For The Serious User Of Apple][Computers

DISK #17

Hardcore COMPUTIST

Issue No. 17 $2.50

A Tutorial For:
Disk Inspection And
The Use of Super IOB
Pg. 9

The Print Shop:
A Softkey
Pg. 13

The Lone
Catalog Arranger
v1.0: Part Two
Pg. 20

The Graphic Grabber
(For The Print Shop)
Pg. 16

Hardcore COMPUTIST
PO Box 110846-T
Tacoma, WA 98411

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

TURBO CHARGE YOUR IIe
AND EXPAND APPLE WORKS TO 101K

MEMORYMASTER IIe™ 128K RAM CARD

As an Apple user, you already know all the things your IIe can do. Now Applied Engineering expands that list with MemoryMaster IIe™.

With the MemoryMaster IIe, you'll have up to 192K RAM at your command. Designed expressly for the auxiliary slot in the IIe, the MemoryMaster IIe provides an 80 column video display and up to 128K of memory using just one slot. But the MemoryMaster IIe differs from other large memory cards in one important way, both the 64K and 128K configurations are TOTALLY compatible with software written for the Apple IIe 80 column and extended 80 column cards. In fact, it's not possible to write a program that uses the smaller cards from Apple that won't work with the MemoryMaster IIe.

But the MemoryMaster IIe is not just another piece of unsupported hardware. Each MemoryMaster IIe includes a multi-programming environment program which will enable you to have three different programs ready to run at any moment.

Many of today's software packages for data-base management, word processing, business applications and spread sheets either require or are enhanced by the MemoryMaster IIe. And more and more programs are using double high resolution graphics for which the MemoryMaster IIe is required.

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 your chips.

- Expands your Apple IIe to 192K memory.
- Provides an 80 column text display.
- TOTALLY compatible with all Apple IIe 80 column and extended 80 column card software—there are NO exceptions.
- Available in 64K and 128K configurations.
- The 64K configuration is USER upgradeable to 128K.
- Can be used as a solid state disk drive to make your programs run over 20 times faster (the 64K configuration will act as half a drive).
- Bank select LED's for each 64K bank.
- Permits your IIe to use double high resolution graphics.
- Uses the same commands as the Apple 80 column card.
- Plugs into the auxiliary slot in the Apple IIe.
- The 64K MemoryMaster IIe will automatically expand VisiCalc to 95K storage, in 80 columns! The 128K MemoryMaster IIe will expand VisiCalc to 141K storage with optional pre-boot disk.
- The 64K MemoryMaster IIe will automatically expand Apple Works to 55K storage, in 80 columns! The 128K MemoryMaster IIe will expand Apple Works to 101K storage with optional expand disk.
- Fully PASCAL and CP/M compatible.
- Lowest power consuming 128K card available.
- Complete documentation included.
- PRO-DOS will automatically use the MemoryMaster IIe as a high speed disk drive.
- Three year warranty.

Low Cost Options.

Apple Works Expand™
Although Apple Works is compatible with both a 64K and 128K MemoryMaster IIe, Apple Works only "sees" it's first 64K bank. Our Apple Works expand program will make a modification to Apple Works that simply lets it know you've got more memory, giving you 101K work space.

$29

Ram Drive IIe™
Sure-fire wait-reduction! Time was, you had to wait for your disk drives. But Ram Drive IIe has changed all that, giving you a large, extremely fast, solid-state disk drive, much faster than floppies or hard disks.

Ram Drive IIe works with either the 64K or 128K MemoryMaster IIe to give you a high speed solid-state disk drive. The Ram Drive IIe software features audio-visual access indicators, easy setup for turnkey operation, and easy menu driven documentation. The program can be modified and is copyable. If you have a 64K MemoryMaster, Ram Drive IIe will act as half a disk drive. If you have a 128K MemoryMaster, Ram Drive IIe will act as a full disk drive. Ram Drive IIe is compatible with APPLESOFT, DOS3.3, PRO-DOS, and PASCAL. Disk also includes a high speed RAM disk copying program. Ram Drive is another disk drive only 20 times faster.

$29

CP/M Ram Drive IIe
CP/M Ram Drive IIe is just like the Ram Drive IIe above, only for CP/M. CP/M Ram Drive IIe runs on any Z-80 card that runs standard CP/M i.e. Applied Engineering Z-80 Plus or Microsoft Soft Card. CP/M Ram Drive will dramatically speed up the operation of most CP/M software because CP/M normally goes to disk fairly often. Fast acting software like dBase II and Wordstar become virtually instantaneous when used with CP/M Ram Drive.

$29

VC IIe Expander
VC IIe Expander will give owners of VisiCalc IIe and the 128K MemoryMaster a total of 141K work space. This disk will pre-boot VC IIe in 1.5 seconds.

$29

Advanced VC IIe Expander
Advanced VC IIe Expander is just like VC IIe Expand only it is designed for Advanced VC IIe. Your work space will be increased to 131K. This disk will pre-boot in 1.5 seconds.

$29

MEMORYMASTER IIe with 128K $249
Upgradeable MEMORYMASTER IIe with 64K $169
A. E. Extended 80 Column Card with 64K $139

Texas Residents Add 5% Sales Tax
Add $10.00 If Outside U.S.A.

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.

Public Domain Software

From the Computer Learning Center's Library

★SPECIAL OFFERS★

(Limited time only)
Order now!!!

S1 EAMON Collectors Offer

Every EAMON scenario and all EAMON utilities. Includes E1 through E76, U1 through U4, D5, D6. *82 EAMON volumes for only $250.*

EAMON adventures are fantasy role-playing games. Your character, which you create using the Master Disk, wanders through a fantasy world full of dangers and rewards. You are the master of your own fate; the course of the adventure is of your own making. Your decision to bargain, steal, fight or run affects the scenario and the action and eventual outcome is rarely the same.

S2 EAMON Introductory Offer

You get the EAMON Master with the Beginner's Cave, The Orb of Polaris, Operation Crab Key, The Feast of Carroll, EAMON Utility 1 and the Dungeon Designer version 6.0. Includes volumes E1, E33, E44, E45, U1, D6. *A popular sampling of 6 EAMON volumes for only $20.*

S3 Art & Graphic

Every Art & Graphic volume. Includes volumes 4 through 13, 93, 94. *12 volumes for only $40.*

S4 Business & Finance

All of the very popular Business & Finance volumes. Includes: 18 through 25. *8 volumes for only $28.*

S5 Games

Golf, hockey, ping-pong, poker, Hi lo, blackjack, cribbage, craps, chess, bowling, parachute, pinball, dragon maze, star trek, fizz bin, tank, keno, war lords, roulette, lunar lander and more. Includes volumes 36 through 57. *22 game-packed volumes for only $75.*

S6 Utilities

Disk utilities, printer programs, assemblers, and disassemblers. It's all here. Includes volumes 74 through 87. *Get 14 Utility volumes for only $48.*

S7 Tutor and Math Combo

From BASIC programming to multiple linear regression, you'll find what you need with these tutor and math volumes. Includes: 1, 2, 3, 59 through 63. *An 8 volume selection from Apple Tutor and Math & Statistics for only $28.*

S8 Chemistry and Math Combo

A choice selection for the scientifically inclined. Includes one Chemistry & Biology and five Math & Statistics: volumes 26, 59 through 63. *All six volumes for only $20.*

S9 Public Domain Software (PDS) Introductory Offer

Includes one volume from each category: Apple Tutor, Art & Graphic, Astronomy, Aviation, Business & Finance, Chemistry & Biology, Demonstration, Education & School, Electronic & Radio, Food, Game, Hello & Menu, Math & Stats, Music & Sound, Passion, Pastime & Other, Unknown, Utility, Apple Bank, and Library. *22 volumes for only $75.*

There are over 175 volumes in the Computer Learning Center's Public Domain Library collection. All of these volumes will run on Apple][Plus computers and Apple-compatibles. Most will also run on the //e and //c. Each program in the collection has been donated to the public and has no copyrights attached. Therefore, each 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.

For more information on Public Domain Software from the Computer Learning Center(CLC), check the box on the order form to receive a copy of our NEW catalog.

Rush me the PDS items I have checked below. I understand that the volumes in each SPECIAL OFFERS package are packed on double-sided disks. (Offer good for a limited time only)

- ☐ $250 S1 EAMON Collectors Offer
- ☐ $20 S2 EAMON Introductory Offer
- ☐ $40 S3 Art & Graphic
- ☐ $28 S4 Business & Finance
- ☐ $75 S5 Games
- ☐ $48 S6 Utilities
- ☐ $28 S7 Tutor & Math Combo
- ☐ $20 S8 Chemistry & Math Combo
- ☐ $75 S9 PDS Introductory Offer

☐ Please send me the NEW PDS Catalog

Name _____

Address _____

City _____ St _____ Zip _____

Country _____ Phone _____

VISA/MC _____-_____-_____-_____ Exp _____

Signature _____

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

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

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

What Is a Softkey Anyway? Softkey is a term which we coined to describe a procedure that removes, or at least circumvents, 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 (on the DOS 3.3 System Master Disk).

Commands and Controls: In any softkey procedure, the actual keystroke commands which a reader is required to perform are set apart from normal text (typed in bold and indented). An example is:

PR#6

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

Control characters and shifted characters are indicated in commands as small superscripts.

6^{CTRL}P

To complete this command, you must first type the number 6, and then place one finger on the CTRL key and one finger on the P key. Shifted characters have a small SHIFT before them.

Requirements: Most of the programs and softkeys which appear in Hardcore COMPUTIST require one of the Apple][series of computers and at least one disk drive with DOS 3.3. Occasionally, some programs and procedures have special requirements: 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.

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

1) **Applesoft Program Editor** such as Global Program Line Editor (GPLE).
2) **Disk Editor** such as DiskEdit, ZAP from Bag of Tricks or Tricky Dick from The CIA.
3) **Disk Search Utility** such as The Inspector, The Tracer from The CIA or The CORE Disk Searcher.
4) **Assembler** such as the S-C Macro Assembler or Merlin/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.

You will also find COPYA, FID and MUFFIN on the DOS 3.3 System Master Disk useful.

Hardware Recommendations: Many softkey procedures require that the user be able to enter the Apple's system monitor during the execution of a copy-protected program. Check the following list to see what you will need:

Apple][Plus / Apple //e / Apple compatibles: 1) Place an Integer BASIC ROM card in one of the Apple's slots. 2) Use an non-maskable interrupt (NMI) card such as Replay or Wildcard.

Apple][Plus / Apple compatibles: 1) Install an F8 ROM with a modified RESET vector on the computer's motherboard. This method was detailed in Ernie Young's article, "Modified ROMS" (Hardcore COMPUTIST No. 6).

Apple //e / Apple //c: 1) Install a modifed CD ROM on the computer's motherboard. Don Lancaster's company (Synergistics, Box 809-AP, Thatcher, AZ 85552) sells the instructions necessary to make this modification but access to an EPROM burner is also required. Making this modification to an Apple //c will void its warranty, but gaining the ability to RESET into the monitor at will greatly enhances the capacity of the Apple owner to remove a disk's copy protection.

A 16K or larger RAM card, a printer, and a second disk drive are also recommended for Apple][or][+ owners.

Recommended Literature: The Apple][and][+ 's came bundled with an Apple Reference Manual, however this book is not included with the purchase of an Apple //e. You'll find that this book is necessary reference material. A DOS 3.3 manual is also recommended.

Other helpful books include: *Beneath Apple DOS*, Don Worth and Peter Leichner, Quality Software. $19.95, *Assembly Lines: The Book*, Roger Wagner, Softalk Books. $19.95, and *What's Where In The Apple*, William Lubert, Micro Ink. $24.95.

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

If you strike these keys:

```
10 HOME:REMCLEAR SCREEN
```

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

```
10  HOME : REM CLEAR SCREEN
```

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

```
10  DATA 67,45,54,52
```

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

```
10  DATA  67,45,54,52
```

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

All of this brings us to the Hardcore COMPUTIST LISTing format. In a BASIC LISTing, a space that *must* be keyed in is printed as a delta character (▲). All other spaces were put in by Applesoft and it doesn't matter whether you key them in or not.

There is one exception: If you want your checksums (see "Computing Checksums" section) to match up, you must not key in any spaces after a DATA command word unless they are marked by delta characters.

Keying In Hexdumps: Machine language programs are printed in Hardcore COMPUTIST as both source code and hexdumps. Only one of these formats need be keyed in to get a machine language program. Hexdumps are the shortest and easiest format to type in. First, you must enter the monitor:

```
CALL -151
```

Now key in the hexdump exactly as it appears in the magazine ignoring the four digit checksum at the end of each line (a "$" and four digits). If you type something incorrectly, a beep will alert you to retype that line.

When finished, return to BASIC with a:

```
E003G
```

Remember to BSAVE the program with the correct filename, address and length parameters as given in the article.

Keying In Source Code: The Source Code portion of a machine language program is provided only to better explain the program's operation. If you wish to key it in anyway, you will need an assembler. The S-C Assembler is used to generate all source code printed in Hardcore COMPUTIST. Without this assembler, you will have to translate pieces of the source code into something *your* assembler will understand. A table of S-C Assembler directives just for this purpose is printed on page 26 of this magazine. To complete the translation process, you need to understand the directives of your assembler and convert the directives used in the source code listing to directives similar to those used by your assembler.

Computing Checksums: Checksums are four digit hexadecimal numbers which verify whether or not you keyed in the program exactly as it was printed. There are two types of checksums: one created by the CHECKBIN program (for machine language programs) and the other created by the CHECKSOFT program (for BASIC programs). If the checksums these programs create on your computer match the checksums accompanying the article, then you keyed in the program correctly. If not, the program is incorrect at the line where the first checksum differs.

1) To compute CHECKSOFT checksums:

```
LOAD filename
BRUN CHECKSOFT
```

Get the checksums with

```
&
```

And correct the program where the checksums differ.

2) To compute CHECKBIN checksums:

```
CALL -151
BLOAD filename
```

Load the CHECKBIN program at an out of the way place and hook it up

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

Then type the starting address, a period and ending address of the file followed by a ^{CTRL}Y.

xxxx.xxxx^{CTRL}Y

And correct the lines at which checksums differ.

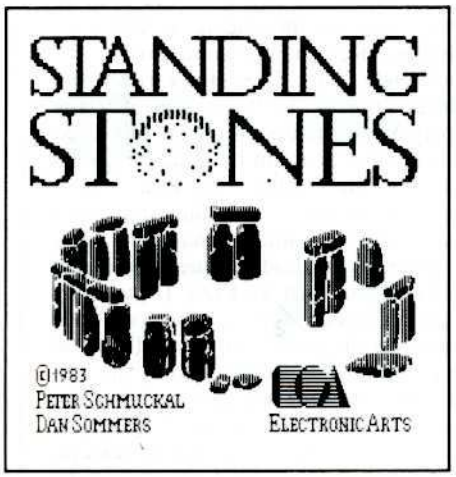
How-To's Of Hardcore

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

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

New readers are advised to read this page carefully to avoid making errors when following the softkeys or typing in the programs printed in this issue.

Page 7



Page 6

Issue No. 17

Hardcore COMPUTIST

Publisher/Editor: Charles R. Haight **Technical Editors:** Gary Peterson, Ray Darrah
Production & Graphics: Lynn Campos-Johnson **Circulation:** Michelle Frank
Business Manager: Valerie Robinson **Advertising:** (206) 474-5750 **Printing:** Grange Printing, Inc., Seattle, WA
Hardcore COMPUTIST is published monthly, except December, by SoftKey Publishing, 5233 S. Washington, Tacoma, WA 98409
Phone: (206) 474-5750

9 A Tutorial For: Disk Inspection And The Use of Super IOB
Confused about how to write a Super IOB controller? This article takes you step-by-step through the deprotection process and uses a generalized procedure to illustrate the creation of Super IOB controllers. You'll also discover how to use some other tools as disk inspection devices. *By Wes Felty.*

13 The Print Shop: A Softkey
This powerful utility program is finally opened up for inspection. Learn how a nibble count routine protects this high performance Broderbund program and why the original disk has two VTOC's. *By William Hinger & Albert ~~Stockton~~ Stockey.*

16 The Graphic Grabber For The Print Shop
The Print Shop's Graphic Editor lacks some of the more sophisticated features of other similar programs (fill, brushes, etc.). Now, with the Graphic Grabber, you'll be able to snatch a picture created with ***another*** editor (such as Koalapainter) and use it as a Print Shop graphic. *By Ray Darrah.*

CORE SECTION

20 The Lone Catalog Arranger v1.0
Part Two: The machine language program. Combine the program in this second installment of The Lone Catalog Arranger with the program which appeared in The Lone Catalog Arranger: Part One, HC No. 16. Together, these two programs provide a useful disk directory editor. Includes these commands: file order modification, renaming, deleting, and undeleting. *By Ray Darrah.*

24 BUGS!

25 Softkey For Crossword Magic
Create a completely copyable version of Crossword Magic with this boot code trace-type procedure. With a little finesse, you may even be able to put both sides of the original onto one side of a disk. *By Paul Selby.*

26 Note To Readers: S-C Macro Assembler Directives (Reprint from Issue No. 5)

DEPARTMENTS

4 INPUT

6 READERS' SOFTKEY & COPY EXCHANGE

Toppling The Standing Stones
By Steven Zupp

Deprotecting Beer Run
By Clay Harrell

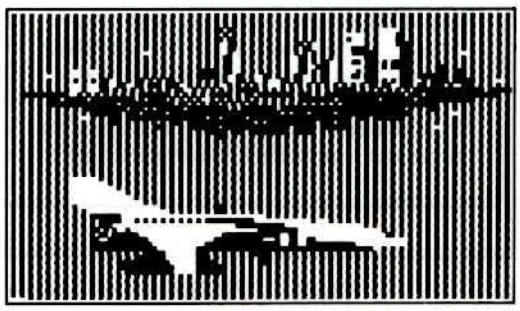
The Skyfox Softkey
By Marshall Strouse

Softkey For Random House Disks
By Mike Stafford

15 ADVENTURE TIPS

27 WHIZ KID

Part Two: DOS & The Disk Drive. A continuation of the topic which appeared in Hardcore COMPUTIST No. 15, this column examines the soft switches used by DOS to control the disk drive. *By Ray Darrah.*



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

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

Entire contents copyright 1985 by SoftKey Publishing. All rights reserved. Copying done for other than personal or internal reference (without express written permission from the publisher) is prohibited.

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

Apple usually refers to the Apple][or][Plus Computer, and is a trademark of Apple Computers, Inc.

SUBSCRIPTIONS: Rates: U.S. $25.00 for 12 issues, Canada $34.00, Mexico $39.00, Foreign (airmail) $60.00, Foreign (surface mail) $40.00. Direct inquiries to: Hardcore COMPUTIST, Subscription Department, PO Box 110846-T, Tacoma, WA 98411. Please include address label with correspondence.

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

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

Input

Superpilot on a Corvus?

I've never seen so much information that I've been looking for in a single publication, ever. I think it would be a great idea if you included copy protection info for the MacIntosh as well. I don't own a Mac but, undoubtedly, there are some interesting protection methods used on its software.

By the way, does anyone know how to put Apple's SuperPilot onto a CORVUS hard disk or any other hard disk?

Victor Tan
Republic of Singapore

Put a Ronco in Your Apple

Going all the way back to Hardcore COMPUTIST No. 2, I have a few corrections to the program called "Three-D Wall Draw":

1) Line 10 should read GOSUB 490
2) Lines 220 through 240 should read THEN 690...710...850, respectively
3) Add line 1055...POKE -16386,0...and GOSUB 1055...from lines 220 through 250 as well as lines 270, 290, 310 & 330
4) Line 250 may be deleted altogether since the Wall only "draws" while the key is depressed (watch out if you have a 'type-ahead' buffer).

Now, can anyone tell me how to eliminate the pivoting effect at the edge of the screen window?

Here is a hardware tip: Technically, the Apple][with its standard power supply is in virtual overload condition with a minimum of boards. Say, a RAM Card, a Drive Controller card and one other, possibly a RAM-DISK Board. You can fry eggs on the power supply as well.

The add-on fan that can be purchased for $40.00 to $50.00 is aesthetically pleasing and moves approximately 20 cfm of air through your Apple. The Ronco room air recycler that I purchased for $7.95 moves 40 to 50 cfm through my Apple. With a full complement of boards, including a Z80 and 256K of RAM-DISK, my power supply is cold to the touch after being left on all day, even during summer temperatures of 90+.

Take a piece of dense sponge foam 10x7x4" and with a magic marker draw a 5" circle on the largest surface about 4" from one end. Using a sharp knife, cut the circle out completely. Now make an L-shape by reducing the 4" dimension by 2/3 the original length and across the 5" hole. Next, remove material on the inside surface of the L but leave the edges intact. Finally, remove the sloping filter holder from atop the fan and discard it. Invert the fan and jam it into the hole. Place the entire assembly on top of your Apple so that it covers the louvres on the most convenient side (the left). You should not experience any more overheating problems. (I have installed an extra heavy-duty power supply in my Apple.)

On another topic: I find it very sinister that LS 5.0 copies itself without much trouble except that the resulting copy doesn't work. Even applying hints gained from Hardcore COMPUTIST has not helped. Also, the original LS 5.0 will make closer working copies of protected programs than will a liberated copy.

I have WAY-OUT by Sirius Software and my original LS 5.0 makes a copy (it goes through the entire boot process before the program detects that it is a clone and reboots). My *copy* of LS 5.0 won't make a duplicate that even comes close to booting. (By the way, can anyone tell me how to decipher the maze loading table in WAY-OUT located in the loader at 988F to 98AB, inclusive?)

I've requested info from Alpha Logic Systems regarding LPL (Locksmith Programming Language) and also info on how to fix the bug in Locksmith's Editor. Obviously, the author of the LS 5.0 review did not try out the program or else he's on the ALS payroll.

Problem 1) The Text Editor gobbles up characters on the following line when you are editing a line of an LPL file.

Problem 2) There is no glossary of programming statements but there are plenty of hints if you review the Newsletter with the various LPL files for backing-up specific programs.

Problem 3) There is a lack of program feedback. If an LPL file is changed, LS does not inform you of what is happening or even that anything has been changed.

Your magazine is ideal for us Apple Hackers. Please keep it up.

Norman F. Hogarth
Lilith, CA

Mr. Hogarth: Thank you very much for your input and tips, especially the one about the inexpensive fan. We have heard that Veg-O-Matics work quite admirably for notching out the back sides of disks (just kidding, folks).

We've forwarded your complaints about Locksmith 5.0 to AL Business Systems.

APT for Advanced Lode Runner

I recently wrote your staff with a question concerning an article in the Best of Hardcore Computing. I was pleasantly surprised when I received a reply in less than a week. The staff of Hardcore COMPUTIST is to be commended for producing a first class magazine. Keep up the good work!

A friend of mine discovered a trick which relieves some of the frustration of trying to play Advanced Lode Runner. Advanced Lode Runner does not have all the special features of the regular Lode Runner such as starting at any level, adding more men or skipping levels. Initialize a new data disk using the original Lode Runner, use a good nibble copier such as EDD III to copy tracks 3 through 6 from Advanced Lode Runner to the newly initialized disk, and the original Lode Runner thinks the new data disk contains levels created by you. You can now play all levels of Advanced Lode Runner using the special features available in the original Lode Runner by ending a game with CTRL-R. Press CTRL-E to enter edit, press P to select level, place the new data disk in the disk drive and press RETURN.

Thanks again for a great magazine and may you continue to prosper.

Warren W. Power
Davenport IA

Sensible Speller on the //c

A public thanks to your publication for a very worthwhile source of information on the ins and outs of disk protection. Keep up the fine work.

Your readers should know that the diskette offered by Marco Hunter in Hardcore COMPUTIST No. 9 is loaded with valuable material in its collection of programs and text files.

The //c owners should know some good news about the DOS 3.3 version of Sensible Speller, Rev. 4.1b, which will not boot on these machines. Once you deprotect the revision (and I assume any of the others), using Lamont Cranston's softkey in Issue No. 10, the program boots very rapidly and

Input cont...

runs on the //c. His procedure does not require a card and can be completed on either a //e or //c. Follow his directions along with the entry points specified in Doni G. Grande's article in Issue No. 11 and you have a program that will work on either your //e or //c...as is the case with me.

Rod Sobieski
Emporia KS

StickyBear Feedback

I would like to add to Mr. Jerry Caldwell's softkey on the Stickybear software series (featured in Issue No. 15).

In his article, Mr. Caldwell uses Stickybear BOP to demonstrate his softkey. It has been verified that Stickybear reads the protected sector into the buffer locations $300. He then boots up the original disk, enters the monitor and saves $300 in a safe place where it will not be overwritten when booting up a normal DOS 3.3 disk. He moved $300 to $9000-$90FF, booted up the DOS 3.3 disk and BSAVEd PROTECTED SECTOR, A$9000,L$100. Writing this protected sector back to the backup you've made can be a little shaky for beginners. Here's where the little adjustment comes in:

Instead of moving $300-$3FF to $9000, move it to $6000 (it's just a nice number to tinker with).

6000<300.3FFM C600G

Now boot a DOS 3.3 diskette without a HELLO program (remember that you copied $300-$3FF into $6000-$60FF). Instead of BSAVEing the protected sector, let's just write it directly to the backup disk that you made (since it's already in memory). This short write routine will do the job:

CALL-151

```
300:20 E3 03    ;THIS SETS UP DOS' RWTS
                 ROUTINE
303:4C D9 03
B7EB:00 02 0F   ;SET TRACK AND SECTOR
                 (TRACK $02 AND SECTOR $0F)
B7F0:00 60      ;BUFFER (STORAGE) IN
                 LO-HIGH ORDER ($6000)
B7F2: 00 00 02  ;THE '02' TELLS RWTS TO
                 WRITE A TRACK & SECTOR
```

Now you're ready to write! Insert the backup you've made of Stickybear and from the monitor type:

300G

You now have a perfectly COPYAble Stickybear BOP diskette.

I used this technique of writing a sector because when you BSAVE the protected sector, it creates a third sector which makes it impossible to write into a single sector without a sector editor. The third sector is due to a binary files address and length information. This technique writes a sector to disk as long as you specify the buffer address and the track and sector, of course.

Another tip: If you have an Apple][Plus and a language card and don't know how to enter the monitor, you can simulate an old 'F8' ROM by programming your language card. For slot 0:

CALL -151

```
C081 N C081     ;WRITE ENABLE RAM CARD
D000<D000.FFFFM ;COPY ALL ROMS INTO THE
                 CARD
FFFC:59 FF      ;SET TO JUMP INTO THE
                 MONITOR UPON RESET
C083 N C083     ;WRITE PROTECT RAM CARD
```

This works perfectly on the Stickybear series because it doesn't check or turn off the language card. If, however, a program turns off or 'sees' the language card, try this: Move the card to slot 1 or 2 and the following number will change accordingly:

```
For Slot 1:     C081 ->C091
                C083 ->C093
For Slot 2:     C081 ->C0A1
                C083 ->C0A3
```

NOTE: This method will not work all the time.

Hardcore COMPUTIST is already my favorite magazine (I just received my first issue). Keep up the great work and don't die out like Softalk did!

Randy Ramirez
Los Angeles CA

Educational Softkeys Please

I am writing to let you know how much I enjoy your magazine. It is very informative and useful. I have used a number of your softkeys to backup my software. However, there is one field that needs more attention: educational software. I'm a teacher and we need to be able to backup our software before little fingers sticky with peanut butter and jelly get ahold of it. The one that I need a softkey for most desperately is Broderbund's, "Print Shop".

Also, you mention using an Integer Basic ROM to RESET into the monitor. We have an old Apple][Plus with an Integer Basic Card (red switch on the back) installed. Is this the same card? If it is, will it work in an Apple //e? If so, how? If not, how can I interrupt into the monitor without a Wildcard?

George Cox
Beaufort Middle School
Beaufort, NC

Mr. Cox: *We agree that it is very important for teachers to be able to backup the software they use in the classroom. We always try to include educational softkeys in Hardcore COMPUTIST and we encourage our readers to submit softkeys for educational software.*

The Integer card that you own will allow you to RESET into the monitor. When you want to perform the RESET, just flip the switch on the card to the up position before hitting the RESET key. This card is compatible with the Apple //e's.

Needs Ultima][Help

Help! A previously published Ultima][article was mentioned in Issue No. 11, but I don't have any of the original magazines. Can anyone tell me how to access Lord British' monster shape tables?

I would like to voice my opinion on bringing other Apples into this magazine. NO WAY! All too many computer magazines contain information on other machines which is of no interest to me, since I can't perform the techniques on my Apple //e.

Also, could you republish a method to RESET into the monitor via "Moving Your RAM Card to Slot 1" from Issue No. 9? My subscription did not include that issue, and I would like to be able to put the "Locksmith 5.0 Fast Copy" into a BRUNable file.

Rob Klingsten
Howell, MI

Mr. Klingsten: *The original Ultima][article appeared in HC No. 4. It is still available as a back issue.*

To move your language card to a slot other than zero, see the above informative letter from Randy Ramirez.

Readers' Softkey & Copy Exchange

Toppling The Standing Stones

By Steven Zupp

Standing Stones
Electronic Arts
2755 Campus Drive
San Mateo, CA 94403
(415) 571-7171
$40.00

Requirements:
48K Apple with one disk drive
COPYA
Sector editing program
One blank disk

The protection for Standing Stones does not use any defenses against normal disk copiers at all. Instead it uses a tricky half-track check. This under normal conditions would be just fine. We could pull out our trusty nibble copier and copy it right off. Unfortunately, Standing Stones cleverly checks not only the half-track of 21.5, but also checks track 21 and 22. Since any attempt to copy the half-track would also destroy 21 and 22, and any attempt to copy 21 or 22 would destroy 21.5, we can throw our nibble copier out the window.

Fortunately, there's a remote possiblity that with a nibble copier that can do quarter tracks, you can copy just the right combination of tracks to do it.

Track 21 and 22 are not used for any purpose except the copy protection which we will later unhook. Track 22 is not COPYAable, but this dosn't matter.

Start up COPYA and copy from Standing Stones to the blank disk. After a while it should beep and say UNABLE TO READ. What has happened is that it has reached track 22. Ignore the message and remove the disks.

Now boot up your favorite sector editor. Insert the copy disk and read in track $11, sector $4. Change bytes $44 - $49 (20 D9 03 20 00 0D) all to EA. Now write it back to the disk. It should now boot perfectly and you can also make error free copies of this disk with any copier.

Looks like EOA has joined Sierra On-Line in leaving nice little messages. If necessary set your sector editor in the ASCII or character mode and read track $11, sector $2.

Deprotecting Beer Run

By Clay Harrell

Beer Run
Sirius Software, Inc.
10364 Rockingham
Sacramento, CA 95827
(916) 366-1195
$29.95

Requirements:
48K Apple][or][Plus
A way to Reset into the monitor
A 48K slave disk with no HELLO program
Beer Run from Sirius Software

Beer Run is a challenging game written by the same fellow that wrote the classic Apple game, Sneakers. Beer Run is a ladder and climb game that is very difficult to play, and challenges you to find the "Artesians".

Sirius has made Beer Run fairly difficult to backup by conventional means. After loading Beer Run and resetting into the monitor, a quick cruise down memory lane reveals that Beer Run is a large game, occupying most of the 48K of motherboard memory.

The large size of the game is a result of the large size of the shapes in the demos. These "shape tables" use much of memory yet are unnecessary for the actual game. If we neglect the demo, we can put Beer Run into an ordinary binary DOS file.

Even though Beer Run uses most of memory, we can still save it to a DOS file because "page switching" is used for its graphics. This technique of graphics involves writing on one of the hi-res pages, while showing the other. The newly drawn page is then shown and the other page is updated and drawn on. This process of "page switching" continues very quickly to provide us with very smooth and flicker-free graphics. Of course, hi-res pages one and two are used ($2000 to $5FFF); therefore, there can be no code there when the game starts. Memory from $400 to $1FFF and $6000 to $BEFF are used for the code and shape tables. Consequently, we must save these portions of memory.

The author uses a conventional starting address of $800. To discourage the Replay and Wildcard owners, Sirius has put in some disk access which checks for your original disk. This routine is easily defeated, but also deletes the demo at the same time.

We know what to do now, and all that's left is to put it in a binary BRUNable file. We must clear the way for a 48K slave disk boot which will wipe out $800 to $8FF and $9600 to $BFFF.

Step-By-Step

1) Boot the game and, when the prompt "Paddles (P) or Keyboard (K)" appears, reset into the monitor.
2) Remove the disk access during the game

```
BB00:4C 00 08
```

3) Compact the Beer Run Code and move page eight out of the way for a boot

```
2000<8000.BEFFM
8000<800.9FFM
```

4) Boot your 48K slave disk with no HELLO program.
5) After DOS is loaded, move $8000 to $80FF back down to $800

```
CALL -151
800<8000.80FFM
```

We now need a way to move memory from $2000 to $5FFF back up to $8000 to $BFFF after the file is loaded by DOS. We can't ask DOS to do this since DOS lives from $9D00 to $BFFF. If we did, DOS would be overwriting itself. Therefore, we must write a "memory move" routine.

The idea behind a memory move is simple: load a file into memory between $800 and $9600 and then jump to a machine language routine that copies memory from lower memory locations into higher locations.

When the routine is done, Beer Run will be back where it belongs and we can jump to the starting location at $800. The routine will look like this (take a look even if you don't know ASSEMBLY language):

```
5F00: LDX #$00     ;X = 0
5F02: LDA $2000,X  ;A = no's at ($2000+X)
5F05: STA $8000,X  ;loc ($8000+X) = A
5F08: INX          ;X = X + 1
5F09: BNE $5F02    ;if not 0, goto $5F02
5F0B: INC $5F04    ;incr pg to load from
5F0E: INC $5F07    ;incr pg to store to
5F11: LDA $5F04    ;A = pg to load from
5F14: CMP #$5F     ;compare to $5F
5F16: BNE $5F00    ;if not =, goto $5F00
```

6) A good place to put this routine would be at $5F00 since this page of memory is unused by Beer Run. Type the following:

5F00: A2 00 BD 00 20 9D 00 80
5F08: E8 D0 F7 EE 04 5F EE 07
5F10: 5F AD 04 5F C9 5F D0 E8

7) Now let's make the Beer Run code JuMP to this routine before starting up

7FD:4C 00 5F

Now we must determine how to save the code that is used across text page one. Notice that when you hit reset, the text screen has "garbage" on it. This code is not necessary, but Beer Run checks for it and, if it is not there, the game will not run. This is easily solved by moving (or filling) the text page with $60's which represents a Return From Subroutine in machine language. Whenever Beer Run tries to jump to a subroutine on the text page, it will immediately return back to the calling routine. We may add this short routine to our memory move. The code looks like this:

```
5F18: LDX #$00     ;X = 0
5F1A: LDA #$60     ;A = $60
5F1C: STA $0400,X ;put A at ($400 + X)
5F1F: INX          ;X = X + 1
5F20: BNE $5F1A    ;if not 0, goto $5F1A
5F22: INC $5F1E    ;incr pg to store to
5F25: LDA $5F1E    ;A = pg to store to
5F28: CMP #$08     ;compare to $08
5F2A: BNE $5F1A    ;if not = goto $5F18
5F2C: JMP $0800    ;start Beer Run
```

8) Enter the following hex code which is the same as the code above

5F18: A2 00 A9 60 9D 00 04 E8
5F20: D0 F8 EE 1E 5F AD 1E 5F
5F28: C9 08 D0 EE 4C 00 08

9) Now we may save our file to disk by typing

BSAVE BEER RUN,A$7FD,L$7803

Beer Run is now deprotected and BRUN-able from DOS 3.3.

IMPORTANT!

Note To Subscribers: Hardcore COMPUTIST does not send subscription renewal notification. This notice appears only on your mailing label.

Please note that the Post Office does not inform our subscription department when you file a change of address. You MUST send a change of address to Hardcore COMPUTIST on postal form 3576. Issues missed due to non-receipt of change of address can be purchased at the current back issue rate.

The Skyfox Softkey
By Marshall Strouse

Skyfox
Electronic Arts
2755 Campus Drive
San Mateo, CA 94403
(415) 571-7171
$40.00

Requirements:
Apple or compatible with 64K
One disk drive
Super IOB
A blank disk

The program Skyfox from Electronic Arts is one of the best programs I've ever seen because of its unique smooth motion graphics.

The protection used on SkyFox is equally good. It is essentially the same one that is used on many of Electronic Arts' newer releases such as One-On-One, Archon, Seven Cities of Gold and others so you may be able to use this procedure on other E.A. programs.

The disk can be copied with any good bit copier without generating any error codes, but due to a difficult-to-reproduce nibble pattern on track 6, such a copy will usually not work. As such, I was forced to boot code trace the disk.

The boot starts in the disk controller in one of your slots. Because the first byte of sector zero, on track zero is a $06 (instead of the usual $01), the controller ROM loads just about all of track zero into $800 up to $DFF. Next, this first boot loads tracks 1 and 2 into $A000 and $B000, respectively. This second code has an entry point of $A806.

This second code first checks for a language card and, if there is one, it continues executing. If there is no language card present, it turns on the text page and just hangs. If there is a 16K card, then tracks 3, 4, 7 and 8 are loaded into memory locations $4000, $5000, $6000, and $8000, respectively.

And Now, The Fun!

This code now does a checksum on its nibble count routine to make sure that it hasn't been changed. If the nibble read routine is unchanged, it continues on. If changed, the computer hangs again.

If the checksum of the nibble count is O.K., it is executed. If the correct nibbles are not found on track six, it again hangs!

If the nibbles on track 6 are correct, it then does a RAM de-encryption to make it runable machine code. By the way, all memory loaded on all E.A. games is encrypted.

We're not finished yet. If you were to pick a tank or low flight misson, it does a check on the disk again, which fails if the original isn't in the drive. In a condensed form, we must remove the protection by disabling the *two* nibble counts.

The Procedure

We will deprotect the disk by using a modified version of the Electronic Arts controller that was printed in Hardcore COMPUTIST No. 13. The controller (at the end of this article) has been modified to skip tracks 5, and 6 and to perform some sector edits. Because of the number of sector edits necessary, I did not use the standard Super IOB sector edit routine, but instead used a routine at lines 1210-1220 to POKE the necessary monitor commands into the Apple's input buffer. This routine is convenient for a Super controller that has to make numerous sector edits during the copy process. You do, however, have to calculate the proper locations in Super IOB's track/sector buffer where the changes are to be made.

1) Get out your Super IOB and use the controller at the end of this article to deprotect SkyFox. You will notice a pause of several seconds after the first range of tracks has been read. This pause is normal and occurs while the monitor commands are being POKEd into the input buffer.
2) That's all, folks.

What Happened?

First of all, track 2, sector 0 was modified in this manner:

a) Byte $AA was changed from a $A0 to a $60.

b) A short intercept routine was placed starting at Byte $D5. This code will end up at $BFD5 in memory.

```
BFD5-  68        PLA
BFD6-  A9 05     LDA #$05
BFD8-  20 00 BC  JSR $BC00
BFDB-  A2 06     LDX #$06
BFDD-  BD F9 BF  LDA $BFF9,X
BFE0-  9D E5 A2  STA $A2E5,X
BFE3-  CA        DEX
BFE4-  10 F7     BPL $BFDD
```

Exchange cont...

```
BFE6-  A9 4D     LDA #$4D
BFE8-  A2 FF     LDX #$FF
BFEA-  A0 0E     LDY #$0E
BFEC-  60        RTS
BFED-  EA        NOP
BFEE-  48        PHA
BFEF-  A5 43     LDA $43
BFF1-  C9 A0     CMP #$A0
BFF3-  F0 E0     BEQ $BFD5
BFF5-  48        PLA
BFF6-  6C 00 42  JMP ($0042)
BFF9-  00 52 AD 4D A6 A1 4D
```

An edit must now be done to make SkyFox jump to this routine. The place to put the jump to our routine was found on track 1, sector 5.

See the short listing below of the code preceding the location where our jump must be placed. With a little luck, you may find it on other E.A. programs as well.

```
$AAA3-  20 77 AA  JSR $AA77
$AAA6-  98        TYA
$AAA7-  48        PHA
$AAA8-  65 48     LDA $48
$AAAA-  20 B7 AA  JSR $AAB7
$AAAD-  85 48     STA $48
$AAAF-  68        PLA
$AAB0-  A8        TAY
$AAB1-  4C 3A AA  JMP $AA3A
$AAB4-  20 77 AA  JSR $AA77
$AAB7-  6C 42 00  JMP ($0042)
```

It is the indirect JMP at location $AAB7 we wish to change. The Super IOB controller changes bytes $B7-$B9 of track 1, sector 5 from $6C, $42, $00 to $4C, $EE, $BF.

Congratulations

You are all done and now the proud owner of an unprotected copy of SkyFox. As I finished up this article, I discovered that track 6 has a self-modifying nibble count, which means that it changes every time you read it.

SkyFox Controller

```
1000 REM SKYFOX CONTROLLER
1010 TK=0:ST=0:LT=34:CD=WR
1020 T1=TK:GOSUB 490
1025 IF TK>3 THEN RESTORE:GOSUB
     210
1030 GOSUB 430:GOSUB 100:ST=ST+
     1:IF ST<DOS THEN 1030
1035 IF TK=2 THEN GOSUB 210
1040 IF BF THEN 1060
1050 ST=0:TK=TK+1+2*(TK=4)
     :IF TK<LT THEN 1025
1060 GOSUB 490:TK=T1:ST=0:
     GOSUB 230
1062 IF TK=0 THEN A$="3CB7:4C^
     EE^BF^N^4A47:AA^N^4A51:AD^
     N^47AA:60^N^47D5:68^A9^
     05^20^00^BC^A2^06^BD^F 9^
     BF^9D^E5^A2^CA^10^F7":GOSUB
     1210
1064 IF TK=0 THEN A$=
     "47E6:A9^4D^A2^FF^A0^0E^60^
     EA^48^A5^43^C9^A0^F0^
     E0^68^6C^42^00^00^52^AD^4D^
     A6^A1^4D":GOSUB 1210
1070 GOSUB 430:GOSUB 100:ST=ST+
     1:IF ST<DOS THEN 1070
1080 ST=0:TK=TK+1+2*(TK=4)
     :IF BF=0 AND TK<LT THEN 1070
1090 IF TK<LT THEN 1020
1100 HOME:PRINT "DONE^WITH^COPY"
     :END
1200 REM  DO MONITOR COMMAND A$
1210 A$=A$+"^N^D9C6G":FOR A=1
     TO LEN(A$):POKE 511+A,ASC(
     MID$(A$,A,1))+128
1220 NEXT:POKE 72,0:CALL -144:
     RETURN
5000 DATA 213,187,207
```

Skyfox Controller Checksums

1000	- $356B	1064	- $F77F
1010	- $6344	1070	- $FF7A
1020	- $C418	1080	- $386D
1025	- $16E4	1090	- $C6B6
1030	- $00E5	1100	- $31E0
1035	- $BB74	1200	- $90D0
1040	- $F54F	1210	- $DE6C
1050	- $4A44	1220	- $EB7C
1060	- $B22E	5000	- $CD04
1062	- $6525		

Softkey For Random House Disks

By Mike Stafford

Peanuts Maze Marathon
Random House Inc.
400 Hahn Rd.
Westminster MD 21157

Requirements:
Apple][Plus or equivalent
A way to RESET into the monitor
One blank disk
Super IOB & Swap Controller (HC No. 14)

The Peanuts educational series by Random House offers an exceptional value for the money. Excellent quality programs featuring beautiful graphics fill the double-sided disks I have seen. In addition to Maze Marathon, I have also softkeyed Charlie Brown's ABC's using the same procedure, so it's probably safe to assume that most of the disks in the Peanuts series can be de-protected using the method described below.

Among other things, these disks employ altered address and data headers as a form of copy protection. Fortunately, we really don't have to get too deeply into this, as you can use the standard Super IOB program and the normal Swap Controller to accomplish this softkey. In fact, I the same RWTS remained installed for both disks. Here's the step-by-step method:

1) First boot up DOS (preferably a fast DOS, such as Diversi-DOS), enter the monitor and apply a patch that will permit the use of a binary greeting program

CALL-151
9E42:34

2) Initialize a blank disk

INIT STEX

3) Boot the Random House disk. As soon as it stops loading, RESET into the monitor by your favorite method.

4) Move the RWTS to a safe location

1900<B800.BFFFM

5) Insert your slave disk and boot it

C600G

6) Save the RWTS

BSAVE RWTS,A$1900,L$800

7) Install the Swap Controller in Super IOB and start the copy process by typing

RUN

8) Repeat Steps 1, 2 and 7 to make a copy of the back side of Peanuts Maze Marathon.

These disks can now be COPYA'd and cataloged. If you used a fast DOS, the loading time of the copies will have been reduced by two-thirds.

Want To Order Back Issues?
See pg. 30

A Tutorial For:

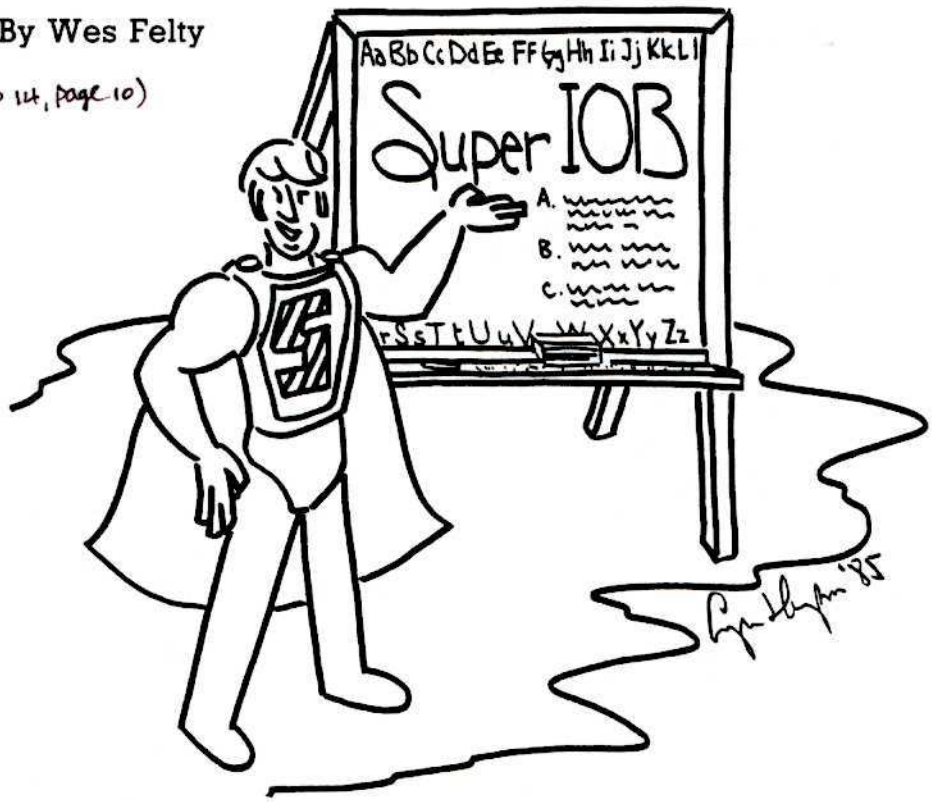
Disk Inspection And The Use of Super IOB

By Wes Felty

Requirements:
Super IOB v1.2
Disk Utility programs such as:
Locksmith 5.Ø
EDD
Bag Of Tricks
CIA
A quest for knowledge

Super IOB is great for deprotecting programs and for opening them up to investigation and/or modification. With this versatile program, deprotecting is usually as easy as typing in the IOB Controller and RUNing Super IOB. Using Super IOB version 1.2, (No 14, page 10) you don't even have to do any sector edits. Fortunately, all of the softkeys which have appeared in Hardcore COMPUTIST in past issues which use the modified COPYA and sector edit method can be modified to use Super IOB v1.2. However, creating your own controllers may pose a problem. Hardcore COMPUTIST frequently prints Super IOB Controllers, but often the author of the controller does not provide a lot of comment on how the controller was created. It's very easy to use a Super IOB Controller once you have one, but creating your own isn't so easy. To do so, you must determine precisely what protection scheme(s) are being used.

In this article, I will try to explain how Super IOB controllers are created. I will also touch on software "snooping" with tools like Locksmith 5.Ø, EDD, Bag of Tricks, CIA, and Super IOB.



The First Step

When investigating a disk, be very alert to everything you see and hear. Watch the screen while the disk is booting. If a prompt appears during the boot, then a reasonably normal DOS is probably in use. In this case, a RESET into the monitor sometimes allows the use of the modified DOS's CATALOG, LOAD and SAVE commands. Even if the text of the DOS commands has been changed, you can often view the disk's directory by calling the CATALOG routine directly with a CALL 42356 (A56EG from the monitor).

Be sure to listen carefully while a normal disk is booting. You should hear the gentle "woosh" sound at somewhat regular intervals as the disk drive read/write head moves to successive tracks. As you become familiar with this sound you will begin to distinguish this from the sound of a protected disk booting. I always compare the boot sound of my protected disks with that of normal disks. If they sound similar (even if no prompt appears on the screen), there is a good chance that a reasonably normal DOS is being used.

On some disks, you may hear noises that do not resemble the sounds produced by the boot of a normal disk at all. The sound that Essential Data Duplicator makes when it boots has been compared to an army of tapdancing cockroaches. This boot sound stands in marked contrast to that of a DOS 3.3 disk. Similarly, with Zardax you hear a suspiciously quick "woosha, woosha, woosha" sound as the drive head pops back and forth between tracks 4.5 and 5.5 looking for the copy protection information. And during the Skyfox boot, there is a fast "click, click, click," sound as the drive reads a small amount of information from eight to ten adjacent half tracks.

Strange boot sounds can tell you that even a super bit copier with extensive parameter settings will probably not work for certain tracks.

See The Tracks

If you take your disk investigation seriously, you'll want to mark your disk drive cam as outlined in Hardcore COMPUTIST No. 5. **WARNING!:** *Be careful when following this procedure! You can knock your drive head out of alignment doing this and you'll definitely invalidate any warranties still in effect.*

Who's What is Where

The next thing to determine is how many tracks need copying. Locksmith 5.0 and EDD both have good options for finding this information. Using either of them, it is easy to tell that Zaxxon has information only on tracks $00-$16. Next, you need to find out what kind of sector modifications on the nibble level have been made. Most bit copy programs such as Locksmith, Nibbles Away, and Copy][+ will aid you with this, but Bag of Tricks and CIA are my favorites.

Bag Of Tricks

When Bag Of Tricks works, it does all of the hard work for you. Therefore, I usually try it first. It will provide you with the address and data prologues, epilogues, and checksums for each sector as well as a raw nibble dump for each track. If any of the information is nonstandard, it is printed in inverse. Unfortunately, Bag of Tricks doesn't always work if the sectors have been too severely altered.

Confidential Information Advisors (CIA)

Up front, CIA gives you less information than Bag of Tricks but it does give a better indication of whether your interpretations are correct or not. When first beginning my investigation of a protected disk, I load Tricky Dick with the Linguist (option 2) into my machine. Next, I try to read several sectors with Tricky Dick. Track 0, sector 0 can usually be read with no changes since your computer needs to be able to read it in order to start loading DOS. You may even be able to read all of the DOS tracks (tracks 0-2). If you can read every sector of every track, then the disk may be copied with COPYA or just about any other normal whole disk copier. Unfortunately, this is seldom the case. More often you will get a drive error on at least one sector of one track. At this point you have to do manually what Bag of Tricks does for you automatically.

Assuming that the disk uses a somewhat normal sectoring structure, you must figure out how each sector has been corrupted. Enter the Linguist module, read a track such as $11, and look for the large groups of sync bytes. Usually they are FF's. If you do not see a large grouping of them, look instead for any other bunch of the same numbers (FE's, FD's, etc.).

Directly following the sync bytes you should find the D5 AA 96 address headers, eight address field bytes, DE AA address trailers, more sync bytes, D5 AA AD data headers, 342 bytes of encoded data and the DE AA data trailers. If all of these are correct, you might try another sector. If every sector seems to be correct, then the address field checksum or data field checksum is probably altered. Don't forget that by positioning the Linguist's cursor over the byte immediately following the address header (D5 AA 96), that it will display what track and sector the nibbles belong to.

Testing Your Findings

When you think you know what the changes are, return to Tricky Dick and use CTRL-S to change the headers and trailers to what you found. To ignore the address field checksum, change the first "Y" to an "N" and to ignore the data field checksum, change the second "Y" to an "N". Now, here is where Tricky Dick pays for itself. You can try to read the sectors now with the modifications you just made and see if your interpretations are correct. If you are correct, then you know what changes must be made to the standard Super IOB Controller. If you can't read the disk's sectors by changing the address and data headers and trailers and ignoring the checksums, then you might not be able to use Super IOB to copy the disk.

Even if Bag of Tricks worked and showed you what changes were made to the sector markers, you should use CIA also, if you have it. Change the markers and see if Tricky Dick can now read the disk sectors. This will confirm that the Super IOB controller will work. You can also use Tricky Dick in conjunction with the Tracer to verify that the entire disk was written in the format that you found with the Linguist.

If you don't have Bag of Tricks and/or CIA, you can still use any bit copy program nibble editor to search a raw nibble dump and look for the changes in the sector markers mentioned above. Those two programs just make the job much easier to confirm the changes.

"Hardcore COMPUTIST gave an involved softkey for this program (Zaxxon) in Issue No. 7 which showed how to locate the copy protection using boot code tracing...Instead, I can show you how to create a Super IOB controller to copy and edit Zaxxon all in one simple step."

An Actual Example

Let's look at Zaxxon. A softkey for this program appeared in Issue No. 7 and, although it showed how to locate the copy protection using boot code tracing, it lacked an explanation of how the necessary sector edits were determined. Instead, I can show you how to create a Super IOB Controller to copy and edit Zaxxon all in one simple step. I must admit though, that Clay Harrell's article on "Deprotecting Zaxxon" was one of the best articles published by Hardcore COMPUTIST. Because it was carefully written and detailed how the protection on Zaxxon was found, I was able to use the principles shown to fix my own copy of Zaxxon even though it was not one of those listed in the sector edit tables. In using Clay's article here, the most difficult and time consuming portion of the deprotection has already been accomplished. In fact, the normal softkey published in Hardcore COMPUTIST is, without doubt, the result of hours of hard work. It may not look like much when you see one or two simple looking sector edits for a softkey procedure (like with Locksmith 5.0 or PFS). But those simple looking changes are arrived at only after hours of investigation.

Back To Zaxxon

Using Locksmith 5.0's Quickscan routine or EDD's option 3 to examine the original Zaxxon disk, it was obvious that data is stored only on tracks $00-$16. EDD indicated that Zaxxon might have some information stored on the quarter or half tracks of $10-$13. I then watched my marked disk drive cam as I played Zaxxon. During the boot, the disk drive read tracks 0, 4, 11, 15, 14, 13, 12, 11, F, E, D, B, C, A, 9, and 8 before reaching the main menu. This told me that the information was only on the whole tracks and that the program most likely didn't use a Catalog structure but instead loaded information directly off of the disk. If it had a CATALOG, then it might have been found be on track 4.

Next, using Bag of Tricks' Trax program, I found that track 0 couldn't be analyzed and that the rest of the tracks had their address epilogues changed from "DE AA" to "CC AA". All of the other marks and checksums seemed to be normal. CIA confirmed that this was the main change. All of the tracks with information, even track 0, read properly. Using this information, I was then able to create a Super IOB Controller for Zaxxon. To create the controller, you will need to copy tracks $00-$16, 00-22 decimal, changing the address epilogue to "CC AA".

Unfortunately, the copy that results from the use of just this information will not work. A nibble counter in the program will prevent a copy like this one from running. But, you will at least have a COPYAble working disk to try making changes on. This is where Clay Harrell's article comes in. He gave the sector edits for three different versions of Zaxxon and I have listed the edits for my version below. Decimal values are shown in parenthesis. These are needed for the DATA statements of a Super IOB controller.

Zaxxon Sector Edits

Trk	Sect	Byte	From	To
$00	$07	$1F (31)	$A9	$4C (76)
$00	$07	$20 (32)	$00	$C0 (192)
$00	$07	$21 (33)	$85	$08 (8)
$00	$04	$4F (79)	$CC	$DE (277)
$00	$04	$50 (80)	$D0	$EA (234)
$00	$04	$51 (81)	$AE	$EA (234)

A Super IOB Revolution

The Super IOB program is essentially an RWTS controller program with a number of subroutines to allow modifications to the standard sector structure. The trick to writing new controllers is in knowing where in the standard controller to call the other subroutines.

Presented below is what I call the SUPER controller. It is the standard controller with a whole bunch of lines inserted between the usual standard controller lines. These extra lines call various subroutines and are well documented. To activate one of these lines, you must remove the REM from the beginning of the statement. The documentation for each line is listed directly after the line and is printed here in lowercase for increased readability. Because of the structure of the standard controller, modifications to the reading of the disk are made in lines 1Ø21 through 1Ø28 and modifications to the writing of a disk are made in lines 1Ø61 through 1Ø69.

A Change Of Address

To change address or data headers or trailers with the Super controller, you must do four things. First of all, you must put the modified values in DATA statements at the end of the controller; line 5ØØØ is a good place to start your data. Secondly, you must activate the RESTORE command of line 1Ø21 by removing the REM just before it. This avoids an ?OUT OF DATA ERROR and ensures that the modified headers or trailers are read at the beginning of each read cycle. Next, you must activate the appropriate GOSUB statement(s) in lines 1Ø22-1Ø24. Note that if more than one mark is changed, the DATA at the end of the controller must appear in the same order as the GOSUBs that read them. For example, if you change the ending marks with line 1Ø24, then you must include both pairs of end marks with the address marks coming first in the DATA statement. Finally, you must activate the GOSUB of line 1Ø65 to normalize all of the address and data marks to be written to the copy disk.

There are really three ending marks on each field, but DOS ignores the third mark. Super IOB does, too. DO NOT try to include the third one. If you do, the controller will not work.

Giving A Checksum The Cold Shoulder

To ignore the address field checksum, activate line 1Ø25. The activation of line 1Ø26 will invoke a data field checksum change. The normal value is zero and the new value should be the next data element. If either of these routines are activated, then you must activate line 1Ø65 so that the data read from the protected disk will be written normally.

I have brought back the "Ignore checksums and end marks" routine that was removed in the creation of Super IOB v1.2. If you wish the checksums and end marks to be ignored, activate both line 1Ø28 and line 1Ø69.

Sector Editing

Sector edits will be performed if line 1Ø67 is activated. Sector edits also require a DATA statement at the end of the controller. Be sure to have all other data appear before the sector edit data. Also, remember to activate the RESTORE of line 1Ø21 if you use any sector edits.

Sector editing with version 1.2 of Super IOB is much easier to do than with the older version but first you must specify how many sector edits will be performed. This is accomplished thorough a DATA statement like this:

5Ø1Ø DATA 2 CHANGES

The next data elements must then contain the decimal values of the track, sector, byte and new value (in that order) for each edit.

Modifying the Super Controller For Zaxxon

All of the routines to create a controller for Zaxxon already exist in the Super controller. All that is neccessary for us to do is to activate the proper line numbers. Basically, the Zaxxon controller has to do three things:

1) Copy tracks $ØØ-$16 (Ø-22)
2) During the read cycle, change the address epilogue bytes to "CC AA" or ignore the end marks
3) Perform the necessary sector edits

I tried using lines 1Ø24 and 1Ø65 to change the end marks, with "DATA 2Ø4, 17Ø, 222, 17Ø" for the end marks "CC, AA, DE, AA', and found that my controller didn't work. Therefore, I decided to use the "Ignore checksums and end marks" routine, lines 1Ø28 and 1Ø69, instead. This subroutine was dropped when Super IOB was updated to Version 1.2, but I still find it very useful.

To copy only tracks $ØØ through $16, I had to change the "LT = 35" in line 1Ø1Ø to read "LT = 23". Ignoring checksums and end marks is accomplished by removing the REM at the start of lines 1Ø28 and 1Ø69.

The resulting Super IOB Controller can now read the original Zaxxon disk, write the program back to a new disk in a deprotected form, and make the sector edits, all in one program. With all of the deactivated lines removed, the resulting controller would look like this:

Zaxxon Controller

```
1ØØØ REM ZAXXON CONTROLLER
1Ø1Ø TK = Ø : ST = Ø : LT = 23 : CD = WR :
     REM "set to 1st track & sector
```

Continued on next page

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 will include it in our Most Wanted List.

Hardcore COMPUTIST
Wanted List
PO Box 110846-K
Tacoma, WA 98411

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. **Critical Reading**
   Borg-Warner
4. **DB Master 4.0**
   Stoneware, Inc.
5. **Bookends**
   Sensible Software
6. **Visiblend**
   Micro Lab
7. **Dollars And Sense**
   Monogram
8. **Lifesaver**
   Micro Lab
9. **Catalyst**
   Quark, Inc.
10. **Gutenburg Jr. & Sr.**
    Micromation LTD
11. **Prime Plotter**
    Primesoft Corp.
12. **SSI Wargame Series**
    Strategic Simulations, Inc.
13. **Sargon III**
    Hayden
14. **Zardax**
    Computer Solutions
15. **List Handler**
    Silicon Valley Systems
16. **Milliken Math Series (NEW)**
    Milliken Publishing
17. **College Entrance Exam Prep**
    Borg Warner
18. **Bank Street Speller**
    Broderbund
19. **Karateka**
    Broderbund

Continued from previous page

```
    and to 1 track past the last one
    to be copied (tracks 0-34 here).
1020 T1 = TK : GOSUB 490 : REM "start
    of reading cycle...toggle
    command to read.
1028 POKE 47405 ,24 : POKE 47406 ,96 :
    POKE 47497 ,24 : POKE 47498 ,96 :
    REM "to ignore checksums and end
    marks
1030 GOSUB 430 : GOSUB 100 :ST = ST + 1
    : IF ST < DOS THEN 1030 : REM
    "print T/S on screen, execute
    read, cycle next sector
1040 IF BF THEN 1060 : REM "if buffer
    full then goto write routine
1050 ST = 0 :TK = TK + 1 : IF TK < LT
    THEN 1030 : REM "read the next
    track into the buffer
1060 GOSUB 490 :TK = T1 :ST = 0 : REM
    "toggle command to write and
    return to first  T/S to write
1067 RESTORE : GOSUB 310 : REM "to do
    some sector edits.
1069 POKE 47405 ,208 : POKE 47406 ,19
    : POKE 47497 ,208 : POKE 47498
    ,183 : REM "to normalize after
    using line 1028 to ignore
    checksums and end marks.
1070 GOSUB 430 : GOSUB 100 :ST = ST + 1
    : IF ST < DOS THEN 1070 : REM
    "print T/S being written,
    execute write, do next sector.
1080 ST = 0 :TK = TK + 1 : IF BF = 0 AND
    TK < LT THEN 1070 : REM "write
    next track to disk.
1090 IF TK < LT THEN 1020 : REM "if not
    done, do next set of read/
    writes.
1100 HOME : PRINT : PRINT "DONE^
    WITH^COPY" : END : REM "DONE WITH
    DISK
5000 DATA 6^CHANGES ,^0 ,7 ,31 ,76 ,^
    0 ,7 ,32 ,192 ,^0 ,7 ,33 ,8 ,^0 ,4
    ,79 ,222 ,^0 ,4 ,80 ,234 ,^0 ,4
    ,81 ,234
5010 DATA 6^CHANGES ,^0 ,7 ,13 ,76 ,^
    0 ,7 ,14 ,212 ,^0 ,7 ,15 ,7 ,^0 ,4
    ,79 ,222 ,^0 ,4 ,80 ,234 ,^0 ,4
    ,81 ,234
5020 DATA 4^CHANGES ,^0 ,7 ,0 ,76 ,^0
    ,7 ,1 ,192 ,^0 ,7 ,2 ,8 ,^0 ,4 ,79
    ,222
```

Zaxxon Controller Checksums

```
1000  - $356B      1069  - $EB22
1010  - $A943      1070  - $FA02
1020  - $FBFC      1080  - $204B
1028  - $9278      1090  - $4F8D
1030  - $BD92      1100  - $AA4D
1040  - $4A68      5000  - $894B
1050  - $477C      5010  - $F270
1060  - $40E9      5020  - $6879
1067  - $96C6
```

Other Versions

If you own one of the other two known versions of Zaxxon, you may have to use line 5010 or perhaps line 5020 for your sector edits. To activate these lines, remove line 5000 or line 5000 and line 5010.

Writing Super IOB Controllers

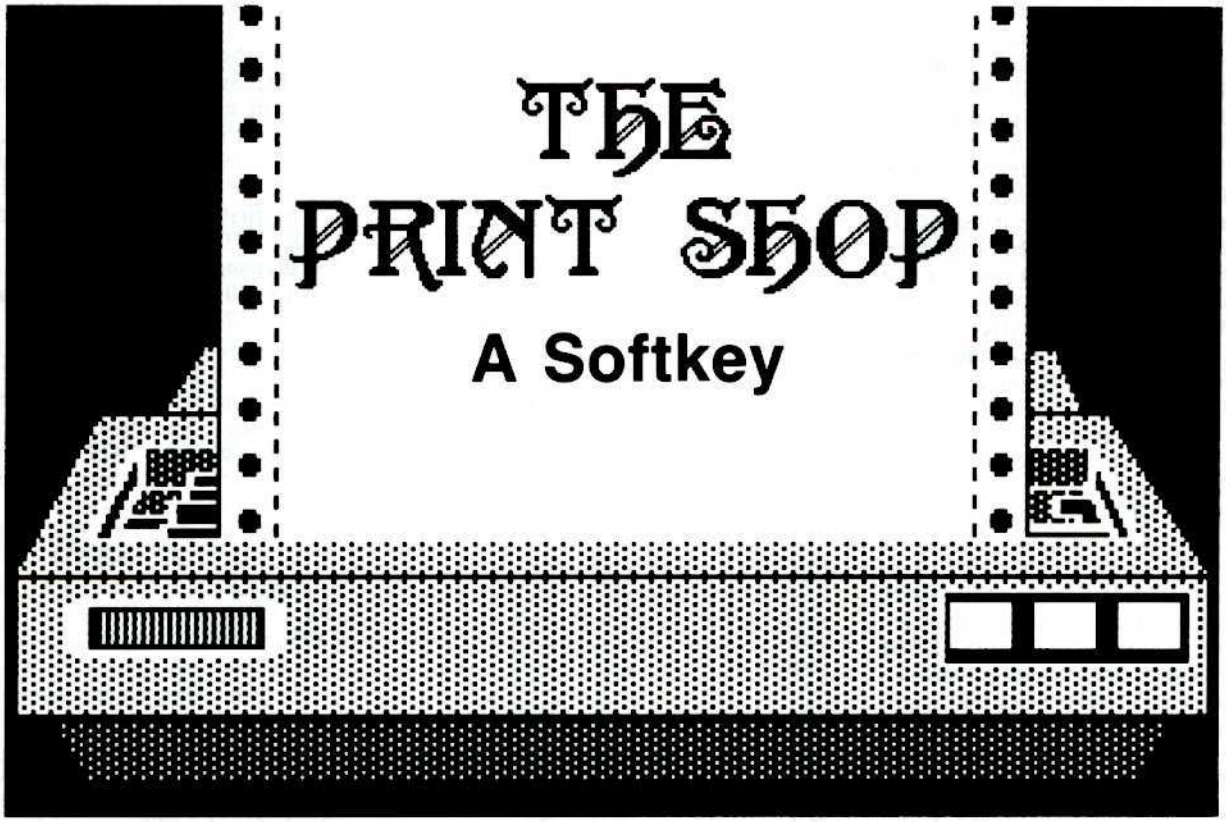
There seem to be many ways to create a Super IOB Controller text file and to integrate it with Super IOB. My favorite method is to write the controller on my word processor, AppleWriter //e, starting each new line with a line number and ending each line with a RETURN. Since my word processor produces standard text files, I have only to "EXEC filename" to load it into memory to find its checksums. To run Super IOB, I just have to "LOAD SUPER IOB", "EXEC filename", and then "RUN" the complete program. Pretty simple.

Super Controller

```
1000 REM SUPER CONTROLLER
1010 TK = 0 :ST = 0 :LT = 35 :CD = WR :
    REM "set to 1st track & sector
    and to 1 track past the last one
    to be copied (tracks 0-34 here).
1020 T1 = TK : GOSUB 490 : REM "start
    of reading cycle...toggle
    command to read.
1021 REM RESTORE :REM "use this
    command ONCE if you use lines
    1022, 1023, 1024, or 1026 below
    (if you use any DATA
    statements).
1022 REM GOSUB190 :REM "to use
    altered address marks...use
    data statement in 62010 and line
    1065 to normalize the writing
    back to disk.
1023 REM GOSUB210 :REM "to use
    altered data marks...use data
    statement in 62010 and line 1065
    to normalize the writing back to
    disk.
1024 REM GOSUB170 :REM "to use
    altered address field and data
    field ENDING MARKS. Use only the
    first two bytes of each in data
    statement at line 62010 and
    include all four even if you only
    need to change one. Use line 1065
    to normalize.
1025 REM GOSUB270 :REM "to ignore
    the address checksums.
1026 REM GOSUB290 :REM "to use
    altered data checksums. The
    normal value is 0. Put the new
    value in the next data statement
    (line 62010) and use line 1065
    below to normalize the writing
    to the disk.
1027 REM GOSUB360 :REM "for an RWTS
    swap routine. RWTS must be
    BLOADed by a line 10010 below.
1028 REM POKE47405 ,24 :POKE47406
    ,96 :POKE47497 ,24 :POKE47498
    ,96 :REM "to ignore checksums
    and end marks...Use line 1069 to
    normalize.
1030 GOSUB 430 : GOSUB 100 :ST = ST + 1
    : IF ST < DOS THEN 1030 : REM
    "print T/S on screen, execute
    read, cycle next sector
1040 IF BF THEN 1060 : REM "if buffer
    full then goto write routine
1050 ST = 0 :TK = TK + 1 : IF TK < LT
    THEN 1030 : REM "read the next
    track into the buffer
1060 GOSUB 490 :TK = T1 :ST = 0 : REM
    "toggle command to write and
    return to first  T/S to write
1063 REM GOSUB360 :REM "for an RWTS
    swap routine.
1065 REM GOSUB230 :REM "to normalize
    writing to the disk if you used
    altered address or data marks or
    i nored checksums or end marks
1067 REM RESTORE :GOSUB310 :REM "to
    do any sector edits. Use a DATA
    statement at line 62010 or
    after. This data statement must
    come AFTER all other DATA
    statements
1069 REM POKE47405 ,208 :POKE47406
    ,19 :POKE47497 ,208 :POKE47498
    ,183 :REM "to normalize after
    using line 1028 to ignore
    checksums and end marks.
1070 GOSUB 430 : GOSUB 100 :ST = ST + 1
    : IF ST < DOS THEN 1070 : REM
    "print T/S being written,
    execute write, do next sector.
1080 ST = 0 :TK = TK + 1 : IF BF = 0 AND
    TK < LT THEN 1070 : REM "write
    next track to disk.
1090 IF TK < LT THEN 1020 : REM "if not
    done, do next set of
    read/writes.
1100 HOME : PRINT : PRINT "DONE^
    WITH^COPY" : END : REM
    DONEWITHDISK
5000 REM DATA "for altered marks.
5010 REM DATA "for sector edits
10010 REM PRINTCHR$(4) "BLOAD
    RWTS,A$1900" :REM "for an RWTS
    swap routine.
```

Super Controller Checksums

```
1000  - $356B      1050  - $7599
1010  - $EDE1      1060  - $A1D9
1020  - $FB5E      1063  - $9017
1021  - $BEF5      1065  - $D665
1022  - $9CF5      1067  - $54DA
1023  - $895D      1069  - $1F21
1024  - $D9CA      1070  - $D8E9
1025  - $97F6      1080  - $8F69
1026  - $26FF      1090  - $6D66
1027  - $869A      1100  - $4D71
1028  - $6A7D      5000  - $6005
1030  - $AF67      5010  - $C44D
1040  - $B889      10010 - $DE89
```



IT WORKS?

By William Hinger &
Albert ~~Stockton~~ Stocker

Additional contributing authors: Brian Chinn, Wes Felty, Clay Ruth & M.M. McFadden

The Print Shop
Broderbund Software
17 Paul Drive
San Rafael, CA 94903
$50.00

Requirements:
Apple][series computer, 48K RAM minimum
Super IOB v1.2
A blank disk
RESET capability optional

After buying The Print Shop about a month ago on the advice of a friend, I became immediately intrigued by the possibilities of the program. Even though The Print Shop will make one backup copy of itself, due to the amount of disk access the program does, my first priority was the deprotection of this disk.

Format of the Disk

When I examined the disk with my sector editor (Tricky Dick), I was pleasantly surprised to find that most of it (tracks $0 through $21) were encoded under completely normal DOS 3.3. Only track $22 was written in a nonstandard format. I assumed that this track was utilized for a nibble count.

After quite a bit of disk snooping I found that The Print Shop disk contains a basically standard DOS 3.3 fastloader which uses normal address (D5 AA 96) and data headers (DE AA) on tracks $0 thru $21 (Hex). The DOS, however, is not in its normal location. If you take a tour through the disk with a sector editor you will find DOS on track $0 sectors $0 to $3 and track $F sector $0 thru track $10 sector $3. You can move this code back to the normal DOS location if you desire but there is really no need; the computer does not care where the operating system is located. The Print Shop DOS uses a bit of self-modifying code and has a few changes to such things as the reset vector and VTOC (more on this later), but generally these have little, if any, effect on the operation of the system. That is until we consider track $22 (Hex).

Track $22 has been written in a very strange format. On a new Print Shop disk there are three sets of bytes inserted into track $22 (I can't really call them headers), along with a lot of garbage with no apparent organization. The first of this sequence of bytes, A5 DF D4, shows that this disk has not been copied. When the built-in copy program is run, these bytes are overwritten and no more copies will be allowed. The second set of bytes on track $22 are F5 AA 00. These bytes identify the disk as a master or copied disk. The third set, D5 DE D4, is used by the nibble count routine.

Finding the Nibble Count

Once I had discovered that track $22 was being used for the nibble count, all that I had to do was to track down the code that was performing the nibble count and somehow bypass it. This required quite a few hours of staring at my computer's monitor.

If you boot up with DOS 3.3 and then try to catalog The Print Shop disk, all you'll get is a message from the authors of the program. This is because DOS 3.3 expects to find a VTOC (Volume Table of Contents) on track $11, sector $0. Bytes $1 and $2 (the first byte is $0) of a VTOC point to the first directory sector. On The Print Shop disk this points to track $11, sector $1, which contains the message from the authors interspersed with a bunch of control characters that cause the screen to scroll up. However, The Print Shop DOS has been modified so that its real VTOC is on track $11, sector $2. So, to catalog the disk from DOS 3.3 you have to do a POKE 45069,2 (B00D:02 from the monitor). This will allow you to view the real directory which has 34 files in it, one of which is a binary file called HELLO. I found that I could BLOAD this file from normal DOS and by examining the contents of AA60.AA61 and AA72.AA73, I discovered that it is loaded at $800 and is $7FC bytes long. For obvious reasons, I assumed that HELLO would be the boot-up file.

Continuing on with my investigation of The Print Shop, I found that if it were copied with a copy program that ignored the errors on track $22, the copy would proceed through the majority of the boot before

hanging up when it accessed the nibble count track. After the disk hung, I could then drop into the monitor with an "old monitor" RESET. Since the HELLO file was loaded at $800, that seemed a good place to start tracking down the nibble count routine.

Let's take a look at this code. At $800 we find a JMP $834. If you look closely, starting at $834, you will see a number of JSR $CD2 instructions. I found that this routine is responsible for BLOADing in several other files from the disk (SYSLIB, TPAGE, MENULIB, CONFIG, MAINPICS, PRCOMS and MENUS1). Looking a little further you'll see some suspicious JSRs. One goes to $4000, but a look at the code there does not turn up any disk access code routines, so it is probably not what we're looking for. A subroutine at $7016 is called from $87A and $8CC. This address bears a little more checking, so let's follow it through.

A listing of $7016 shows a JMP $77F9. That's pretty clear-cut, so let's move on. Listing the code at $77F9 shows a JSR $77B6 and another JSR $8DD7, but an examination of the routine at $77B6 reveals that it just sets up an Input/Output Block (IOB) at $77E4 and calls the RWTS through the vector at $3D9. As track $22 has no sectors, this is not the code we are looking for. Listing the code at $8DD7 shows something we've not seen yet, an indirect JMP. It's located at $8DF0 and does a JMP to the address contained at $8DF3. Typing 8DF3.8DF4 results in 8DF3- E0 BC. The JMP actually goes to address $BCE0.

Let's start by listing the code there. Aha! This is much more interesting! This code first positions the read head to track $22 and then JMPs to $B619. When we look at the code after $B619 we find a lot to digest, but due to all the references to the disk controller ($C08C, $C089, etc.) it becomes evident that we have found the routine that verifies the nibble count.

This code first fills page $BB00 with $FFs. It then turns the drive on and searches for a sequence of four bytes ($D4, $D5, $DE, $D4), reads two more bytes (encoded in a 4 + 4 format) which it combines in the same manner that DOS uses to read track and sector information from the address marks, and finds two trailer bytes ($F5, $AA). The rest of the routine checks the number of bytes in page $BB00 which are minus (or greater than $7F). If this number is less than $A0, the check will fail and will recycle. Otherwise, it turns the drive off and returns to the caller. Some of these routines associated with the nibble count are listed below:

```
8DD7-   EE F5 8D    INC   $8DF5
8DDA-   AD F5 8D    LDA   $8DF5
8DDD-   29 03       AND   #$03
8DDF-   D0 F4       BNE   $8DD5
8DE1-   A9 01       LDA   #$01
8DE3-   8D EA B7    STA   $B7EA
8DE6-   8D F8 B7    STA   $B7F8
8DE9-   A9 AB       LDA   #$AB
8DEB-   49 17       EOR   #$17
8DED-   8D F4 8D    STA   $8DF4
8DF0-   6C F3 8D    JMP   ($8DF3)
.       .  .  .     .     .
.       .  .  .     .     .
.       .  .  .     .     .
BCE0-   A9 00       LDA   #$00
BCE2-   8D F4 B7    STA   $B7F4
BCE5-   A9 22       LDA   #$22
BCE7-   8D EC B7    STA   $B7EC
BCEA-   20 FC BC    JSR   $BCFC
BCED-   4C 19 B6    JMP   $B619
.       .  .  .     .     .
.       .  .  .     .     .
.       .  .  .     .     .
B619-   A0 00       LDY   #$00
B61B-   A9 FF       LDA   #$FF
B61D-   99 00 BB    STA   $BB00,Y
B620-   C8          INY
B621-   D0 FA       BNE   $B61D
B623-   AE F8 05    LDX   $05F8
B626-   BD 89 C0    LDA   $C089,X
B629-   BD 8C C0    LDA   $C08C,X
B62C-   10 FB       BPL   $B629
B62E-   C9 D4       CMP   #$D4
B630-   D0 F1       BNE   $B623
B632-   20 E5 B6    JSR   $B6E5
B635-   C9 D5       CMP   #$D5
B637-   D0 F5       BNE   $B62E
B639-   20 E5 B6    JSR   $B6E5
B63C-   C9 DE       CMP   #$DE
B63E-   D0 F5       BNE   $B635
B640-   20 E5 B6    JSR   $B6E5
B643-   C9 D4       CMP   #$D4
B645-   D0 F5       BNE   $B63C
B647-   EA          NOP
B648-   BD 8C C0    LDA   $C08C,X
B64B-   10 FB       BPL   $B648
B64D-   2A          ROL
B64E-   85 26       STA   $26
B650-   BD 8C C0    LDA   $C08C,X
B653-   10 FB       BPL   $B650
B655-   25 26       AND   $26
B657-   A8          TAY
B658-   4C B5 B6    JMP   $B6B5
```

Defeating the Nibble Count

Now that the nibble count routine has been found *all* that has to be done is to figure out how to disable it. Since the nibble count routine is called via a JSR $8DD7, it seems reasonable to just NOP the call to it that sits at $7806. Looking for this code with a disk search utility turned up only one occurence of it (track $02, sector $0F on my copy).

To test whether The Print Shop could be deprotected by merely NOPing this one piece of code, I first made a copy with the Locksmith 5.0 Fast Copy utility (any copy program that will ignore the errors on track $22 will work). I then used a sector editor to change the JSR $8DD7 to 3 NOP instructions. After I booted up the copy and tested it a bit, I was quite pleased that it seemed to work perfectly.

Since I had come up with a softkey for The Print Shop, I told a friend who also had the package what I had done to deprotect it. He tried the procedure on his disk, but could not find the JSR $8DD7 instruction on the disk where I told him it should have been. Instead, he found that this instruction was contained in the file called MENULIB, which he could BLOAD from DOS 3.3. He just NOPed the bytes starting at $7806 and resaved the file (BSAVE MENULIB, A$6000,L$21F8). This method has worked on his and on several other copies of The Print Shop that I have seen.

Modifying the VTOC

Since the nibble count of track $22 has been bypassed, this track can be freed so that it can be used for extra storage. It is also possible to make The Print Shop catalogable from Normal DOS 3.3. Some changes to the VTOC are needed to do these things.

The first change must be made to the "real" VTOC on track $11, sector $02. Byte $30 is changed from $1B to $21, so that the RWTS will think that the last track written to it was $21 and it will now look at the next track in ascending order to see if it is free. Track $22 is freed in the VTOC's bit map by changing bytes $C0 and $C1 from 00 00 to FF FF. To make The Print Shop catalogable from DOS 3.3, the dummy VTOC on track $11, sector 0, must also be altered. The easiest way to do this is to overwrite this dummy VTOC with an image of the real VTOC (track $11, sector $2) and change byte $FF of it to a 01. This byte is not used by normal DOS, but the Print Shop DOS uses this byte to find the correct VTOC. If it is not a 01, the system will crash.

The Print Shop copy program must also be removed. As it is now, the system will hang if someone presses the ESC key during the boot because the system has to verify the data on track $22 before making a copy. The check for the ESC key is done from the HELLO program at $83D and also by some code that I found on track $0, sector $5. Here is a listing of the code from the HELLO file:

```
0834-   A2 1A       LDX   #$1A
0836-   A0 0B       LDY   #$0B
0838-   20 D2 0C    JSR   $0CD2
083B-   D0 F7       BNE   $0834
083D-   AD 00 C0    LDA   $C000
0840-   C9 9B       CMP   #$9B
0842-   D0 06       BNE   $084A
0844-   8D 10 C0    STA   $C010
0847-   4C 00 10    JMP   $1000
```

This check from the HELLO file can be removed by changing the CMP #$9B to LDA #$01 and resaving the file. Because of the differences in sector allocation on different copies of The Print Shop, it is best to make this change manually. The other check can be removed by changing bytes $39 and $3A on track $0, sector $5 from C9 9B to A9 01 with a sector editor.

To come up with a softkey procedure that would work on any original copy of The Print Shop, I created a controller for Super IOB that will copy tracks $0-$21, make the changes to the VTOC to allow cataloging from normal DOS and also free up track $22 for storage. The changes to remove the call to the nibble count routine and the built-in copy program are then performed manually.

Making the Copy

1) Boot up Super IOB v1.2 and install The Print Shop Controller that is listed at the end of this article.

2) Run Super IOB and follow the prompts to copy The Print Shop to an initialized disk.
3) Insert the Super IOB copy of The Print Shop into your drive and remove the call to the nibble count routine by typing

```
BLOAD MENULIB
CALL -151
7806:EA EA EA
BSAVE MENULIB,A$6000,L$21F8
```

4) Remove the access to the built-in copy program by typing

```
BLOAD HELLO
840:A9 01
BSAVE HELLO,A$800,L$7FC
```

Your deprotected copy of The Print Shop should now have about 22 free sectors available for your own use. — IT WORKS —

Print Shop Controller

```
1000 REM  PRINT SHOP CONTROLLER
1010 TK = 0 : ST = 0 : LT = 34 : CD = WR
1020 T1 = TK : GOSUB 490
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 RESTORE : GOSUB 310 : GOSUB 490
     : TK = T1 : ST = 0
1065 IF TK = 14 THEN A$ =
     "5700<5900.59FFM^N^57FF:01" :
     GOSUB 1210
1070 GOSUB 430 : GOSUB 100 : ST = ST + 1
     : IF ST < DOS THEN 1070
1080 ST = 0 : TK = TK + 1 : IF BF = 0 AND
     TK < LT THEN 1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "DONE^WITH^COPY" :
     END
1200 REM DO MONITOR COMMAND A$
1210 A$ = A$ + "^N^D9C6G" : FOR A = 1
     TO LEN (A$) : POKE 511 + A , ASC (
     MID$ (A$ , A , 1 ) ) + 128
1220 NEXT : POKE 72 , 0 : CALL - 144 :
     RETURN
5000 DATA 5^CHANGES
5010 DATA 0 , 5 , 57 , 169 , 0 , 5 , 58 , 1
5020 DATA 17 , 2 , 48 , 33 , ^17 , 2 , 192
     , 255
5030 DATA 17 , 2 , 193 , 255
```

The Print Shop Controller Checksums

```
1000  - $356B      1090  - $4B7C
1010  - $6344      1100  - $5B29
1020  - $C418      1200  - $2D80
1030  - $D219      1210  - $1475
1040  - $D240      1220  - $66B6
1050  - $8A43      5000  - $B2D8
1060  - $9C31      5010  - $3B04
1065  - $7E1E      5020  - $4415
1070  - $761B      5030  - $E562
1080  - $B240
```

ADVENTURE TIPS ADVENTURE TIPS

Colossal Caves
Adventure International

You need keys to unlock the grate. Look in the well house.
Birds are natural predators of snakes. Drop it.
Can't cross the fissure? You need something from the debris room.
Grease up that iron door with oil from the east pit. It should open easily, now.

* Ultima III
Origin Systems

To find the town of Dawn (and get exotic weapons) you must begin at Lord British' castle. Go West-8, South-35 and wait for the moon cycle to reach (0)(0). Then type, E.

**Contributed by Robert Ellerby.*

Time Zone
Sierra On-Line

Can't satisfy Her Highness, Cleopatra? Check the perfume counter in a European department store in 1700 A.D.
You can't use, "Open sesame" to get into the pyramid. Climb the back side, instead.
You'll find great wealth hidden in the pyramid. Don't leave a single stone unturned.
Get the shovel from the rock garden. You'll find a valuable item in the rice garden if you do some digging.
Deal carefully with the peasant. He sells wares only for the right price.

Zork I
Infocom, Inc.

Looking for anything special? You'll find something in the tree tops.
Only a delicate touch will open the egg without ruining it. Let the master of burglary do it for you.
Don't bother with the door to the house. Be creative.
Every self respecting house has a dungeon under it, doesn't it? You'll find the entrance hidden in the living room.
The trap door will keep you locked in the dungeon until you explore it and find another way out, so don't waste your time trying to get it open.

Ulysses & The Golden Fleece
Sierra On-Line

The castle guard is a pretty nasty guy. If he asks you a question, it's best to answer truthfully.
Before going out to sea, you must get the map from the dock guard. Without it, the hurricane will sink you for sure.
To outwit your fine feathered friends, "go island" to find all your stolen possessions.
The empty bottle should help quench your thirst at the island spring.

Transylvania
Penguin Software

You might need some of the items in the shack at some time later.
Look for a hidden room at the log cabin. The deer antlers hold the clue.
Don't be queasy about opening the coffin.
What's a hut without "three blind mice"? You'll find them in the wagon.
You will face the perils of the cemetary much better with a cross of your own. Take it and then move the gravestone.

Pirate Adventure
Adventure International

Don't be out on the ledge without good reading material.
The torch will light your way but you'll need matches from the apartment. Try the bag.
Brute force won't work to open the chest. Check out the rug in the apartment first, but make sure you bring along a hammer.
Crocodiles eat seafood, don't they? What better way to distract them?

Mission Asteroid
Sierra On-Line

The explosives must be set to blow at a certain time. Try a few to see which one works.
Only a more "in depth" study of the asteroid will reveal the correct place to set off the explosives.
Still looking for the right place to set the explosion? Drop them in the pit.

Adventureland
Adventure International

What's the magic word for using the ax? "Bunyon", of course.
If you've lost the ax, check out the hidden grove.
Need someplace to store all your treasure? Tree climbing is never out of season.
"Scream" loudly to become "unbearable".

* see No. 20, page 15 - updated, improved Version 3.0

CORE

* The Graphic Grabber For The Print Shop

By Ray Darrah

One of the best features of The Print Shop by Broderbund is the ability to create graphics and store them on an unprotected diskette so that they can be used for letterheads, greeting cards, banners, etc. Unfortunately, the graphic editor supplied by Broderbund is not of superior quality. If you are using the keyboard, it is particularly unfriendly. To make a long story short, there are several graphic editors on the market (The Micro Illustrator and The Graphics Magician, for example) which offer more versatility.

Wouldn't it be great if you could design a graphic image using your favorite editor and then use it with The Print Shop?

Or, how about capturing an image from a favorite game and using that as a graphic in The Print Shop?

Dream no longer! On the following pages you'll find The Graphic Grabber, a program designed to perform this very task! The Graphic Grabber can extract an image from any normally saved hi-res screen for use with The Print Shop.

Typing It In

Use the procedure outlined on the page opposite the contents page to type in the BASIC program listing and also the Hexdump at the end of this article. Save the BASIC program with:

SAVE GRAPHIC GRABBER

Save the Hexdump with:

BSAVE OBJ.GRAPHIC GRABBER, A$4000,L$E2

Using The Graphic Grabber

When The Graphic Grabber is RUN, the hi-res screen page 1 is not altered. This is so that you may use what is currently on the screen as a graphic rather than going through the step of saving the hi-res screen and then later loading it again (although you may do it this way if you wish).

There should be a flashing rectangle in the center of the screen and at the bottom of the screen there should be a list of commands. Think of the rectangle as a window. Everything inside this window and pixels behind the edges of the window itself will be saved when you use the "SAVE WINDOW" option. The maximum size graphic that can be transferred to The Print Shop is 88 x 52 pixels.

Moving the Window

Motion is accomplished by using the standard ESCape keys. Pressing the keys "**I,J,K,M**" will move the window up, left, right and down respectively.

Toggle Text

This option is invoked by pressing the **spacebar**. It will either make the command menu visible or invisible. When the command menu is invisible, the bottom 32 lines of the hi-res screen are visible.

Load Screen

If you type "**L**", an entire normally

BSAVEd hi-res picture will be loaded. When you are asked for the filename, a special disk access routine is invoked. You may press ESC to abort the Load Screen option. If you type a null as the filename, then a CATALOG will be displayed. You may specify Slot, Drive and Volume parameters by including them in the filename. For example, if you typed

EXODUS1,D2

as a filename, the file EXODUS1 would be loaded in to the hi-res screen from drive 2. This also works for the directory function. If you want a directory of a different disk, simply make the first character a comma. For example, if you typed

,S6,D1

as a filename, the CATALOG of the disk in slot 6, drive 1 would be displayed.

Save Window

Pressing **RETURN** will invoke the save window routine. This also uses the special disk access routine mentioned above. The only difference is that the graphic inside the window is saved rather than a whole hi-res picture loaded.

Print Mode

The print mode is the manner in which the screen will be printed by The Print Shop. When the words "Print Mode" are inverse, dots on the screen which are lit ("on") be printed as black on your paper. When the words "Print Mode" are normal, then the lit dots on the screen will be white (not printed) on your paper. The print mode is toggled by pressing "**P**".

Leaving The Program

You may leave the program by pressing **ESC**. If you should accidentally escape from The Graphic Grabber, a RUN will get things going again. Nothing except the window's position will be changed when you restart the program this way.

A Bit About Graphics

A graphic created with the graphic editor section of The Print Shop (or the Graphic Grabber) is a normal Binary file with an address of $5800 and a length of $240. However, The Print Shop will recognize any normal Binary file that is four sectors long as a graphic.

The first 572 (or $23C) bytes define the graphic and the rest are ignored. These bytes are arranged as fifty-two rows of eleven bytes. Each bit in the rows corresponds to a dot with the most significant bit of each byte being the leftmost dot. A one bit is a black dot and a zero bit is a white dot. Row two sequentially follows row one. This

Continued on page 19

The Graphic Grabber Source Code

```
00E2- HGR.Y     .EQ $E2
00E0- HGR.X     .EQ $E0
0030- MON.HMASK .EQ $30
00E5- HGR.HORIZ .EQ $E5
00FE- ROWS      .EQ $FE       NUMBER OF ROWS LEFT
00FF- COLUMNS   .EQ $FF       NUMBER OF COLUMNS LEFT
0026- MON.GBASL .EQ $26
00FD- BITS      .EQ $FD       NUMBER OF BITS LEFT
00FC- XSAVE     .EQ $FC
00E7- HGR.SCALE .EQ $E7
00FB- STATUS.SQ .EQ $FB

F417- HPOSN     .EQ $F417
F48A- MOVE.RIGHT .EQ $F48A
F65D- XDRAW     .EQ $F65D

                .OR $4000     HGR2 IS UNUSED MOSTLY
                .TF OBJ.GRAPHIC GRABBER

4000: A9 08             LDA #8        8 BITS IN A BYTE
4002: 85 FD             STA BITS
4004: A9 58             LDA #$58      MSB=$58
4006: 8D 37 40          STA ROLLER+2
4009: A2 00             LDX #0        INITIALIZE X
400B: A9 34             LDA #52       52 ROWS
400D: 85 FE             STA ROWS
400F: A9 58     DO.ROW  LDA #88       88 COLUMNS
4011: 85 FF             STA COLUMNS
4013: 86 FC             STX XSAVE
4015: 20 CD 40          JSR HPOSN1
4018: A6 FC             LDX XSAVE
401A: A5 30     DO.NEXT LDA MON.HMASK
401C: 29 7F             AND #$7F      IGNORE MSB
401E: 31 26             AND (MON.GBASL),Y
4020: 20 31 40          JSR SAVE.BIT
4023: 20 8A F4          JSR MOVE.RIGHT GET NEXT DOT
4026: C6 FF             DEC COLUMNS   DONE?
4028: D0 F0             BNE DO.NEXT   NOPE!
402A: E6 E2             INC HGR.Y     NEXT Y POSITION
402C: C6 FE             DEC ROWS      DONE?
402E: D0 DF             BNE DO.ROW    NOPE!
4030: 60                RTS           YUP!

4031: 18        SAVE.BIT CLC          NO DOT?
4032: F0 01             BEQ ROLLER    YUP!
4034: 38                SEC
4035: 3E 00 58  ROLLER  ROL $5800,X   PUT BIT INTO PICTURE
4038: C6 FD             DEC BITS      BYTE DONE?
403A: D0 0A             BNE RTS.1     NOPE!
403C: A9 08             LDA #8        FIX BITS
403E: 85 FD             STA BITS
4040: E8                INX           NEXT POS
4041: D0 03             BNE RTS.1     IF NO PAGE CROSSING
4043: EE 37 40          INC ROLLER+2 NEXT PAGE
4046: 60        RTS.1   RTS

*-------------------------------*
*       MOVE WINDOW             *
*-------------------------------*

4047: 8D 10 C0  DONE.MOV STA $C010    CLEAR KEY
404A: 20 9E 40  MOVE.W   JSR DO.SQUARE
404D: A0 50              LDY #80
404F: A2 50     TEST.X   LDX #80
4051: 2C 00 C0  TEST.Y   BIT $C000    KEYPRESS?
4054: 30 08              BMI EVALUATE YUP
4056: CA                 DEX          WAIT A WHILE
4057: D0 F8              BNE TEST.Y
4059: 88                 DEY
405A: D0 F3              BNE TEST.X
405C: F0 EC              BEQ MOVE.W

405E: A5 FB     EVALUATE LDA STATUS.SQ SQUARE ON?
4060: F0 03              BEQ SKIP.ER  NOPE
4062: 20 9E 40           JSR DO.SQUARE
```

Continued on next page

Hardcore COMPUTIST welcomes articles of interest to users of the Apple][(or compatible) computers and would like to publish well-written material including:

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

Send your submission on a DOS 3.3 disk using an Apple (or compatible) editing program. Enclose a double-spaced hardcopy manuscript (typewritten or computer printed). Submissions will be returned only if adequate packaging is enclosed.

Have you written an ARTICLE or PROGRAM you'd like to see published in Hardcore COMPUTIST?

We would like to hear from you!

Hardcore COMPUTIST pays on publication. Rate of payment depends on the amount of editing necessary and the length of the article- generally between $10 for a short softkey, and $50 per typeset page for a full-length article. For a higher pay rate, enclose the original commercial disk for verification of softkeys. 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. However, alternate arrangements may be made with individual authors, depending on the merit of the contribution.

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
PO Box 110846-K
Tacoma, WA 98411

Continued from previous page

```
4065: AD 00 C0  SKIP.ER  LDA $C000     GET VALUE
4068: C9 C9              CMP #$C9      MOVE.UP?
406A: D0 09              BNE TST.J
406C: A5 E2              LDA HGR.Y
406E: F0 D7              BEQ DONE.MOV
4070: C6 E2              DEC HGR.Y
4072: 4C 47 40           JMP DONE.MOV
4075: C9 CA      TST.J   CMP #$CA      MOVE LEFT?
4077: D0 09              BNE TST.K
4079: A5 E0              LDA HGR.X
407B: F0 CA              BEQ DONE.MOV
407D: C6 E0              DEC HGR.X
407F: 4C 47 40           JMP DONE.MOV
4082: C9 CB      TST.K   CMP #$CB      MOVE RIGHT?
4084: D0 0A              BNE TST.M
4086: A5 E0              LDA HGR.X
4088: C9 C0              CMP #$C0
408A: B0 BB              BCS DONE.MOV
408C: E6 E0              INC HGR.X
408E: D0 B7              BNE DONE.MOV
4090: C9 CD      TST.M   CMP #$CD      MOVE DOWN?
4092: D0 2F              BNE OTHER.KEY
4094: A5 E2              LDA HGR.Y
4096: C9 8C              CMP #140
4098: F0 AD              BEQ DONE.MOV
409A: E6 E2              INC HGR.Y
409C: D0 A9              BNE DONE.MOV

      DO.SQUARE
409E: 20 CD 40           JSR HPOSN1    GET FIRST POS
40A1: A2 57              LDX #87       SCALE=87
40A3: A9 00              LDA #0        ROT=0
40A5: 20 C4 40           JSR XDRAW1    DRAW IT
40A8: A2 33              LDX #51       SCALE=51
40AA: A9 10              LDA #16       ROT=16
40AC: 20 C4 40           JSR XDRAW1    DRAW IT
40AF: A2 57              LDX #87       SCALE=87
40B1: A9 20              LDA #32       ROT=32
40B3: 20 C4 40           JSR XDRAW1    DRAW IT
40B6: A2 33              LDX #51       SCALE=51
40B8: A9 30              LDA #48       ROT=48
40BA: 20 C4 40           JSR XDRAW1    DRAW IT
40BD: A5 FB              LDA STATUS.SQ
40BF: 49 01              EOR #$01
40C1: 85 FB              STA STATUS.SQ
40C3: 60        OTHER.KEY RTS

40C4: 86 E7      XDRAW1  STX HGR.SCALE
40C6: A2 E0              LDX #SHAPE
40C8: A0 40              LDY /SHAPE    MSB
40CA: 4C 5D F6           JMP XDRAW

40CD: A5 E2      HPOSN1  LDA HGR.Y
40CF: A6 E0              LDX HGR.X
40D1: A4 E1              LDY HGR.X+1
40D3: 4C 17 F4           JMP HPOSN     CALCULATE STARTING ADDR

40D6: 68         FIX.ERR PLA           POP STACK
40D7: A8                 TAY
40D8: 68                 PLA
40D9: A6 DF              LDX $DF       FIX POINTER
40DB: 9A                 TXS
40DC: 48                 PHA
40DD: 98                 TYA
40DE: 48                 PHA
40DF: 60                 RTS

40E0: 05 00      SHAPE   .HS 0500
```

Continued from page 17

makes the maximum size of a graphic 88 pixels horizontally by 52 pixels vertically for a total of 4576 pixels.

The Graphic Grabber BASIC Program

```
10 REM &&&&&&&&&&&&&&&&&&&&&&&&&&
20 REM &                        &
30 REM &    GRAPHIC GRABBER     &
40 REM &          FOR           &
50 REM &     THE PRINT SHOP     &
60 REM &          BY            &
70 REM &      RAY DARRAH        &
80 REM &                        &
90 REM &&&&&&&&&&&&&&&&&&&&&&&&&&
100 REM
110 REM INITIALIZE VARIABLES
120 IF PEEK (16384 ) + PEEK (16385 ) <
    > 177 THEN PRINT CHR$ (4 )
    "BLOAD^OBJ.GRAPHIC^
    GRABBER,A$4000"
130 HIMEM: 8191 : POKE 251 ,0 :W = 1 :
    POKE 230 ,32
140 K$ = "LP^" + CHR$ (13 ) + CHR$ (27
    ) :X = 96 :Y = 70
150 TEXT : HOME : POKE - 16301 ,0 :
    GOSUB 460 : ONERR GOTO 420
160 REM GET A KEY
170 POKE 224 ,X - INT (X / 256 ) * 256 :
    POKE 225 ,X / 256 : POKE 226 ,Y
180 CALL 16455 :X = PEEK (224 ) + PEEK
    (225 ) * 256 :Y = PEEK (226 ) : GET
    A$
190 FOR A = W TO LEN (K$ ) : IF A$ <>
    MID$ (K$ ,A ,W ) THEN NEXT : GOTO
    170
200 ON A GOSUB 310 ,240 ,220 ,270 ,510
    : GOTO 170
210 REM TOGGLE TEXT
220 POKE - 16302 + WI ,W :WI = W - WI :
    RETURN
230 REM CHANGE PRINT MODE
240 IF PEEK (16434 ) = 208 THEN POKE
    16434 ,240 : GOTO 460
250 POKE 16434 ,208 : GOTO 460
260 REM SAVE WINDOW
270 A$ = "SAVE^WINDOW" : GOSUB 340 :
    IF F$ = CHR$ (27 ) THEN 460
280 POKE 224 ,X - INT (X / 256 ) * 256 :
    POKE 225 ,X / 256 : POKE 226 ,Y :
    CALL 16384
290 PRINT CHR$ (4 ) "BSAVE" F$
    ",A$5800,L$240" : GOTO 460
300 REM LOAD A SCREEN
310 A$ = "LOAD^SCREEN" : GOSUB 340 :
    IF F$ = CHR$ (27 ) THEN 460
320 PRINT CHR$ (4 ) "BLOAD" F$
    ",A$2000" : GOTO 460
330 REM GET A FILENAME
340 TEXT : HOME : HTAB 20 - LEN (A$ ) /
    2 : PRINT A$ : VTAB 5 : PRINT TAB(
    8 ) "RETURN^DOES^A^DIRECTORY"
350 VTAB 12 : PRINT "FILENAME=>" ; :
    FLASH : PRINT "^" ; : NORMAL :
    PRINT CHR$ (8 );
360 WAIT - 16384 ,128 : IF PEEK ( -
    16384 ) = 155 THEN GET F$ : RETURN
370 INPUT "" ;F$ : IF PEEK (512 + LEN
    (F$ ) ) THEN VTAB PEEK (37 ) : CALL
    64578
380 FOR A = 512 + LEN (F$ ) TO 768 : IF
    PEEK (A ) THEN F$ = F$ + CHR$ (
    PEEK (A ) ) : NEXT
390 IF F$ = "" OR LEFT$ (F$ ,W ) = "," "
    THEN HOME : PRINT CHR$ (4 )
    "CATALOG" F$ : PRINT : PRINT
    "PRESS^A^KEY^" ; : GET F$ : GOTO
    340
400 RETURN
410 REM ONERR GOTO
420 CALL 16598 :ERR = PEEK (222 ) : IF
    ERR = 255 THEN RESUME
430 TEXT : HOME : VTAB 12 : PRINT
    "ERROR^#" ERR CHR$ (7 )
440 F$ = CHR$ (27 ) : FOR B = 1 TO 1500 :
    NEXT
450 REM PRINT COMMANDS
460 HTAB W : VTAB 22 : PRINT
    "L=>LOAD^SCREEN" TAB( 21 ) "I"
    TAB( 25 ) "SPC=>TOGGLE^TEXT" ;
470 HTAB 20 : PRINT "J^K" : PRINT
    "RET=>SAVE^WINDOW" TAB( 21 )
    "M" TAB( 25 ); "P=>" ;
480 INVERSE : IF PEEK (16434 ) = 208
    THEN NORMAL
490 PRINT "PRINT^MODE" ; : NORMAL :
    POKE - 16304 ,0 : POKE - 16297 ,0 :
    RETURN
500 REM LEAVE PROGRAM
510 TEXT : HOME : END
```

The Graphic Grabber Hexdump

```
4000: A9 08 85 FD A9 58 8D 37    $3A45
4008: 40 A2 00 A9 34 85 FE A9    $EA2A
4010: 58 85 FF 86 FC 20 CD 40    $A6F3
4018: A6 FC A5 30 29 7F 31 26    $28D2
4020: 20 31 40 20 8A F4 C6 FF    $28DB
4028: D0 F0 E6 E2 C6 FE D0 DF    $B154
4030: 60 18 F0 01 38 3E 00 58    $300F
4038: C6 FD D0 0A A9 08 85 FD    $E958
4040: E8 D0 03 EE 37 40 60 8D    $C207
4048: 10 C0 20 9E 40 A0 50 A2    $0E65

4050: 50 2C 00 C0 30 08 CA D0    $7F15
4058: F8 88 D0 F3 F0 EC A5 FB    $17E2
4060: F0 03 20 9E 40 AD 00 C0    $357E
4068: C9 C9 D0 09 A5 E2 F0 D7    $50FE
4070: C6 E2 4C 47 40 C9 CA D0    $1270
4078: 09 A5 E0 F0 CA C6 E0 4C    $E208
4080: 47 40 C9 CB D0 0A A5 E0    $F009
4088: C9 C0 B0 BB E6 E0 D0 B7    $730A
4090: C9 CD D0 2F A5 E2 C9 8C    $8513
4098: F0 AD E6 E2 D0 A9 20 CD    $8DD4

40A0: 40 A2 57 A9 00 20 C4 40    $D3CE
40A8: A2 33 A9 10 20 C4 40 A2    $97AB
40B0: 57 A9 20 20 C4 40 A2 33    $F1C7
40B8: A9 30 20 C4 40 A5 FB 49    $D613
40C0: 01 85 FB 60 86 E7 A2 E0    $DFBA
40C8: A0 40 4C 5D F6 A5 E2 A6    $51B9
40D0: E0 A4 E1 4C 17 F4 68 A8    $2CC3
40D8: 68 A6 DF 9A 48 98 48 60    $9F36
40E0: 05 00                      $078D
```

The Graphic Grabber BASIC Checksums

```
10  - $BADD     270 - $CAA0
20  - $9B13     280 - $45F1
30  - $4D3B     290 - $A65C
40  - $AD92     300 - $1C10
50  - $C899     310 - $1DBA
60  - $FF65     320 - $8958
70  - $A3BF     330 - $E874
80  - $A900     340 - $14A1
90  - $924D     350 - $D9C6
100 - $CB63     360 - $405E
110 - $BD13     370 - $4C26
120 - $4215     380 - $069F
130 - $0074     390 - $6F60
140 - $CA99     400 - $6B22
150 - $AD80     410 - $2A22
160 - $DAD0     420 - $2D36
170 - $2FB8     430 - $6E89
180 - $5695     440 - $6D39
190 - $7FE3     450 - $0D96
200 - $074B     460 - $D576
210 - $A599     470 - $DCA0
220 - $9D01     480 - $DEA2
230 - $C78D     490 - $6D6B
240 - $4A64     500 - $CF45
250 - $F7DE     510 - $6764
260 - $BE45
```


We've Moved!

Our mailing address remains the same: (PO Box addresses for the various departments can be found on the Contents page), but we have a NEW phone number.

Technical Support

If you have a question about something that has appeared in Hardcore COMPUTIST, our technical staff is available to help you from 1:30-4:30 pm (Pacific Time) on Tuesdays and Thursdays at our new number:

(206) 474-5750

Due to the small size of our staff, technical services can be provided at the above times only.

PART TWO:

The Lone Catalog Arranger v1.0

By Ray Darrah

Requirements:
Apple][Plus with slot Ø RAM card or Apple //e
One disk drive with DOS 3.3
An accurate typing hand

This article describes the final portion of a program called The Lone Catalog Arranger (henceforth referred to as LCA). As you may remember from the article which appeared in Hardcore COMPUTIST No. 16, LCA was designed for the manipulation of DOS 3.3 disk directories. With this program you can:

1) View the free and used space on a disk
2) Undelete, Delete, Lock and Unlock files
3) Insert and see illegal characters in filenames
4) Change the order of the files in the CATALOG
5) Remove deleted files from the directory
6) Create dazzling titles for your disks

The program is comprised of two parts: An Applesoft BASIC program and a Machine Language program. *Due to space limitations, only the BASIC portion of the program was presented in the previous issue. This is the machine language portion that was missing from Hardcore COMPUTIST No. 16.* Please note that this program *does* require the BASIC program in Hardcore COMPUTIST No. 16.

CORE

The Lone Catalog Arranger Source Code

```
03E3- GETIOB    .EQ $03E3     ROUTINE THAT MAKES A,Y POINT TO RWTS
03D9- RWTS      .EQ $03D9     PAGE THREE VECTOR FOR CALLING RWTS
B7F5- RWTS.ERR  .EQ $B7F5     RWTS ERROR CODE IF C=1
D412- BASIC.ERR .EQ $D412     INVOKE A BASIC ERROR
FDF0- COUT1     .EQ $FDF0     ABSOLUTE PR#0 OUTPUT
00FB- YSAVE1    .EQ $FB       TEMPORARY STORAGE OF Y
00FC- COUNTER   .EQ $FC       COUNTER OF NUMBER OF ELEMENTS IN F$S
00FD- YSAVE     .EQ $FD       TEMPORARY STORAGE FOR Y
00FE- TEMP.PTR  .EQ $FE       AND $FF MAKE A TEMPORARY POINTER
FDED- COUT      .EQ $FDED     PRINT A CHARACTER
0083- CUR.VAR   .EQ $83       POINTER TO THE LAST USED VARIABLES VALUE
00FA- ARRY.PTR  .EQ $FA       TEMPORARY STORAGE
006F- STRNG.BOT .EQ $6F       POINTER TO START OF STRING STORAGE
FC24- VTAB      .EQ $FC24     PERFORM THE VTAB FUNCTION
0028- BASL      .EQ $28       BAS ADDRESS FOR SCREEN
00FD- ANSWER    .EQ $FD       ANSWER OF WHETHER A SECTOR IS ALLOCATED OR NOT
00FE- TRACK     .EQ $FE       TRACK FOR ALLOCATION
00FF- SECTOR    .EQ $FF       SECTOR FOR ALLOCATION

                .OR $800      TOO BIG FOR PAGE THREE
                .TF OBJ.LONE ARRANGER

0800: 00 0D 08
0803: 0A 00     LINE      .HS 000D080A00
0805: 8C 32 30
0808: 36 33 3A
080B: AC                  .HS 8C323036333AAC
080C: 00 00 00            .HS 000000

*-------------------------------------------------------------*
*                    COPY ROM INTO LANGUAGE CARD              *
*-------------------------------------------------------------*

080F: AD 81 C0            LDA $C081     ROM READ, RAM WRITE
0812: AD 81 C0            LDA $C081     TWICE
```

Continued on next page

Typing It In

The machine language portion of LCA is very special. It was designed to be combined with the Applesoft portion of LCA in order to form one file that can be RUN. In order to do this, you must follow these steps:

1) Clear the program in memory via DOS

FP

2) Enter the monitor

CALL-151

3) Key in the hexdump at the end of this article and save it

BSAVE OBJ.LONE ARRANGER, A$800,L$1DF

4) Pop back into BASIC

E003G

5) Start the program and change the beginning of program pointers

RUN

6) Load in the BASIC portion of LCA that you saved from Hardcore COMPUTIST No. 16

LOAD BAS.LONE ARRANGER

7) Start up this part of the program

RUN

8) As soon as the title page appears press ESC to leave the program.
9) Save both the machine language and the BASIC portions as one file

SAVE LONE ARRANGER

The One File Concept

As indicated in Step 9, both the machine language program and the BASIC program are saved as one Applesoft file. Here is an explanation of how this is accomplished. If you are a new computist, you may get a little queasy during this explanation, but try to hang in there anyway.

First of all, any BASIC program in memory was cleared with an "FP". This has the effect of restoring the beginning of program pointers ($67 and $68 or 103 and 104) to $801 or 2049. This is the usual starting place of BASIC programs. It just so happens that the machine language part of LCA starts at $800 or 2048, one byte before this address. The first 15 bytes of the program are designed to look like a BASIC program at $801 or 2049. That is why, after keying in the hexdump, a LIST will reveal:

```
10 CALL 2063 : RUN
```

The RUN command in Step 5 executes this artificial program. The CALL 2063 in the program executes a real machine language program starting at $80F or 2063. This program changes the beginning of program pointers to point just beyond the end of the

```
0815: A9 D0             LDA #$D0     START WITH PAGE $D0
0817: 8D 21 08          STA MOVER+2  FIX STORER
081A: 8D 24 08          STA MOVER+5
081D: A0 00             LDY #0       OFFSET ZERO
081F: B9 00 D0  MOVER   LDA $D000,Y  GET A BYTE
0822: 99 00 D0          STA $D000,Y  STORE SAME PLACE
0825: C8                INY
0826: D0 F7             BNE MOVER
0828: EE 21 08          INC MOVER+2  NEXT PAGE
082B: EE 24 08          INC MOVER+5
082E: D0 EF             BNE MOVER    UNTIL ADDRESS FLIPS OVER

*-------------------------------------------------------------*
*                  MOVE POINTER TO BEGINNING OF               *
*                          BASIC PROGRAM                      *
*-------------------------------------------------------------*

0830: A9 DD             LDA #END.OBJ GET END OF SOURCE
0832: 85 67             STA $67      POINTER LSB
0834: A9 09             LDA /END.OBJ
0836: 85 68             STA $68      POINTER MSB
0838: 60                RTS          RETURN FOR RUN

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

0839: 20 E3 03          JSR GETIOB   GET POINTERS
083C: 20 D9 03          JSR RWTS     DO THE FUNCTION SPECIFIED BY BASIC
083F: B0 01             BCS MAKE.ERR IF C=1 THEN AN ERROR HAS OCCURRED
0841: 60                RTS          EVERYTHING O.K. RETURN TO PROGRAM
0842: AD F5 B7  MAKE.ERR LDA RWTS.ERR UH OH, AN ERROR
0845: 4A                LSR          DIVIDE ERROR CODE
0846: 4A                LSR          BY 16
0847: 4A                LSR
0848: 4A                LSR
0849: AA                TAX          PUT INTO X FOR BASIC ROUTINE
084A: 4C 12 D4          JMP BASIC.ERR

*-------------------------------------------------------------*
*                    SHOW CONTROLS AS INVERSE                 *
*-------------------------------------------------------------*

084D: C9 80     COUTR   CMP #$80     IF LESS THAN CTRL.@
084F: 90 06             BCC PRINT.IT THEN O.K.
0851: C9 A0             CMP #$A0     IF GREATER THAN OR = NORMAL SPACE
0853: B0 02             BCS PRINT.IT THEN O.K.
0855: 29 3F             AND #$3F     MAKE INVERSE
0857: 4C F0 FD  PRINT.IT JMP COUT1   SEND IT TO THE SCREEN

*-------------------------------------------------------------*
*                    COLLECT FILES INTO F$(XXX)               *
*-------------------------------------------------------------*

085A: 84 FB             STY YSAVE1   ALWAYS SAVE Y
085C: A4 FD             LDY YSAVE
085E: 91 6F     STORER  STA (STRNG.BOT),Y
0860: C8                INY
0861: 84 FD             STY YSAVE
0863: D0 02             BNE RTS.1
0865: E6 70             INC STRNG.BOT+1
0867: A4 FB     RTS.1   LDY YSAVE1   RESTORE Y
0869: 60        EXIT    RTS
086A: 84 FB     NXT.STNG STY YSAVE1  SAVE Y
086C: E6 FC             INC COUNTER  ONE MORE STRING
086E: A9 00             LDA #0       RESET OFFSET
0870: 85 FD             STA YSAVE
0872: A4 FA             LDY ARRY.PTR GET POINTER
0874: A9 28             LDA #40      LENGTH 40
0876: 91 83             STA (CUR.VAR),Y ASSIGN
0878: C8                INY          FOR ADDR
0879: A5 6F             LDA STRNG.BOT MAKE ROOM FOR FORTY CHARACTERS
087B: 38                SEC
087C: E9 28             SBC #40
087E: 85 6F             STA STRNG.BOT
0880: 91 83             STA (CUR.VAR),Y POINTER
0882: A5 70             LDA STRNG.BOT+1 MSB
0884: E9 00             SBC #0       IF C=0
```

Continued on next page

machine language routines. Therefore, the LOAD command of Step 6 puts the BASIC portion of the program directly above the machine language program. This program is then executed in Step 7. When you press ESC, the real BASIC program puts the pointers to beginning of BASIC back to $801 or 2049. Since the end of program pointers have remained unchanged, memory from $801 all the way to the end of the BASIC portion is saved with the SAVE command of Step 9.

This is illustrated in the following diagram:

```
**********************************
* Addr.  *          Function      *
*        *                        *
* $0800  * Must be a zero so that BASIC *
*  $0800 * will work.              *
*        *                        *
* $0801  * Artificial BASIC line which *
*  $080E * calls $080F and then RUNs. *
*        *                        *
* $080F  * Copies ROM into the 16K card *
*  $082F * so that BASIC can be modified.*
*        *                        *
* $0830  * Sets start of BASIC program *
*  $0838 * pointers to $09DD.      *
*        *                        *
* $0839  * Machine language routines *
*  $09DB * that are used by BASIC LCA. *
*        *                        *
* $09DC  * When start of BASIC program *
*  $09DC * pointer is set to $9DD, this *
*        * byte is just like $0800 (00). *
*        *                        *
* $09DD  * A couple of zeroes that *
*  $09DE * indicate this as the end of *
*        * the BASIC program. These *
*        * zeroes will be changed when a *
*        * program is LOADed.      *
*        *                        *
**********************************
```

A Word About The ML of LCA

To gain a better insight of LCA operation, a discussion of the main machine language subroutines follows. If you get weak in the knees whenever machine language is discussed, you may wish to skip this part.

I put the machine language program at $0800 because the routines quickly exceeded the $D0 length limit imposed upon code placed in page $03 of memory.

Copy ROM Into Language Card

This routine spans from $80F to $82F and is called only once when LCA is first run. The rare occurrence of two consecutive LDA instructions to ensure that the language card is configured so that memory reads return the ROM values and memory writes are stored into the language card RAM. The entire ROM image is then stored into the language card RAM. This is necessary because of several routines in the BASIC portion of LCA that modify BASIC. One example of this is line 840. The modification performed in this line alters BASIC so that it allows normal, inverse and flashing

characters to be printed simultaneously. The "X=" statement of line 1320 configures the language card so that its RAM is both readable and writable. The language card stays this way while LCA is running.

Move Beginning of BASIC Program Pointers

This routine is also called only once when LCA is first run. It spans from $830 to $838 and sets the beginning of program pointer to $9DD. This is where the BASIC portion of LCA begins.

Call RWTS

This routine occupies memory from $839 through $84C and exists only because DOS does not allow an easy method to call the RWTS directly from BASIC. BASIC routines that use this ML routine are "DELETE SUBROUTINE", "EXHUME SUBROUTINE", and "BUILD DIRECTORY". These routines alter the IOB parameter list in DOS (starting at location $B7E8 or 47080) before calling "CALL RWTS". The ML routine calls the RWTS with the 6502 registers set correctly and, if an error occurs, invokes a BASIC error which will then be handled by an ONERR GOTO statement.

Show Controls As Inverse

This routine exists as an interceptor between the program that is printing and the actual COUT routine. It spans from $84D to $859 and is used whenever control characters are to be displayed as inverse characters. An example of this is when the directory is displayed during LCA execution.

Collect Files Into F$(xxx)

Memory addresses from $85A through $891 are occupied by this routine which is used by "NEW CATALOG ROUTINE" to store the directory into the locations in memory where the F$ array is stored.

New Catalog Routine

This routine spans from $892 through $910 and is used to read the directory. By using "COLLECT FILES INTO F$(XXX)", this routine stores the directory into an array so that BASIC can manipulate it. The current variable pointer must have been previously set (the F$(0)="" statement in line 1500) to the array that is to hold the directory. It is very similar to the CATALOG routine in DOS. The strings that are sent to "COLLECT FILES INTO F$(xxx)" are identical to the file entries that a CATALOG command would produce, with two exceptions. First of all, deleted files are not ommitted and are preceded with a hyphen.

Continued from previous page

```
0886: 85 70             STA STRNG.BOT+1 SAVE IT
0888: C8                INY
0889: 91 83             STA (CUR.VAR),Y
088B: C8                INY
088C: 84 FA             STY ARRY.PTR UPDATE POINTER
088E: D0 D7             BNE RTS.1     ..ALWAYS
0890-          .BS 2       BECAUSE OF UPDATE BETWEEN HC16 AND HC17

*-------------------------------------------------------------*
*                      NEW CATALOG ROUTINE                    *
*-------------------------------------------------------------*

0892: A9 00             LDA #0        INITIAL OFFSET
0894: 85 FA             STA ARRY.PTR
0896: 20 DC AB          JSR $ABDC     INITIALIZE FILE MANAGER WORK AREA
0899: 20 F7 AF          JSR $AFF7     GET VTOC
089C: 18                CLC           FOR FIRST DIRECTORY SECTOR
089D: 20 11 B0  READ.NXT JSR $B011    GET NEXT SECTOR
08A0: B0 C7             BCS EXIT      NO MORE SECTORS
08A2: A2 00             LDX #0        START AT ZERO
08A4: 8E 9C B3  DO.FILE  STX $B39C    SAVE IT
08A7: 20 6A 08          JSR NXT.STNG BEFORE EACH ENTRY
08AA: BD C6 B4          LDA $B4C6,X   GET TRACK NUMBER
08AD: F0 BA             BEQ EXIT      ALL FILES GOTTEN
08AF: 10 03             BPL DEL.FLG   NOT DELETED
08B1: A9 AD             LDA #$AD      HYPHEN FOR DELETED
08B3: 2C                .HS 2C        SKIP TWO BYTES
08B4: A9 A0     DEL.FLG  LDA #$A0     SPACE
08B6: 20 ED FD          JSR COUT      PUT INTO STRING
08B9: BD C8 B4          LDA $B4C8,X   TEST LOCK BIT
08BC: 48                PHA           SAVE FOR LATER
08BD: 10 03             BPL LOC.FLG   NOT.LOCKED
08BF: A9 AA             LDA #$AA      ASTERISK FOR LOCKED
08C1: 2C                .HS 2C        SKIP TWO BYTES
08C2: A9 A0     LOC.FLG  LDA #$A0     SPACE
08C4: 20 ED FD          JSR COUT      STORE IN STRING
08C7: 68                PLA           RESTORE TYPE
08C8: 29 7F             AND #$7F      STRIP OFF LOCK BIT
08CA: A0 07             LDY #7        CALCULATE FILE TYPE
08CC: 0A                ASL
08CD: 0A        SHIFTER  ASL
08CE: B0 03             BCS FOUND.TYPE
08D0: 88                DEY           NEXT TYPE
08D1: D0 FA             BNE SHIFTER   IF NOT TEXT
08D3: B9 A7 B3  FOUND.TYPE LDA $B3A7,Y GET TYPE
08D6: 20 ED FD          JSR COUT      STORE IT
08D9: 20 0C 09          JSR SPC.PRINT STORE A SPACE
08DC: BD E7 B4          LDA $B4E7,X   GET NUMBER OF SECTORS
08DF: 85 44             STA $44       FOR CONVERSION ROUTINE
08E1: 20 42 AE          JSR $AE42     NUMBER OF SECTORS
08E4: 20 0C 09          JSR SPC.PRINT ANOTHER SPACE
08E7: E8                INX           INDEX TO FILENAME
08E8: E8                INX
08E9: E8                INX
08EA: A0 1D             LDY #29       THIRTY CHARACTERS
08EC: BD C6 B4  FILER    LDA $B4C6,X   GET CHARACTER
08EF: 20 ED FD          JSR COUT
08F2: E8                INX
08F3: 88                DEY           ALL THIRTY?
08F4: 10 F6             BPL FILER     NOPE!
08F6: AE 9C B3          LDX $B39C     GET INDEX
08F9: BD C6 B4          LDA $B4C6,X   GET TRACK NUMBER
08FC: 20 ED FD          JSR COUT      SAVE IT
08FF: BD C7 B4          LDA $B4C7,X   GET SECTOR NUMBER
0902: 20 ED FD          JSR COUT      SAVE IT
0905: 20 30 B2          JSR $B230     GET NEXT ENTRY
0908: 90 9A             BCC DO.FILE   NO SECTOR CHANGE
090A: B0 91             BCS READ.NXT ..ALWAYS
090C: A9 A0     SPC.PRINT LDA #$A0    SPACE
090E: 4C ED FD          JMP COUT      STORE A SPACE

*-------------------------------------------------------------*
*                  PRINT A BUNCH OF STRINGS VIA USR           *
*-------------------------------------------------------------*

0911: A5 6B             LDA $6B       START OF ARRAYS
0913: 18                CLC
0914: 69 07             ADC #7        PLUS SEVEN
```

```
0916: 85 83             STA CUR.VAR
0918: A5 6C             LDA $6C       DO MSB
091A: 69 00             ADC #0        ADD CARRY
091C: 85 84             STA CUR.VAR+1
091E: A9 00             LDA #0        VTAB1
0920: 85 25             STA $25
0922: 20 24 FC          JSR VTAB
0925: 20 0C E1          JSR $E10C     GET INTEGER VALUE FROM FAC
0928: A5 A1             LDA $A1       GET NUMBER
092A: 38                SEC           FOR SUBTRACTION
092B: E9 09             SBC #9        NINE LESS
092D: B0 0F             BCS MULT1     EVERYTHING O.K.
092F: A9 08             LDA #8        NEW START VTAB
0931: 38                SEC           FOR SUBTRACTION
0932: E5 A1             SBC $A1
0934: 85 25             STA $25       VTAB THERE
0936: 20 24 FC          JSR VTAB
0939: 20 7A 09          JSR CLEOL
093C: A9 00             LDA #0
093E: 85 A1     MULT1   STA $A1
0940: 0A                ASL           TIMES 3
0941: 65 A1             ADC $A1
0943: 85 A0             STA $A0
0945: A4 A0     STRT.PRNT LDY $A0     OFFSET
0947: A6 25             LDX $25       CV
0949: C8                INY           GET ADDR
094A: B1 83             LDA (CUR.VAR),Y
094C: 85 FE             STA TEMP.PTR
094E: C8                INY
094F: B1 83             LDA (CUR.VAR),Y
0951: 85 FF             STA TEMP.PTR+1
0953: C8                INY
0954: 84 A0             STY $A0       FOR NEXT TIME
0956: A0 00             LDY #0        START AT BEGINNING OF STRING
0958: B1 FE     DO.STRNG LDA (TEMP.PTR),Y
095A: E0 09             CPX #9        ON VTAB 10?
095C: D0 02             BNE DOER      NOPE!
095E: 29 3F             AND #$3F      INVERSE IT
0960: 20 4D 08  DOER    JSR COUTR
0963: C8                INY
0964: C0 26             CPY #38
0966: 90 F0             BCC DO.STRNG
0968: 20 8E FD          JSR $FD8E     NEXT LINE
096B: E6 A1             INC $A1       COUNT PRINTINGS
096D: A5 A1             LDA $A1       PAST MAX?
096F: C5 FC             CMP COUNTER
0971: F0 07             BEQ CLEOL     YUP, EXIT
0973: A5 25             LDA $25       ON LINE TWENTY?
0975: C9 13             CMP #19
0977: 90 CC             BCC STRT.PRNT
0979: 60                RTS

097A: A0 26     CLEOL   LDY #38       THIRTY NINE SPACES
097C: A9 A0     P1      LDA #$A0      SPACE
097E: 91 28             STA (BASL),Y
0980: 88                DEY
0981: 10 F9             BPL P1
0983: 4C 8E FD          JMP $FD8E     RETURN

*-------------------------------------------------------------*
*                  TEST WHETHER A SECTOR IS FREE              *
*-------------------------------------------------------------*

0986: A9 00             LDA #0        ZERO ANSWER
0988: 85 FD             STA ANSWER
098A: 20 AB 09          JSR VTOC.BIT GET ADDR OF BYTE
098D: 39 F3 B3          AND $B3F3,Y   USED?
0990: D0 02             BNE RTS.2     NOPE!
0992: E6 FD             INC ANSWER    YUP
0994: 60        RTS.2   RTS

*-------------------------------------------------------------*
*                        FREE A SECTOR                        *
*-------------------------------------------------------------*

0995: 20 AB 09          JSR VTOC.BIT GET BYTE
0998: 19 F3 B3          ORA $B3F3,Y   NOW UNUSED!
099B: 99 F3 B3          STA $B3F3,Y
```

Continued on next page

Secondly, the last two characters of each string correspond to the track/sector list pointer of the file.

Print A Bunch Of Strings USR

This is the routine that is responsible for displaying a portion of the directory during most of LCA. It is designed to hook up to the USR command and occupies memory from $911 through $985. It assumes that the first array is the array that you wish to display. It evaluates the expression between the parenthesis in the USR command and puts that array element at the top of the list. VTAB position number 10 is inversed by the statements at $95A through $95F. A blank line is printed before the first element and after the last one which allows scrolling without leaving traces.

Test Whether A Sector Is Free

Spanning from $986 through $994, this routine examines the last used VTOC map (stored in DOS at location $B3BB) to test whether the sector specified by memory locations $FE and $FF is used or not. The answer is returned in memory location $FD.

Free A Sector

This routine which occupies memory from $995 through $99E alters the last used VTOC map (stored in DOS at location $B3BB) so that the sector specified by $FE and $FF is unused. This routine is used by the "DELETE SUBROUTINE" of the BASIC portion of LCA.

Allocate A Sector

This routine resides at $99F through $9AA and is similar to the above routine except that it marks the specified sector as used. It is used by the "EXHUME SUBROUTINE" of the BASIC portion of LCA.

Blank The Catalog

Spanning from $9C3 through $9DB, this routine is called just before the new directory is created. It zeroes memory from $8100 through $8F00. This is where the new directory is stored as it is created by the "BUILD DIRECTORY" subroutine of the BASIC portion of LCA. Each page of memory corresponds to a sector of the directory.

Enough About The ML. What About Updates?!?

In the future, I intend to update this program to make it function more quickly and be more user friendly. In the meantime, I'm hoping this program will prove helpful to you. I would like to hear from those readers who have ideas about what they would like to see happen to The Lone Catalog Arranger.

The Lone Catalog Arranger Hexdump

```
0800: 00 0D 08 0A 00 8C 32 30   $C2F1
0808: 36 33 3A AC 00 00 00 AD   $1468
0810: 81 C0 AD 81 C0 A9 D0 8D   $9578
0818: 21 08 8D 24 08 A0 00 B9   $9822
0820: 00 00 99 00 00 C8 D0 F7   $A9D0
0828: EE 21 08 EE 24 08 D0 EF   $D892
0830: A9 DD 85 67 A9 09 85 68   $075B
0838: 60 20 E3 03 20 D9 03 B0   $B174
0840: 01 60 AD F5 B7 4A 4A 4A   $0895
0848: 4A AA 4C 12 D4 C9 80 90   $DE1F

0850: 06 C9 A0 B0 02 29 3F 4C   $E1E6
0858: F0 FD 84 FB A4 FD 91 6F   $7F9D
0860: C8 84 FD D0 02 E6 70 A4   $F69B
0868: FB 60 84 FB E6 FC A9 00   $2F28
0870: 85 FD A4 FA A9 28 91 83   $81C9
0878: C8 A5 6F 38 E9 28 85 6F   $522D
0880: 91 83 A5 70 E9 00 85 70   $9F6E
0888: C8 91 83 C8 84 FA D0 D7   $3C45
0890: 00 00 A9 00 85 FA 20 DC   $4CBA
0898: AB 20 F7 AF 18 20 11 B0   $DDD2

08A0: B0 C7 A2 00 8E 9C B3 20   $012C
08A8: 6A 08 BD C6 B4 F0 BA 10   $A5E4
08B0: 03 A9 AD 2C A9 A0 20 ED   $3C77
08B8: FD BD C8 B4 48 10 03 A9   $790F
08C0: AA 2C A9 A0 20 ED FD 68   $F301
08C8: 29 7F A0 07 0A 0A B0 03   $F7B2
08D0: 88 D0 FA B9 A7 B3 20 ED   $D430
08D8: FD 20 0C 09 BD E7 B4 85   $955F
08E0: 44 20 42 AE 20 0C 09 E8   $2605
08E8: E8 E8 A0 1D BD C6 B4 20   $2FE1

08F0: ED FD E8 88 10 F6 AE 9C   $0317
08F8: B3 BD C6 B4 20 ED FD BD   $9051
0900: C7 B4 20 ED FD 20 30 B2   $0E80
0908: 90 9A B0 91 A9 A0 4C ED   $E482
0910: FD A5 6B 18 69 07 85 83   $8EEC
0918: A5 6C 69 00 85 84 A9 00   $BEA0
0920: 85 25 20 24 FC 20 0C E1   $80E7
0928: A5 A1 38 E9 09 B0 0F A9   $AE66
0930: 08 38 E5 A1 85 25 20 24   $4C8C
0938: FC 20 7A 09 A9 00 85 A1   $07AA

0940: 0A 65 A1 85 A0 A4 A0 A6   $6684
0948: 25 C8 B1 83 85 FE C8 B1   $264F
0950: 83 85 FF C8 84 A0 A0 00   $5971
0958: B1 FE E0 09 D0 02 29 3F   $0570
0960: 20 4D 08 C8 C0 26 90 F0   $B89F
0968: 20 8E FD E6 A1 A5 A1 C5   $C8A0
0970: FC F0 07 A5 25 C9 13 90   $9E21
0978: CC 60 A0 26 A9 A0 91 28   $60BE
0980: 88 10 F9 4C 8E FD A9 00   $AEF0
0988: 85 FD 20 AB 09 39 F3 B3   $6705

0990: D0 02 E6 FD 60 20 AB 09   $EF53
0998: 19 F3 B3 99 F3 B3 60 20   $7C48
09A0: AB 09 49 FF 39 F3 B3 99   $AA5C
09A8: F3 B3 60 A5 FE 0A 0A A8   $5103
09B0: A5 FF C9 08 90 03 E9 08   $E7EC
09B8: 24 C8 AA A9 00 38 2A CA   $F6ED
09C0: 10 FC 60 A9 81 8D CD 09   $9BDF
09C8: A9 00 A8 99 00 81 C8 D0   $C740
09D0: FA EE CD 09 AE CD 09 E0   $9BFF
09D8: 90 90 F0 60 00 00 00      $0735
```

Continued from previous page

```
099E: 60                RTS           DONE

*--------------------------------------------------------------*
*                        ALLOCATE A SECTOR                     *
*--------------------------------------------------------------*

099F: 20 AB 09          JSR VTOC.BIT GET BYTE
09A2: 49 FF             EOR #$FF     COMPLEMENT
09A4: 39 F3 B3          AND $B3F3,Y  NOW USED!
09A7: 99 F3 B3          STA $B3F3,Y
09AA: 60                RTS          DONE

09AB: A5 FE     VTOC.BIT LDA TRACK   BYTE=TRACK*4
09AD: 0A                 ASL
09AE: 0A                 ASL
09AF: A8                 TAY         FOR OFFSET
09B0: A5 FF              LDA SECTOR  GET SECTOR
09B2: C9 08              CMP #8      LESS THAN EIGHT?
09B4: 90 03              BCC NXT.1   YES, INY
09B6: E9 08              SBC #8      STRIP OFF BIT3
09B8: 24                 .HS 24      SKIP ONE BYTE
09B9: C8        NXT.1    INY         USE NEXT BYTE
09BA: AA                 TAX         SAVE SECTOR NUMBER IN X
09BB: A9 00              LDA #0      CALCULATE WHICH BIT STARTING WITH 0
09BD: 38                 SEC         BIT TO SHIFT THROUGH
09BE: 2A        ROL.1    ROL         SHIFT C THROUGH BYTE
09BF: CA                 DEX         DONE?
09C0: 10 FC              BPL ROL.1   NOPE!
09C2: 60                 RTS

*--------------------------------------------------------------*
*                        BLANK CATALOG                         *
*--------------------------------------------------------------*

09C3: A9 81              LDA #$81    CATALOG AT PAGE $81
09C5: 8D CD 09           STA BLANKER+2
09C8: A9 00              LDA #0      FILL WITH ZEROS
09CA: A8                 TAY         OFFSET ZERO
09CB: 99 00 81  BLANKER  STA $8100,Y STORE A ZERO
09CE: C8                 INY         PAGE DONE?
09CF: D0 FA              BNE BLANKER
09D1: EE CD 09           INC BLANKER+2 DONE?
09D4: AE CD 09           LDX BLANKER+2
09D7: E0 90              CPX #$90
09D9: 90 F0              BCC BLANKER NOPE!
09DB: 60                 RTS

09DC: 00                 .HS 00      EOL
09DD: 00 00     END.OBJ  .HS 0000
```

Bugs From Hardcore COMPUTIST No.'s 15 & 16

Hardcore COMPUTIST No. 15

Boot Code Trace For Tic Tac Show, pg. 25: Line 10 of the short HELLO program that appears in Step 14 should be as follows

```
10 ON PEEK (104)=96 GOTO 20 :
   POKE 104,96 : POKE 24576,0
   : PRINT CHR$ (4) "RUN
   ^HELLO"
```

Hardcore COMPUTIST No. 16

Rescue Raiders Softkey, pg. 6: The values in the Track column of the table in Step 4 should all be $0E (not $18 as shown in the last two lines). Likewise, the values in the Sector column should all be $0B (not $02 as shown in the last two lines).

The Controller Writer, pg. 17: The values in the last two lines of the Electronic Arts Controller screen dump should be

```
TRK-$02 SCTR-$03 BYTE-$47 TO-$AA
TRK-$02 SCTR-$03 BYTE-$51 TO-$AD
```

CORE

*see No. 20 - page 14 - Variation to work with Ver 3.2

Softkey For:

crossword magic

— Ver 3.0 —

By Paul Selby

Crossword Magic Version 3.0
L & S Computerware
1008 Stewart Drive
Sunnyvale, California 94086
(408) 738-3416
$49.95

Requirements:
48K Apple][Plus or equivalent
DOS 3.3 System Master
One Blank Disk

Crossword Magic is a fantastic program which will create a crossword puzzle using your words and clues. Once created, your puzzle may be played on the screen or a hardcopy version may be produced.

The Protection

The program is protected by encoding the disk in a non-standard format. A custom loader program takes care of reading the program from the disk and bringing it into memory. Since this program loads in all at once and does not need the original disk once it is loaded, it is a candidate for both a copy card copy or boot code tracing.

Stack Returns

One of the things which makes this program hard to trace is its liberal use of stack returns. Let me illustrate how a stack return works. Whenever a JSR (Jump to SubRoutine) is performed, two bytes are pushed on the stack (a section of memory used to hold temporary data). These two values represent the address to return to after the subroutine is finished being executed. The RTS (ReTurn from Subroutine) instruction causes the two values pushed on the stack to be fetched. These two values are then put together to form the return address. Keeping this in mind, one can see that it is possible for the programmer to put his own values on the stack and when a RTS is encountered, the microprocessor will pull the new values off the stack and use them as the return address. This may seem fairly straightforward but the programmer can make it very hard to figure out which numbers are pushed on the stack. Without knowing which values are placed there, it becomes almost impossible to trace the program flow.

To copy Crossword Magic, it is necessary to write a controller program which will monitor and modify the loader program. These modifications will take advantage of the stack returns by changing the values pushed on the stack. These values would then point back to the controller so that more values could be pushed on the stack and eventually jump to the monitor. In addition, the self volatile memory areas have to be relocated to a safe position.

The Procedure

1) Boot System Master to load DOS and then clear memory

PR#6
FP

2) Insert blank disk to create a slave disk

INIT HELLO

3) The following program consists of two parts. The first part is the controller (responsible for loading in the program), and the second is the code relocator (responsible for placing the packed Crossword Magic code *back* into its proper locations). Type it in

CALL -151

```
1E00: A9 1E 8D 50 08 A9 0C 8D    $A22D
1E08: 52 08 4C 01 08 A2 00 BD    $C996
1E10: 00 02 9D 00 1F E8 D0 F7    $C19C
1E18: A9 59 8D 2B 05 A9 FF 8D    $C4A9 32 BB
* 1E20: 2C 05 4C 00 05 A2 00 BD  $722B 32 BB
1E28: 00 1F 9D 00 02 E8 D0 F7    $0508
1E30: BD 00 20 9D 00 80 E8 D0    $8627
1E38: F7 EE 32 1E EE 35 1E AD    $4AD0
1E40: 35 1E C9 C0 D0 EA 4C 3B    $F3B3
1E48: B7                         $E228
```

4) Check your typing against this disassembly:

```
1E00-  A9 1E     LDA  #$1E
1E02-  8D 50 08  STA  $0850
1E05-  A9 0C     LDA  #$0C
1E07-  8D 52 08  STA  $0852
1E0A-  4C 01 08  JMP  $0801
1E0D-  A2 00     LDX  #$00
1E0F-  BD 00 02  LDA  $0200,X
1E12-  9D 00 1F  STA  $1F00,X
1E15-  E8        INX
1E16-  D0 F7     BNE  $1E0F
1E18-  A9 59     LDA  #$59
1E1A-  8D 2B 05  STA  $052B
1E1D-  A9 FF     LDA  #$FF
1E1F-  8D 2C 05  STA  $052C
1E22-  4C 00 05  JMP  $0500
1E25-  A2 00     LDX  #$00
1E27-  BD 00 1F  LDA  $1F00,X
1E2A-  9D 00 02  STA  $0200,X
1E2D-  E8        INX
1E2E-  D0 F7     BNE  $1E27
1E30-  BD 00 20  LDA  $2000,X
1E33-  9D 00 80  STA  $8000,X
1E36-  E8        INX
1E37-  D0 F7     BNE  $1E30
1E39-  EE 32 1E  INC  $1E32
1E3C-  EE 35 1E  INC  $1E35
1E3F-  AD 35 1E  LDA  $1E35
1E42-  C9 C0     CMP  #$C0
1E44-  D0 EA     BNE  $1E30
1E46-  4C 3B B7  JMP  $B73B
```

Continued on next page

Note
To Our Readers:

(REPRINT FROM ISSUE No. 5)

Several of our readers have called or written to ask what assembler was used to produce source code listings printed in Hardcore COMPUTIST. The answer to this question is that we use the S-C Macro Assembler from the S-C Software Corporation (2331 Gus Thomasson, Suite 125, PO Box 280300, Dallas, TX 75228, Cost: $80.00).

To convert the source files published in our magazine to a format that is compatible with the particular assembler that you use, consult the following list of S-C Assembler directives:

S-C Assembler Directives:

.OR - ORigin. This sets the address of program origin to the value of this expression.
Example: **.OR $300**

.TA - Target Address. This sets the location or Target address at which the object code will be placed during assembly.
Example: **.TA $4000**

.TF - Target File. This directive causes the object code to be stored to disk rather than in memory during assembly.
Example: **.TF CHECKERS GAME**

.IN - INclude. This causes the contents of the specified source file to be included in the assembly.
Example: **.IN CHECKER BOARD**

.EN - ENd of program. This is an optional directive which indicates the end of the source code to be assembled.

.EQ - EQuate. Defines a label to have the value of the expression.
Example: **COUT .EQ $FDED**

.DA - DAta. Creates constants or variables in the program.

.HS - Hexadecimal String. This directive converts a string of hex characters to binary and stores them at the current location in memory.

.AS - ASCII String. Stores the binary equivalent of the ASCII characters in quotes.
Example: **.AS "APPLE II"**

.AT - ASCII Terminated. This operates the same as .AS except that the high-order bit of the last character in the string is set opposite to that of the preceding characters.

.BS - Block Storage. This reserves a specified number of bytes for storage.
Example: **.BS 6**

.TI - TItle. This is used to print a program title and page number at the beginning of each page during assembly.

.LI - LIst control. Controls whether a program listing will be generated during assembly. The listing can be either turned on or off.
Example: **.LI OFF**

.MA - MAcro definition. Beginning of Macro definition.

.EM - End Macro definition.

.US - USer directive. Allows the user to indirectly jump to a set of user-supplied instructions.

.PG - PaGe control. Prints an ASCII form feed character during assembly.

.DO - DO conditional assembly. If the value of the expression following the DO statement evaluates true, then the code up to the .FIN directive will be assembled. Otherwise, it will be ignored.
Example:

```
FLAG    .EQ 1
        .DO FLAG
```

.FIN - FINish conditional assembly. Indicates the end of the code to be assembled under conditional assembly.

.ELSE - This is used to introduce an IF-THEN-ELSE structure into portions of source code affected by conditional assembly.

In addition, the S-C Assembler uses a pound sign (#) to indicate the lower byte of a label's address and a slash (/) to indicate the upper byte of a label's address.

Example:
```
LDA #COUT
LDX /COUT
```

We are now using a standard version of the S-C Assembler but, in the past, several source files produced by a custom version of the assembler managed to slip by our staff and were published in Hardcore. As a result, some source code listings contained BGE (Branch on Greater or Equal) and/or BLT (Branch on Less Than) instructions. To make these source files compatible with your assembler, convert BGE to BCS and BLT to BCC.

Continued from previous page

5) Save program in case you make a mistake

BSAVE MAGIC 1,A$1E00,L$49

6) The initial boot code is a program stored on a PROM on the disk interface card that reads track 0 sector 0 into memory at $800-$8FF and then jumps to it. Because it is stored on a PROM, it cannot be modified. To remedy this, the code will be moved into RAM where it can be modified

6600<C600.C6F8M

7) The jump out of the bootstrap loader normally goes to $801. The following change will make control jump to $1E00, the loader program

66F9:00 1E

8) Insert Crossword Magic disk and execute the boot

6600G

9) At this point the computer will be in the monitor and the drive will be spinning. The following command will stop the drive to prevent wear on the disk

C0E8

10) The Crossword Magic program actually uses track $10 sector $0 as the VTOC (the disk map of what parts of the disk are occupied by programs) and also stores the printer data here. To correct this situation, the Crossword Magic DOS must be patched so that it will use the normal track $11 and sector $0 for the VTOC. This can be accomplished by changing the value at $AC01 from a previous value of $10 to a $11

AC01:11

During the remainder of the procedure, to compress Crossword Magic so that it will take up less disk space, the following memory locations will be relocated:

Original Location	Relocated Location	Length
$0200-02FF	$1F00-1FFF	$0100
$0800-1DFF	$0800-1DFF	$0600
$6000-7FFF	$6000-7FFF	$2000
$8000-9AFF	$2000-3AFF	$1B00
$9D00-BFFF	$3D00-5FFF	$2300

11) Next, relocate most of DOS so that it won't be overwritten on the next boot

2000<8000.BFFFM

12) The code at $800-$8FF must be moved to a temporary place so it will be safe when the slave disk is booted (this area is also overwritten when a boot is performed)

3B00<800.8FFM

13) Boot the slave disk so the program can be saved

6CTRLP

* see No. 20, page 6 - comment (letter)

14) Move back the code which was at $800-$8FF

CALL-151
800<3B00.3BFFM

15) Now set up a location so that when the program is BRUN it will jump to $1E25 where the code relocator resides

7FD:4C 25 1E

16) Save the CROSSWORD MAGIC program

BSAVE MAKER,A$7FD,L$7803

17) Although the CROSSWORD MAGIC program has been copied, the files which support the program must also be copied. There are three files on the disk:

1. USER*
2. Z<CTRL Z>
3. CROSSWORD MAGIC DEMO

The first file is the user-written printer dump program, the second file holds the names of the puzzles, and the last file is the demo puzzle.

Use FID on the system master to copy these files to the slave disk.

Getting The Other Side

Although the player side can be copied by COPYA, it can also be copied by boot code tracing. By doing this, the player and the maker can occupy the same disk with room to spare.

18) Enter the following program:

```
1800: A9 18 8D 0B 08 A9 0C 8D    $CD79
1808: 2C 08 4C 01 08 A9 1A 8D    $BD93
1810: B8 04 A9 18 8D B9 04 4C    $0AB9
1818: 7E 04 4C 59 FF A2 00 BD    $E025
1820: 00 19 9D 00 60 E8 D0 F7    $521D
1828: EE 21 18 EE 24 18 AD 24    $146F
1830: 18 C9 68 D0 EA BD 00 21    $C865
1838: 9D 00 7F E8 D0 F7 EE 37    $2A6B
1840: 18 EE 3A 18 AD 3A 18 C9    $A81A
1848: 96 D0 EA BD 00 38 9D 00    $8B27

1850: 9D E8 D0 F7 EE 4D 18 EE    $82E6
1858: 50 18 AD 50 18 C9 C0 D0    $A785
1860: EA 4C 3B B7                $7EDF
```

19) Check your typing against this listing.

```
1800-   A9 18      LDA   #$18
1802-   8D 0B 08   STA   $080B
1805-   A9 0C      LDA   #$0C
1807-   8D 2C 08   STA   $082C
180A-   4C 01 08   JMP   $0801
180D-   A9 1A      LDA   #$1A
180F-   8D B8 04   STA   $04B8
1812-   A9 18      LDA   #$18
1814-   8D B9 04   STA   $04B9
1817-   4C 7E 04   JMP   $047E
181A-   4C 59 FF   JMP   $FF59
181D-   A2 00      LDX   #$00
```

Continued on next page

Whiz Kid by Ray Darrah

page 26

The following article is a continuation of the Whiz Kid article which appeared in Hardcore COMPUTIST No. 15. If you haven't read that article, I suggest you do so immediately. If you have read the article, you may want to read it again to refresh your memory.

As demonstrated by the BASIC program in the previous article, moving the disk arm is a simple procedure. Because of the relative nature of disk arm movement, the current phase that the disk arm is over must be known. DOS uses several memory locations on text page 1, that aren't displayed in the 40 column by 24 row display, to keep track of the current phase of every disk arm connected to your computer.

To move the disk arm, you should follow these steps:

1) Turn the drive on.
2) Turn on an adjacent magnet (the magnet turned on depends on which way the head is to be moved).
3) Wait a while so that the shaft will become aligned.
4) If you are not at the destination phase then go to Step 2.

BASIC Movements

The most crucial step in the head movement algorithm is the delay routine that waits while the shaft of the stepper motor aligns itself with one of the magnets. DOS has a table of values at $BA11-$BA28 that are used by the SEEKABS routine at $B9A0 to generate these delays. To speed up the movement of the head, this table contains values that generate delays which are inversely proportional to the velocity of the motor's shaft. In other words, the first wait that DOS performs is the longest and the wait is reduced for each new magnet. Using this method, DOS moves the drive head at the fastest possible rate.

The BASIC program from the last issue can reliably move the drive head because the BASIC interpreter is so slow that no delay routine is necessary.

The following is an explanation of how this BASIC program functions:

Line 10 clears the screen, puts the drive slot number in SL and puts the drive number in DR.

Line 20 puts the current phase in CP and puts the destination phase in DP.

Line 30 puts the beginning address of the control registers for the specified drive in AD, turns on the power to the drive(s) by referencing register 9 ($C0n9) and engages the proper drive by referencing either register 10 or 11 ($C0nA or $C0nB) depending on the setting of DR.

Line 40 compares the current phase to the destination phase. If they are the same it exits via line 110.

Line 50 determines which direction the drive head has to move by comparing the destination phase to the current phase. If the drive head has to move toward higher tracks, then execution continues at line 80. Otherwise, execution continues at line 60.

Line 60 sets up a loop to count downwards, by one, from one more than twice the current phase to twice the destination phase.

Line 70 references the next register to either turn on or off the current magnet, gets the next element in the loop and, when done, exits via line 110.

Line 80 sets up a loop to count upwards, by twos, from twice the current phase plus one to twice destination phase plus one.

Line 90 references the on register to determine which magnet the stepper motor shaft is to be aligned with next.

Line 100 references the off register for the magnet that was turned on in line 90.

Line 110 references the motor off register for the specified drive and, therefore, turns off the drive and ends.

With a couple of changes, this program could easily be incorporated as a subroutine of a larger program that for some reason must manipulate the disk drive arm without going through DOS. Note: If you compile this program it probably will no longer work. This is because it depends upon the Applesloth interpreter for the necessary delays.

Quarter Tracks

* In my opinion, quarter tracks aren't worth much. DOS only writes on the even phases of the disk because the resolution of the drive head and the accuracy of the stepper motor are too poor to ensure accurate data on adjacent phases.

A quarter track is accessed by turning on two adjacent magnets in the stepper motor almost simultaneously. Since the shaft permanent magnet will be attracted to two energized magnets, it will be positioned in the middle of them, thus in-between phases. If data written on phase 0 will overwrite data written on phase 1, then imagine how much overlap there is between phase 0 and phase 0.5!

Continued from previous page

```
181F-  BD 00 19   LDA   $1900,X
1822-  9D 00 60   STA   $6000,X
1825-  E8         INX
1826-  D0 F7      BNE   $181F
1828-  EE 21 18   INC   $1821
182B-  EE 24 18   INC   $1824
182E-  AD 24 18   LDA   $1824
1831-  C9 68      CMP   #$68
1833-  D0 EA      BNE   $181F
1835-  BD 00 21   LDA   $2100,X
1838-  9D 00 7F   STA   $7F00,X
183B-  E8         INX
183C-  D0 F7      BNE   $1835
183E-  EE 37 18   INC   $1837
1841-  EE 3A 18   INC   $183A
1844-  AD 3A 18   LDA   $183A
1847-  C9 96      CMP   #$96
1849-  D0 EA      BNE   $1835
184B-  BD 00 38   LDA   $3800,X
184E-  9D 00 9D   STA   $9D00,X
1851-  E8         INX
1852-  D0 F7      BNE   $184B
1854-  EE 4D 18   INC   $184D
1857-  EE 50 18   INC   $1850
185A-  AD 50 18   LDA   $1850
185D-  C9 C0      CMP   #$C0
185F-  D0 EA      BNE   $184B
1861-  4C 3B B7   JMP   $B73B
```

20) Save this second controller program

BSAVE MAGIC 2,A$1800,L$64

21) Move boot code into RAM

6600<C600.C6F8M

22) Change jump to point to controller

66F9:00 18

23) Insert the Crossword Magic player disk and execute its boot

6600G

24) Stop drive

C0E8

25) Patch DOS

AC01:11

For this side of the disk, memory will be compressed in the following manner:

Original Location	Relocated Location	Length
$0800-17FF	$0800-17FF	$0800
$6000-67FF	$1900-20FF	$0800
$7F00-95FF	$2100-37FF	$1700
$9D00-BFFF	$3800-5AFF	$2300

26) Compress the code

1900<6000.67FFM
2100<7F00.95FFM
3800<9D00.BFFFM
6000<800.8FFM

27) Boot slave disk

6^{CTRL}P

28) Move the $0800 code back

CALL -151
800<6000.60FFM

29) Set up jump at the beginning of the program

7FD:4C 1D 18

30) Save PLAYER program

BSAVE PLAYER,A$7FD,L$~~4A05~~ 5303

We're Done

This completes the softkey for CROSSWORD MAGIC. For those of you who would like to copy the logo, all that has to be done is to boot the CROSSWORD MAGIC disk and hit RESET the moment the logo appears, boot the slave disk, and type

BSAVE LOGO.MAKER,A$900,L$5B5

The same applies to the player logo except you must type

BSAVE LOGO.PLAYER,A$1400,L$5B0

To display the logos, just BRUN them. I also recommend the use of a fast DOS to speed up the loading time.

BACKUP PROTECTED SOFTWARE WITH COPY II PLUS™ 5.0

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

- Fully automatic bit copy. All parameters are stored on disk and are revised quarterly. Simply type in the name of the program you wish to backup, and **COPY II PLUS** does the rest!
- New utilities include Alphabetize Catalog, Fast 2-pass Disk Copy on a *IIc* or *IIe*, and an all-new Sector Editor.
- Supplied on a standard DOS diskette. Not copy protected, of course.

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

Available at your local dealer or direct from us.
Current owners may update for $20 with the return of your original disk.

CENTRAL POINT Software, Inc.

ONLY **$39.95**
(Plus $3 Shipping & Handling)

9700 S.W. Capitol Highway, #100/Portland, OR 97219

(503) 244-5782
M-F 8-5:30 (W. Coast Time)

MasterCard VISA **CHECK WELCOME**

(Prepayment Required)
Backup Utilities Also Available For
IBM PC, MACINTOSH, and COMMODORE 64

These products are provided for the purpose of enabling you to make archival backups only.

Public Domain Software

SPECIAL

From The COMPUTER LEARNING CENTER!

GET Over 170

BUSINESS & FINANCE

Programs FOR ONLY

$28.00

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

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

Annuity	Real Estate Plot	Savings Growth
Bond Price & Interest	Visicalc Formulas	Treasury Bill Valuation
Budget Monthly	Business Finance	Address File
Decision Matrix	Check Stub	Compound Int. Tables
Financial Pak	Household Exp. Profile	File Manager
Keogh Savings Program	Income Tax 1040 For 77	Home Accounting
Mortgage Calculation	Inventory Company	Names Search
Regular Deposits I	Average Growth Rate	File Cabinet I
Stocks	Check Book Balancer	Family Finance
Calendar Personal	Income Taxes	Gen Ledger Printer
Letter Writer	Interest Earned	Constar
Phone List	Loan Balance	etc...

ORDER NOW!

You will also receive a complete catalog of Public Domain Software available from the CLC with your order.

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

Name _____ S-4

Address _____

City _____ St _____ Zip _____

Country _____ Phone _____

VISA/MC _____-_____-_____-_____ Exp _____

Signature _____

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

(The Computer Learning Center's PDS Library contains over 175 volumes, which will run on Apple][Plus computers and Apple compatibles. Most will also run on Apple //e and //c.)

Now-the ultimate back-up system!

EDD III® and TRAK STAR™

COMPLETE PACKAGE:

159.95*

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

PRICED SEPARATELY:

TRAK STAR 99.95

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

Documentation: $3 Refundable with the purchase of TRAK STAR

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

Save copying time with nibble programs

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

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

Midwest Microsystems

To order, phone:
913 676-7242

9071 Metcalf / Suite 124
Overland Park, KS 66212

NOT PIRACY
Just Good Sense!

SUBSCRIBE to Hardcore COMPUTIST and...

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

Annual Subscription Rates:

Please check one

☐	U.S.	$25
☐	Canada, U.S. 1st Class	$34
☐	Mexico	$39
☐	Foreign Airmail	$60
☐	Foreign surface mail	$40
☐	SAMPLE, U.S.	$3.50
☐	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 _____ Phone _____ VISA/MC _____-_____-_____-_____ Exp _____ Signature _____

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

Are you a NEW SUBSCRIBER?

BACK ISSUES of Hardcore COMPUTIST and ★ CORE are PACKED with information that you won't want to miss.

Hardcore COMPUTIST 16: Softkeys for Rescue Raiders, Sheila, Basic Building Blocks, Artsci Programs, Crossfire, Sensible Speller for ProDOS and Sideways / Secret Weapon: RAMcard / The Controller Writer / A Fix For The Beyond Castle Wolfenstein Softkey / The Lone Catalog Arranger Part 1

Hardcore COMPUTIST 15: Softkeys for Mastertype, Stickybear BOP, Tic Tac Show, The Financial Cookbook, Escape From Rungistan, Alien Munchies, Millionaire & Plato / MREAD/MWRT Update / Cumulative Index to Hardcore Publications: 1981-1984 / A Boot From Drive 2 / DB Master's Data Compression Techniques

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

Hardcore COMPUTIST 13: Softkeys for Laf Pak, Beyond Castle Wolfenstein, Transylvania and The Quest, Electronic Arts, Snooper Troops (Case 2), DLM Software, Learning With Leeper, & TellStar / CSaver: The Advanced Way to Store Super IOB Controllers / Adding New Commands to DOS 3.3 / Fixing A ProDOS 1.0.1 BSAVE Bug / REVIEW: Enhancing Your Apple / Locksmith 5.0 and the Locksmith Programming Language.

Hardcore COMPUTIST 12: Softkeys for Zoom Grafix, Flip Out, Lion's Share, Music Construction Set, Hi-Res Computer Golf II, Suicide, Sabotage, Millionaire, Time Is Money, & Type Attack, Psychedelic Symphony / The CORE Disk Searcher / The Armonitor / Pseudo-ROMs on the Franklin Ace.

Hardcore COMPUTIST 11: Copy II Plus 4.4C Update / PARMS for Essential Data Duplicator / Ultimaker III / Mapping of Ultima III / Ultima II...The Rest of the Picture / Softkeys for Sensible Speller, Ultima III, Softporn Adventure, The Einstein Compiler v5.3, & Mask of the Sun.

Hardcore COMPUTIST 10: Controller Saver / Softkeys for The Arcade Machine, Bankstreet Writer, Minit Man, Sensible Speller IV, Essential Data Duplicator III, Krell LOGO, & Canyon Climber, ApplEar / REVIEWS: The 65SC802 & 65SC816 Chips and Dino Eggs / Examining Protected Applesoft Basic Programs / Crunchlist II / Parms for DB Master v4.

Hardcore COMPUTIST 4: Ultima II Character Editor / Softkeys for Ultima II, Witness, Prisoner II, & Pest Patrol / Adventure Tips for Ultima II & III / Copy II Plus PARMS Update.

Hardcore COMPUTIST 1: Softkeys for Data Reporter, Multiplan & Zork / PARMS for Copy II Plus / No More Bugs / APT's for Choplifter & Cannonball Blitz / Reviews: Replay, Crackshot, Snapshot & Wildcard copy cards.

CORE 3 Games: Constructing Your Own Joystick / Compiling Games / GAME REVIEWS: Over 30 of the latest and best / Pick Of The Pack: All-time TOP 20 games / Destructive Forces / EAMON / Graphics Magician and GraFORTH / and Dragon Dungeon.

CORE 2 Utilities: Dynamic Menu / High Res: Scroll Demo / GOTO Label: Replace / Line Find / Quick Copy: Copy.

CORE 1 Graphics: Memory Map / Text Graphics: Marquee, Boxes, Jagged Scroller / Low Res: Color Character Chart / High Res: Screen Cruncher, The UFO Factory / Color / Vector Graphics:Shimmering Shapes, A Shape Table Mini-Editor / Block Graphics: Arcade Quality Graphics for BASIC Programmers / Animation.

(★CORE is no longer published as an independent quarterly magazine. Back issues not listed are no longer available.)

Send me the back issues indicated below:

HC 16	$3.50 ☐	HC 11	$3.50 ☐
HC 15	$3.50 ☐	HC 10	$3.50 ☐
HC 14	$3.50 ☐	HC 4	$3.50 ☐
HC 13	$3.50 ☐	HC 1	$3.50 ☐
HC 12	$3.50 ☐	CORE 1 Graphics	$3.50 ☐
		CORE 2 Utilities	$3.50 ☐
		CORE 3 Games	$3.50 ☐

Name _____ ID# _____

Address _____

City _____ St _____ Zip _____

Phone _____

VISA/MC _____-_____-_____-_____ Exp _____

Signature _____

Send check or money order to: Hardcore COMPUTIST, PO Box 110846-B, Tacoma, WA 98411. Most orders shipped UPS. Please use street address. Washington residents add 7.8% sales tax. Foreign orders add 20% shipping and handling. US funds drawn on US bank.

DISKBUSTERS ALERT!

Want to show your friends
just what you think of software COPY-PROTECTION?

Wear the *ORIGINAL* Hardcore COMPUTIST Diskbusters t-shirt!

Diskbusters

*Hardcore COMPUTIST 1984

Join The "OFFICIAL" DISKBUSTERS team!!!

Don't miss this opportunity! Order the ORIGINAL Diskbusters t-shirt TODAY and you'll never again be "just one of the crowd". Let the world know that YOU are a member of the DISKBUSTERS team of Hardcore COMPUTIST, the magazine for the serious user of Apple][computers.

Tell your friends! Tell your enemies! (Maybe even tell your folks!) They'll want to know how they, too, can become a member of the team.

Hardcore COMPUTIST says:
"We ain't afraid of no disk,"
and YOU CAN, TOO!

ORDER TODAY!

Use your VISA/MC, personal check or money order

Available in adult sizes only (black & red on gray).

Only
$9.95

* Hanes 50% cotton/ 50% polyester

Rush me _____ (total) DISKBUSTERS T-shirts in the sizes indicated below. I have enclosed $9.95 (plus applicable tax & shipping) for each shirt.

ADULT MENS: ___Small ___Med. ___Large ___X-Large

Name _____

Address _____

City _____ St _____ Zip _____

Phone _____

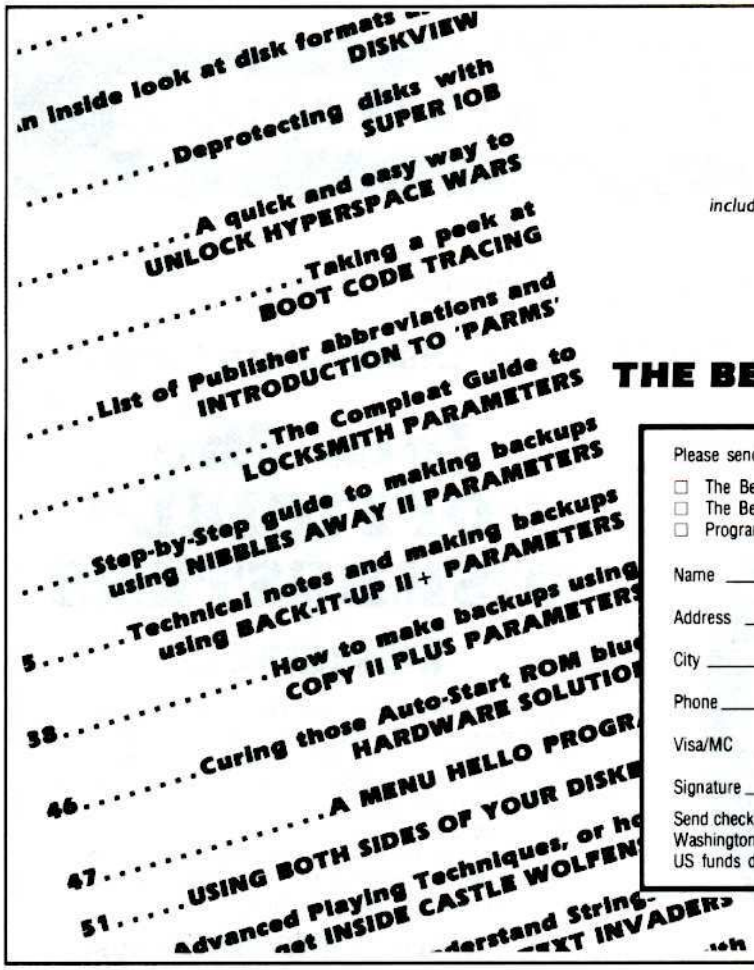
VISA/MC _____-_____-_____-_____ Exp _____

Signature _____

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

Hardcore
COMPUTIST

PO Box 110816
Tacoma, WA 98411



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

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 _____ ID#_____

Address _____

City _____ St _____ Zip _____

Phone _____

Visa/MC _____-_____-_____-_____ Exp _____

Signature _____

Send check or money order to: Hardcore COMPUTIST, PO Box 110846-B, Tacoma, WA 98411. Washington State residents add 7.8% sales tax. Foreign orders add 20% shipping & handling. US funds drawn on US bank.

10 Diskettes (SS/SD for the Apple)
PLUS case: $13.00

Plastic disk case in your choice of color!

Send me the following: (identify number and type of disk sets)

_____ SS/SD _____ SS/DD _____ DS/DD

Name _____

Address _____

City _____ St _____ Zip _____

Phone _____

VISA/MC _____-_____-_____-_____ Exp _____

Signature _____

Choose 1 case per disk set from these available colors:

_____ yellow _____ red _____ green _____ beige _____ blue _____ black _____ grey

For Apple only SS/SD ..... $13 per set

For Apple, IBM, Commodore
SS/DD ..... $14 per set

IBM only DS/DD ..... $16 per set

(10 5 1/4 inch diskettes (with Tyvek sleeves) per set. Hub rings. Guaranteed 100% certified, full surface tested.)

Send check, money order, VISA/MC with signature and exp date to: SoftKey Publishing, PO Box 110816, Tacoma, WA 98411. Enclose $3 shipping and handling. Washington residents add 7.8% sales tax. US funds drawn on US bank. For faster service, send postal money order or certified check. Orders shipped via UPS. Please use street address.

FREE shipping on 3 or more sets

By Hackers
For Hackers

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

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

The BOOT-LEGGER
MAGAZINE

Subscribe Now!

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

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

SIMPLY·SOFTWARE

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

LIST $24.95
Fracas
Minicrossword

LIST $29.95
Alphabet Zoo
Castle Wolfenstein
Compu-Read
Dunzhin
Early Games for Y.C

LIST $34.00
Alien Addition
Alligator Mix

LIST $34.95
BC's Quest for Tires
Beyond Castle Wolfenstein
Bruce Lee
Championship Lode Runner
Earthquake: San Francisco 1906
Frogger
Lode Runner
The Quest

LIST $39.95
Algebra 1
Algebra 3
Aplus
Broadsides
Copy II +
Face Maker
Master Type

SPECIAL $17.47
Sword of Zedek
Typing Tutor II

SPECIAL $20.97
The Eternal Curse
Sea Fox
Learning with Leeper
Jellyfish
Study Quiz Files

SPECIAL $23.80
Demolition Division
Minus Mission

SPECIAL $24.47
Murder by the Dozen
Night Mission Pinball
Hayden Reversal
Sea Dragon
Star Maze
Story Machine
Transylvania
Wizard of Id's Wiztype

SPECIAL $27.97
Seastalker
Spellicopter
Spelling Bee Games
Triple Dump
Type Attack
Zaxxon
Zork I, II or III

LIST $44.95
DaVinci: Interiors
Snooper Troops #1

LIST $49.95
Algebra 5 & 6
Clickart
Clickart Publications
DaVinci: Building
Deadline
Enchanter
Flight Simulator II
Letter Wizard
Lexicom
Math Blaster!

LIST $59.95
Graphics Magician
Megaspell
Millionaire

LIST $69.95
Bank Street Speller
Bank Street Writer

LIST $89.95
Grapics Application System

LIST $100.00
Chart
Megamerge
PFS: File

SPECIAL $31.47
Snooper Troops #2
States & Traits

SPECIAL $34.97
Planetfall
Print Shop
Questron
Run for the Money
Sargon II
Sorcerer
Typing Tutor III
Witness
Wizardry
Word Attack

SPECIAL $41.97
Reading Machine
S.A.M.
Ultima III

SPECIAL $48.97
Homeward
Speed Reader II

SPECIAL $62.97
Study Prog. for SAT
Systems Saver

SPECIAL $87.50
PFS: Graph
PFS: Report
PFS: Write
Sensible Speller

	LIST	SPECIAL
Alpha Plot	$ 39.50	$ 27.65
The Antonym Game	$ 39.50	$ 27.65
Basic Interpreter	$150.00	$105.00
Basic 3D Graphics	$ 40.00	$ 28.00
Bookends	$124.95	$ 87.47
BPI General Ledger	$395.00	$276.50
dBase II	$495.00	$346.50
DB Master 4.0	$350.00	$245.00
Dollars & Sense	$100.00	$ 70.00
Es-cape	$ 60.00	$ 42.00
Estate Tax Plan	$750.00	$525.00
The Filer	$ 19.95	$ 13.97
The General Manager	$229.95	$160.97
Home Accountant	$ 74.95	$ 52.47
Homewriter	$ 50.00	$ 35.00
Linkindex	$195.00	$136.50
Mac the Knife	$ 39.00	$ 27.30
Mach III Joystick	$ 54.95	$ 38.47
Medical Insurance Form Writer	$100.00	$ 70.00
Micro Cookbook	$ 40.00	$ 28.00
Micro DSS/Finance	$795.00	$556.50
Multipan	$195.00	$136.50
Sideways	$ 60.00	$ 42.00
Summer Games	$ 40.00	$ 28.00
Telengard	$ 28.00	$ 19.60
Think Tank	$150.00	$105.00
Ultraplan	$169.00	$118.30
Visicalc	$250.00	$175.00
Visitrend/Visiplot	$ 99.00	$ 69.30
Wordstar	$495.00	$346.50
Z-term 'the Professional'	$149.95	$104.97

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.

DISCOUNTS!

5¼ DISKETTES & STORAGE

- SS/DD, BOX OF 10 — **$12.00**
- SS/DD, 10 BOXES — **$115.00**
- DOUBLE-NOTCHED DS/DD, EACH — **$1.35**
- DOUBLE-NOTCHED DS/DD, 100 — **$125.00**
- HARD PLASTIC STAND-UP DISKETTE LIBRARY CASES — **$2.75 EACH, 4 FOR $10.00**
  (specify color choices: beige, black, blue, green, grey, red, yellow)
- SMOKED PLASTIC JUMBO-SIZE FLIP-TOP 75 DISKETTE FILE CASES — **$16.00** *
- 70-DISKETTE FILE CASES — **$12.00** *
- 140-DISKETTE LOCKING WOOD FILE CABINET — **$33.00**

PRINTERS

- PANASONIC P1090 — **$239.00**
- PANASONIC 1091 — **$299.00**
- EPSON RX-80 F/T — **$289.00**
- EPSON RX-100 — **$389.00**
- EPSON FX-80 — **$379.00**
- OKI-DATA MICROLINE 92A DOT MATRIX — **$369.00**
- SILVER REED 400 LETTER QUALITY — **$269.00**
- STARWRITER A-10 25CPS LETTER QUALITY — **$495.00**
- TOSHIBA 1340 DOT MATRIX AND LETTER QUALITY COMBINED — **$795.00**

PRINTER INTERFACES AND ACCESSORIES

- STANDARD PARALLEL INTERFACE CARD — **$49.00**
- GRAPHICS PARALLEL INTERFACE CARD — **$75.00**
- FINGERPRINT PUSH-BUTTON GRAPHICS CARD — **$119.00**
- MICROFAZER GENERAL PRINT BUFFER — **$149.00**
- PRINTER STAND — **$14.00**
- SWITCH BOX 3 PARALLEL PORTS — **$129.00**
- SWITCH BOX 3 SERIAL PORTS — **$79.00**

FLOPPY DISK DRIVES

- FOURTH DIMENSION (FULL OR SLIMLINE) — **$179.00**
- DISTAR — **$159.00**
- LASER — **$139.00** *
- IIc DRIVES — **$158.00** *
- DISK CONTROLLER — **$59.00**
- DOUBLE-SIDED DRIVE — **$199.00**
- 650K RANA DRIVE — **$439.00**

HARD DISK DRIVES

- 5 MEGABYTE WITH CONTROLLER AND SOFTWARE — **$749.00**
- 10 MEGABYTE — **$1175.00**

MONITORS

- GORILLA 12-INCH GREEN — **$84.00**
- USI 12-INCH GREEN — **$94.00** *
- USI 12-INCH AMBER — **$99.00** *
- INTRA 14-INCH COMPOSITE COLOR/80 COLUMN — **$239.00**

MODEMS

- ZOOM TELEPHONICS 300-BAUD — **$109.00**
- CENTAURI 300 BAUD — **$179.00**
- PRO-MODEM 1200 — **$349.00**
- PRO-MODEM 1200A INTERNAL — **$279.00**
- SINGALMAN MARK XII — **$239.00**

GRAPHICS DEVICES

- POWER PAD & STARTER KIT — **$99.00**

VIDEO & DISPLAY EQUIPMENT

- DIGITIZER — **$299.00**
- B & W CAMERA — **$195.00**
- COLOR PROCESSOR — **$99.00**
- COLOR PROCESSOR/ENHANCER STABILIZER/SYNTHESIZER — **$279.00**

GENERAL ITEMS

- 6 OUTLET POWER STRIP — **$19.00**
- SURGE PROTECTOR — **$11.00**
- RF MODULATOR — **$49.00**
- COMPUTER STAND — **$24.00**

GAME I/O DEVICES

- CH MACH II JOYSTICK — **$37.00**
- CH MACH III JOYSTICK — **$45.00**

★ DENOTES NEW PRICE OR ITEM

SLOT EXPANSION

- 16 RAM CARD — **$49.00**
- 64K RAM & 80 COLUMN CARD FOR IIe — **$109.00**
- MEMORY MASTER IIE 64K + RAM & 80 COLUMN CARD — **$145.00**
- MEMORY MASTER IIE 128K RAM & 80 COLUMN CARD — **$175.00** *
- MICROTEK II + 128 K VISICALC AND MEMORY EXPANSION — **$219.00**
- MODEM ELIMINATOR CABLE — **$21.00**
- SERI-ALL SERIAL INTERFACE CARD — **$119.00**
- 80-COLUMN CARD (VIEWMASTER) WITH SOFT-SWITCH — **$129.00**
- CENTAURI APS Z-80 CARD — **$59.00**
- Z-80 PLUS CARD (CPM FOR APPLE) — **$115.00**
- FAST Z-80 CARD APPLICARD — **$175.00**
- TIMEMASTER II CLOCK/CALENDAR CARD — **$109.00**
- QUICK-LOADER PROM BOARD — **$149.00**
- ANALOG/DIGITAL BOARD — **$99.00**
- SUPER I/O BOARD — **$49.00**
- MULTIPLE-SLOT EXPANSION CHASSIS — **$149.00**
- SINGLE-SLOT EXTENDER — **$29.00**

SPECIAL PERIPHERALS

- COOLING FAN WITH SURGE PROTECTOR — **$39.00**
- TITAN KEYBOARD — **$159.00**
- LIFETIME EXTERNAL POWER SUPPLY — **$179.00**
- SHIFT KEY MOD KIT — **$8.00**
- SCREEN SWITCHER/ DRIVE STEPPER — **$74.00**

APPLE SOFTWARE

- WORD STAR — **$195.00**
- MAIL MERGE/SPELL STAR/ STAR INDEX — **$125.00**

LONG DISTANCE CALL TOLL FREE WITH TOUCH TONE PHONE FROM ANY CITY! DIAL 950-1088 WAIT FOR TONE DIAL 363-1313

ORDERS & CALL BACK MESSAGES (ANSWERING MACHINE) (202) 362-9176

UPS shipping, $4.00 per order plus $6.00 per printer or monitor

VISIT OUR NEW STORE LOCATION 8231 WOODMONT AVE. (AT BATTERY LANE) IN BETHESDA

STORE HOURS:

12-8	M-TH
12-6	FRI
11-5	SAT

SPECIAL. APPLE IIc PRINTER CABLES $19.00*

(202) 363-1313

VF ASSOCIATES

8231 WOODMONT AVE., BETHESDA, MARYLAND