at all times,
with **TRAK STAR** constant digital readout



- Saves copying time
- For nibble programs

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

FREE INTRODUCTORY BONUS with purchase of Trak Star

- Trak Star disk contains patching software.
- Simple-to-operate, menu-driven Trak Star software automatically repairs a bad track without requiring technical expertise.

99⁹⁵

Plus \$3 shipping and handling charge

Foreign airmail & handling \$8.00.

Adapter required for 2-drive systems: \$12

Documentation only: \$3

Refundable with purchase of Trak Star
Personal checks, M.O.,
Visa and Mastercard

Midwest



Phone 913 676-7242

Microsystems

9071 Metcalf / Suite 124
Overland Park, KS 66212

Easy-View™

Disk File
Work
Station



- Stores 100 Disks, Dust Free
- 25 Disk Titles Clearly Visible
- Fast, Easy Access
- Top Flips Back, Locks Upright
- Closed Files are Stackable

\$9⁹⁵

Add \$2.50
Postage & Handling
Cash, check or M.O. No C.O.D.'s

RULE ONE 42 Oliver Street Dept. H
Newark, N.J. 07105

BACK UP YOUR DISKS



NOW AVAILABLE
AT YOUR LOCAL
COMPUTER STORE!

\$79⁹⁵

EDD runs on Apple II, II plus (including most compatibles), IIe, and III (in emulation mode), with one or two 3.3 disk drives.

EDD is the most powerful copy program available for backing up your protected Apple software. Since EDD has been preset to copy a broad range of copy-protections, many disks can be copied easily, without changing messy parameters. Even though you rarely need to change them, each parameter is fully described in the operating manual. Unlike the copycards, which only copy single load programs, EDD backs up entire disks. Thus, not only copying single load, but, multi disk access programs as well. We feel on an average, EDD can back up many more protected disks than all other copy programs or copycards put together.

ESSENTIAL DATA DUPLICATOR III™

- Automatically copies most protections.
- Rarely needs parameter changing
- Average duplication time 2½ minutes
- Accurately finds "self-sync" bytes and their lengths
- Can copy ¼ and ¾ tracks
- Updated program lists available
- Unlike copycards, EDD backs up entire disks, not just what's in memory

To order direct; send \$79.95 plus \$2 shipping (\$5 foreign), California residents add 6%. Mastercard/Visa accepted. Prepayment required.

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

Do you need BACK ISSUES?

Are you tired of typing in programs that are available on disk from Hardcore COMPUTIST's PROGRAM LIBRARY?

If you're reading Hardcore COMPUTIST for the first time,
don't miss out on past issues.
And, take advantage of our SPECIAL OFFER.

Please send me the back issues and/or library disks I have checked below:

- Hardcore COMPUTIST 1.....\$3.50
- Hardcore COMPUTIST 4.....\$3.50
- Hardcore COMPUTIST 6*.....\$3.50
- Hardcore COMPUTIST 8*.....\$3.50
- Hardcore COMPUTIST 9*.....\$3.50
- Hardcore COMPUTIST 10.....\$3.50
- Hardcore COMPUTIST 11.....\$3.50
- CORE 1 Graphics.....\$4.50
- CORE 2 Utilities.....\$4.50
- CORE 3 Games.....\$4.50

* Limited supplies

SPECIAL OFFER!

Order these magazine/disk combinations
and SAVE.

- CORE 1,2,3.....\$10.00
- CORE 1, Hardcore COMPUTIST 1,
Library Disk #1.....\$19.95

HC12

Name _____

Address _____

City _____ St _____ Zip _____

Country _____

VISA/MC _____ Exp _____

Signature _____

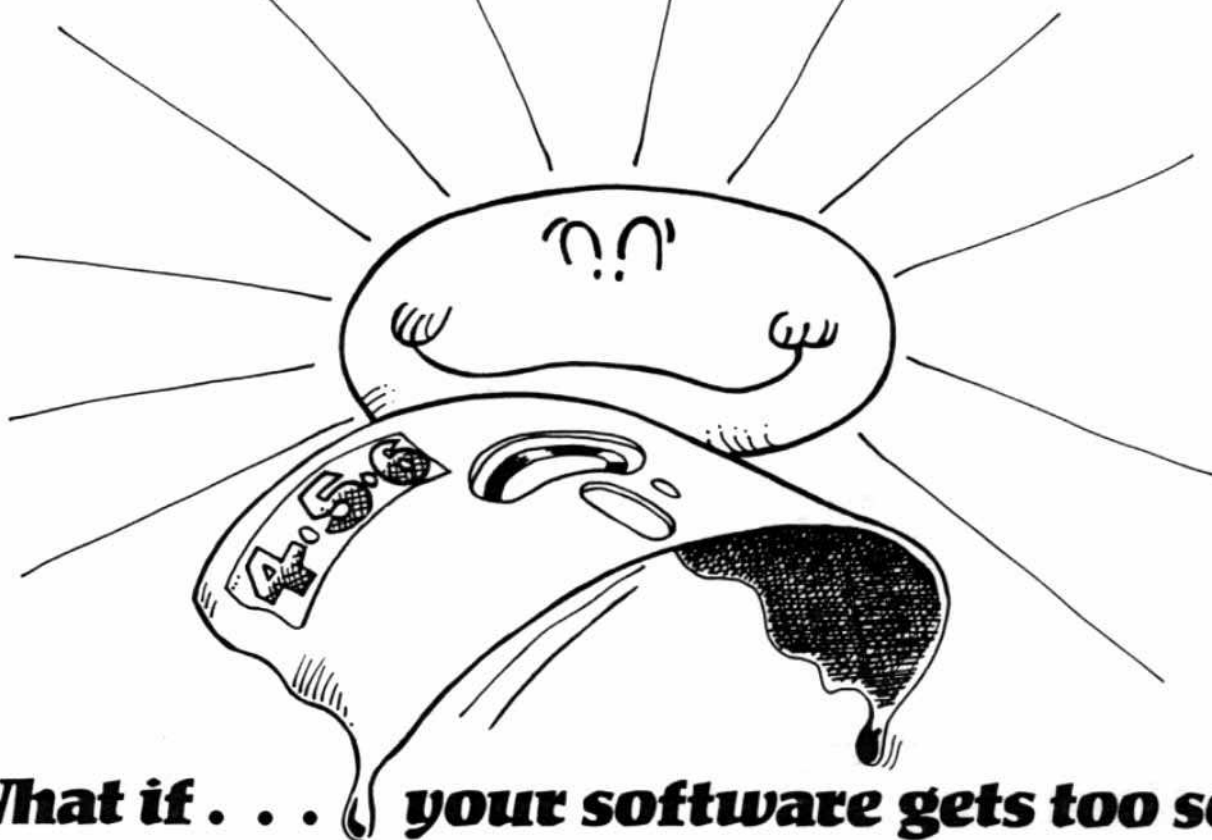
Send check or money order to:

Hardcore COMPUTIST
Back Issues/Library Disk Offer
P.O. Box 44549B
Tacoma, WA 98444

Washington state residents add 7.8% sales tax. Foreign orders add 20% shipping and handling. U.S. funds only. Most orders shipped UPS. Please use street address.

- Library Disk #8.....\$9.95
- Hardcore COMPUTIST 11:
Ultimaker
Ultimapper
UltimaIII Controller
Sensible Speller
- Library Disk #7.....\$9.95
- Hardcore COMPUTIST 10:
Sensible Speller
Arcade Machine Controller
Controller Saver
AppEar
Crunchlist II
Minit Man Controller
- Library Disk #6.....\$9.95
- Hardcore COMPUTIST 9:
Super IOB and related Controllers
CORE Word Search Generator
Sensible Speller Loader & Saver
- Library Disk #5.....\$9.95
- Hardcore COMPUTIST 7:
Corefiler
Disk Directory Designer
Hardcore COMPUTIST 8:
Corefiler Formatter
- Library Disk #4.....\$9.95
- Hardcore COMPUTIST 6:
Modified ROMs
Crunchlist
Crucial Code Finder
- Library Disk #3.....\$9.95
- CORE Games issue:
Destructive Forces
Dragon Dungeon
- Library Disk #2.....\$19.95
- CORE Utilities issue
Hardcore COMPUTIST 4
- Library Disk #1.....\$19.95
- CORE Graphics issue
Hardcore COMPUTIST 1
- Disk Control.....\$15.00
- Disk Edit
IOB
Menu
Disk View

NOTE: Most Library Disks need accompanying magazine for documentation. Those back issues not listed are no longer available.



What if . . . your software gets too soft?

You need software insurance.

Diskettes are fragile, and when a protected program is damaged, the results are expensive and inconvenient. If you have a backup diskette, though, you can have your Apple, IBM or compatible computer back on line within seconds . . . affordably. That's software insurance.

Copy II Plus (Apple][,][Plus, //e)

This is the most widely used backup program for the Apple. Rated as "one of the best software buys of the year" by InCider magazine, its simple menu puts nearly every disk command at your fingertips. The manual, with more than 70 pages, describes protection schemes, and our **Backup Book™** lists simple instructions for backing up over 300 popular programs. A new version is now available that is easier to use and more powerful than before. Best of all, Copy II Plus is still only \$39.95.

WildCard 2 (Apple][,][Plus, //e)

Designed by us and produced by Eastside Software, WildCard 2 is the easiest-to-use, most reliable card available. Making backups of your total load software can be as easy as pressing the button, inserting a blank disk and hitting the return key twice. WildCard 2 copies 48K, 64K and 128K software, and, unlike other cards, is always ready to go. No preloading software into the card or special, preformatted diskettes are required. Your backups can be run with or without the card in place and can be transferred to hard disks. \$139.95 complete.

Copy II PC (IBM)

This is **THE** disk backup program for the IBM PC and PC/XT that backs up almost anything. Others may make similar claims, but in reality, nothing out performs Copy II PC . . . at any price. Copy II PC even includes a disk speed check and is another "best buy" at only \$39.95.

We are the backup professionals. Instead of diluting our efforts in creating a wide variety of programs, we specialize in offering the very best in backup products. So, protect your software investment, before things get too hot.



CENTRAL POINT
Software, Inc.

The Backup Professionals

To order, call 503/244-5782, 8:00-5:30 Mon.-Fri., or send your order to: Central Point Software, 9700 SW Capitol Hwy, Suite 100, Portland, OR 97219. Prepayment is required. Please include \$3 for shipping and handling (\$8 outside U.S. or Canada).

Important Notice: These products are provided for the purpose of enabling you to make archival copies only. Under the Copyright Law, you, as the owner of a computer program, are entitled to make a new copy for archival purposes only, and these products will enable you to do so.

These products are supplied for no other purpose and you are not permitted to utilize them for any use, other than that specified.