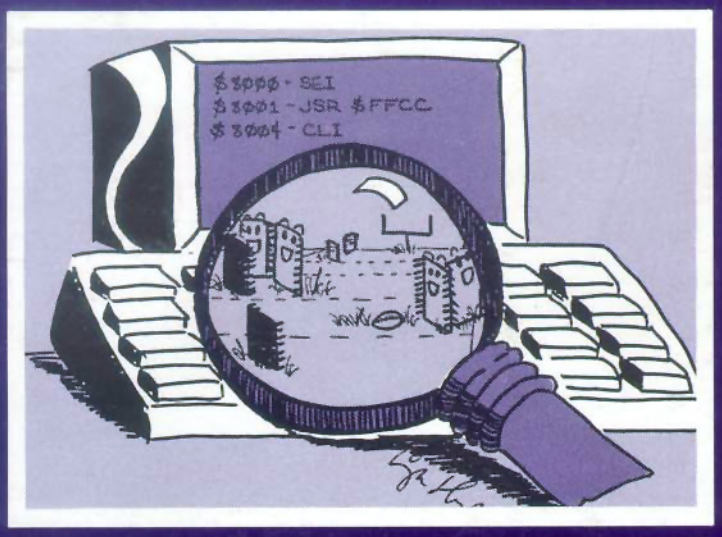
HOW TO BACKUP, UNLOCK, OR MODIFY COPY-PROTECTED SOFTWARE

Hardcore

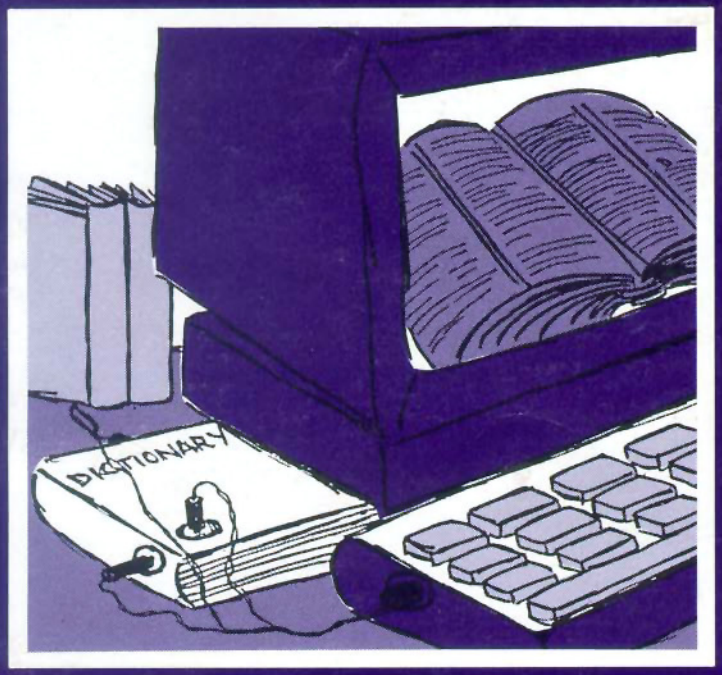
COMPUTIST

Issue No. 9 $2.50



Softkey For The Visible Computer: 6502

Unlock this 6502 Simulator



Super IOB

Hardcore COMPUTIST's own versatile deprotection program

Sensible Speller Softkey

The long awaited backup procedure for this useful spelling checker

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

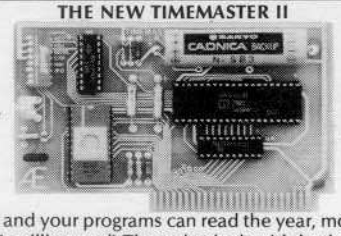
BULK RATE
U.S. Postage
PAID
Tacoma, WA
Permit No. 269

APPLIED ENGINEERING IS 100% APPLE

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

THE NEW TIMEMASTER II

Automatically date stamps files with PRO-DOS



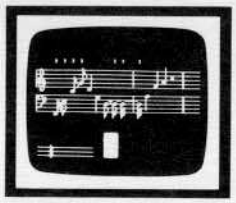
NEW 1984 DESIGN
An official PRO-DOS Clock

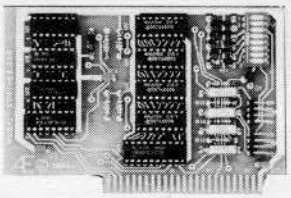
- Just plug it in and your programs can read the year, month, date, day, and time to 1 millisecond! The only clock with both year and ms.
- A rechargeable NiCad battery will keep the TIMEMASTER II running for over ten years.
- Powerful 2K ROM driver — No clock could be easier to use.
- Full emulation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

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

PRICE $129.00

Super Music Synthesizer
Improved Hardware and Software

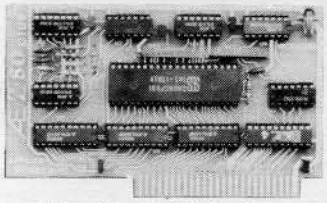




- Complete 16 voice music synthesizer on one card. Just plug it into your Apple, connect the audio cable (supplied) to your stereo, boot the disk supplied and you are ready to input and play songs.
- It's easy to program music with our compose software. You will start right away at inputting your favorite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

PRICE $159.00

Z-80 PLUS!



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

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

PRICE $139.00

Viewmaster 80

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

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

	PRICE	BUILT IN SOFTSWITCH	SHIFT KEY SUPPORT	LOW POWER DESIGN	80 COLUMN HOME	7x9 DOT MATRIX	LIGHT PEN INPUTS	40 COLUMN OVERRIDE	INVERSE CHARACTERS
VIEWMASTER	169	YES	YES	YES	YES	YES	YES	YES	YES
SUP'RTERM	MORE	NO	YES	NO	NO	NO	NO	YES	YES
WIZARD80	MORE	NO	NO	NO	NO	YES	NO	YES	YES
VISION80	MORE	YES	YES	NO	NO	YES	NO	NO	NO
OMNIVISION	MORE	NO	YES	NO	NO	NO	NO	YES	YES
VIEWMAX80	MORE	YES	YES	NO	NO	YES	NO	NO	YES
SMARTERM	MORE	YES	YES	NO	NO	NO	YES	YES	NO
VIDEOTERM	MORE	NO	NO	YES	NO	YES	YES	NO	YES

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

PRICE $179.00

MemoryMaster IIe 128K RAM Card

- Expands your Apple IIe to 192K memory.
- Provides an 80 column text display.
- Compatible with all Apple IIe 80 column and extended 80 column card software (same physical size as Apple's 64K card).
- Can be used as a solid state disk drive to make your programs run up to 20 times FASTER (the 64K configuration will act as half a drive).
- Permits your IIe to use the new double high resolution graphics.
- Automatically expands Visicalc to 95 K storage in 80 columns! The 64K config. is all that's needed, 128K can take you even higher.
- PRO-DOS will use the MemoryMaster IIe as a high speed disk drive.
- Precision software disk emulation for Basic, Pascal and CP/M is available at a very low cost. NOT copy protected.
- Documentation included, we show you how to use all 192K.

If you already have Apple's 64K card, just order the MEMORYMASTER IIe with 64K and use the 64K from your old board to give you a full 128K. (The board is fully socketed so you simply plug in more chips.)

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

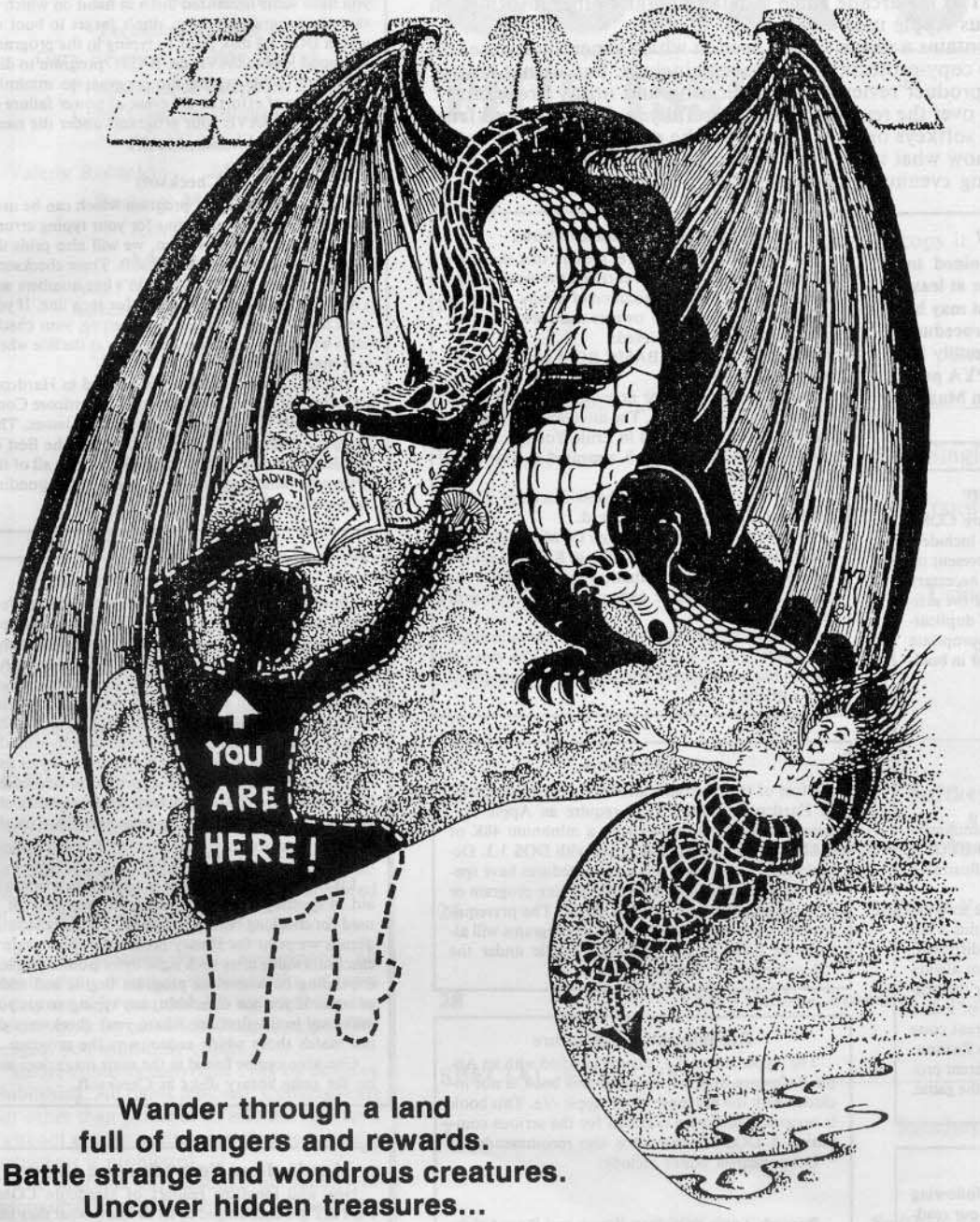
Our boards are far superior to most of the consumer electronics made today. All I.C.'s are in high quality sockets with mil-spec. components used throughout. P.C. boards are glass-epoxy with gold contacts. Made in America to be the best in the world. All products work in the APPLE IIE, II, II+ and Franklin.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

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

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

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

Where the Adventures never end.



Wander through a land full of dangers and rewards. Battle strange and wondrous creatures. Uncover hidden treasures...

In EAMON, you roam through a fantasy world where YOU control the action...and your destiny.

Enclose $4.00 for each EAMON scenario volume that you order or use our special EAMON collectors offer. Adventure scenarios are packed on double-sided disks. Minimum Order: 2 Volumes. ($8.00)

HC9

Name _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone _____

VISA/MC # _____ Exp _____

Signature _____

Make checks payable to:

Computer Learning Center
P.O. Box 45202
Tacoma, WA 98445

US funds only.
No purchase orders or C.O.D.
Washington state residents add 7.8% sales tax.
Foreign orders add 20% for shipping and handling.

The MASTER disk is required to play any EAMON scenario.

- ☐ 01 MASTER/Beginners Cave
- ☐ 02 Lair of the Minotaur
- ☐ 03 Cave of the Mind
- ☐ 04 Zyphur River Venture
- ☐ 05 Castle of Doom
- ☐ 06 Death Star
- ☐ 07 Devil's Tomb
- ☐ 08 Abductor's Quarters
- ☐ 09 Assault of the Clone Master
- ☐ 10 Magic Kingdom
- ☐ 11 Tomb of Molinar
- ☐ 12 Quest for Trezore
- ☐ 13 Caves of Treasure Island
- ☐ 14 Furioso
- ☐ 15 The Heroes' Castle
- ☐ 16 Caves of Mondamen
- ☐ 17 Merlin's Castle
- ☐ 18 Hogarth Castle
- ☐ 19 Death Trap
- ☐ 20 The Black Death
- ☐ 21 Quest for Marron
- ☐ 22 Senators' Chambers
- ☐ 23 Temple of Ngurct
- ☐ 24 Black Mountain
- ☐ 25 Nuclear Nightmare
- ☐ 26 Assault on the Moleman
- ☐ 27 Revenge of the Moleman
- ☐ 28 Tower of London
- ☐ 29 Lost Island of Apple
- ☐ 30 Underground City
- ☐ 31 Gauntlet
- ☐ 32 House of Ill Repute
- ☐ 33 Orb of Polaris
- ☐ 34 Death's Gateway
- ☐ 35 Lair of the Mutants
- ☐ 36 Citadel of Blood
- ☐ 37 Quest for the Holy Grail
- ☐ 38 City in the Clouds
- ☐ 39 Museum of Unnatural History
- ☐ 40 Deamons Playground
- ☐ 41 Caverns of Lanst
- ☐ 42 Alternate Beginners Cave
- ☐ 43 Tomb of Y'Golonac
- ☐ 44 Operation Crab Key
- ☐ 45 Feast of Carroll
- ☐ 46 The Master's Dungeon
- ☐ 47 Crystal Mountain
- ☐ 48 Lost Adventure
- ☐ 49 The Manxome Foe
- ☐ 50 Behind the Sealed Door
- ☐ 51 Land of Death
- ☐ 52 Jungles of Vietnam
- ☐ 53 Black Castle of Nagog
- ☐ 54 Sewers of Chicago
- ☐ 55 Caverns of Doom
- ☐ 56 Valkenburg Castle
- ☐ 57 Modern Problems

Tournament Adventures

- ☐ 60 Castle of Count Fuey
- ☐ 61 Search for the Key
- ☐ 62 The Rescue Mission

EAMON Utilities

- ☐ 01 EAMON Utilities
- ☐ 02 EAMON Utilities
- ☐ 03 EAMON Utilities
- ☐ Dungeon Designer Ver 5

SPECIAL COLLECTORS'S OFFER

Every EAMON scenario & Utilities.
All for only **$200.00**

☐ YES! Send me the entire EAMON collection of 64 volumes.

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

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

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

What Is a Softkey Anyway?

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

Following A Softkey Procedure

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

1) Boot the disk in slot 6

PR#6

or 2) Enter the monitor

CALL -151

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

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

Software Recommendations

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

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

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

Hardware Recommendations

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

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

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

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

Requirements

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

Recommended Literature

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

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

Typing in BASIC Programs

Before you begin typing in any of the Applesoft programs printed in Hardcore COMPUTIST, be sure you have some initialized disks at hand on which to save the program(s). Also, don't forget to boot up with a DOS 3.3 disk prior to typing in the program. It is good idea to SAVE the BASIC program to disk frequently while typing the program to minimize duplication of effort in the event of power failure or nuclear war. SAVE your programs under the name that is indicated in the article.

Checksoft

Checksoft is a Binary program which can be used to check Applesoft programs for your typing errors. For each Applesoft program, we will also print the checksums generated by Checksoft. These checksums consist of the Applesoft program's line numbers and a hexadecimal (base 16) number for each line. If you use Checksoft and make a typing error, your checksums will differ from ours beginning at the line where you made the error.

The Checksoft program was printed in Hardcore COMPUTIST no.1 and The Best of Hardcore Computing, both which are available as back issues. This program is also on Library Disk #1 and the Best of Hardcore Library disk. These disks contain all of the programs which appeared in their corresponding magazines.

Typing in Binary Programs

Binary programs are printed in two different formats, as source code and as object code in a hexadecimal dump. If you want to type in the source code, you will need an assembler. The S-C Macro Assembler is used to generate all the source code which we print, although any assembler whose use you understand will do just fine for entering source code. Binary programs can also be entered directly with the use of the Apple monitor by typing in the bytes listed in the hexdump at the appropriate addresses. Don't forget to BSAVE binary programs with the proper address and length parameters listed in the article.

Checkbin

Like Checksoft, Checkbin is a program which will aid in spotting the typing errors in programs. It is used for checking binary programs. The hexadecimal dumps we print for Binary programs will contain a checksum value after each eight bytes (sometimes less depending on where the program begins and ends) of code. If you use Checkbin, any typing errors you make are in the first line where your checksums do not match those which accompany the program.

Checkbin can be found in the same magazines and on the same library disks as Checksoft.

Let Us Hear Your Likes and Gripes

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

How-To's Of Hardcore

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

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

Publisher/Editor
Charles R. Haight
Technical Editors
Gary Peterson Ray Darrah
Production & Graphics
Lynn Campos-Johnson
Circulation
Valerie Robinson Michelle Frank
Business Manager
Ken Fields
Advertising
Attn: Valerie Robinson
Advertising Department
3710 100th Street SW
Tacoma, WA 98499
Printing
Grange Printing, Inc.
Seattle, WA
Publishing
Softkey Publishing
P.O. Box 44549
Tacoma, WA 98444
USA

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

MAILING NOTICE: Change of address must be postmarked at least 30 days prior to move. Paste your present mailing label on postal form 3576 and supply your new address for our records. Issues missed due to non-receipt of change of address may be acquired at the regular back issue rate.

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

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

Hardcore COMPUTIST

THIS ISSUE:

INPUT INPUT INPUT

A Reader's Recommendations

I would like to recommend two books about machine language.

"Assembly Lines: The Book" by Roger Wagner is an excellent book for the beginner, who should be equipped with either the DOS Toolkit Assembler from Apple, or Merlin by Roger Wagner Publishing, formerly Southwestern Data Systems. Merlin is also offered through CALL A.P.P.L.E.to their club members as Big Mac at a substantially reduced price.

"Apple Machine Language" by Don and Kurt Inman is a great book for anyone who already understands BASIC. It is extremely simple to follow and does not require any specific assembler. I would not recommend using the book "Using 6502 Assembly Language" for a beginner. Randy Hyde assumes too much and is vague in some important areas.

My personal preference in assemblers is Lisa 2.5 which is extremely quick but has a truly lousy editor. Unfortunately, I do not have Lisa 2.6, the new version by Lazerware. Merlin/Big Mac and the S-C Assembler are also very good.

Although I do not have Ultima III or Flight Simulator II yet, I have heard that they can be copied with the Essential Data Duplicator. For Ultima III, copy tracks 0-11 normally. If you get an error on track A, copy it over with auto lengths.. For Flight Simulator II, use EDD Version 3 and copy tracks 0 through 22 auto lengths and sync.

Re. Hardcore COMPUTIST no.7: Although interviewing Mr. Xerox or Krac-Man would be a good idea, it would prove difficult. Mr. Xerox has turned traitor and become a copy protector while Mr. Krac-Man is usually extremely busy.

Pirate's Friend, the copier everyone seems to be talking about, is in fact a modified (or rather renamed) copy of the early Copy II Plus. The newer version is much better.

Lastly, I have compiled a double-sided disk of public domain cracking utilities and copiers. These programs are all public domain and therefore, if you have them, you can distribute them among your friends.

On my disk are programs like DeMuffin Plus, Advanced DeMuffin, Fastload Create, Disk Muncher 1.1, CopyB, CopyC, Crack 'Em, Unlock It Up, Craft Copy, Mini-RWTS, A "Crack ROM", and others. Each includes a text-file describing basic use. I have cracked many protected programs with these.

If you would like a copy of this disk, send a blank disk (or two, if you prefer that I don't put anything on the back) with $4. for postage and copying. Don't forget to package carefully. The post office has a tendency to damage unprotected disks.

Marco Hunter
2606 Roscomare Rd
Los Angeles, CA 90077

Tricking The Accountant

I noted in your Volume 3, Number 3 edition of Hardcore COMPUTIST that you are requesting assistance with softkeys to a number of programs, in particular, the Accountant from Decision Support Software. As far as I know, this program is not protected except for a sixteen pin connector that you plug into the game port.

Several years ago I accidentally severed several of the pins for my connector. Instead of taking the time to contact the supplier, I explored the connector with an ohmmeter and discovered that there was a 10,000 ohm resistance between pins 11 and 13.

I soldered a 10,000 ohm resistor to a new connector and was back in business with very little down time. The resistors and connectors are available from most electronic stores such as Radio Shack.

Ted M. Doniguian
Laguna Beach CA

A Chess Check

Re. Modified ROMS article by Ernie Young

Many thanks for your article. I made an F8 EPROM per your instructions and everything worked as you indicated.

However, my Sargon III chess game will no longer boot. I get a flashing screen with garbage and the drive won't stop. My intent was to make a backup copy (preferably an unprotected copy I could modify). None of my copy programs will copy it. Any ideas? My system is a 48K Apple II+, 2DD, 16K RAM, INT Card in slot 4, MX-80 and modem.

I would appreciate any comments you might have on this.

Phil Boling
Dallas TX

Mr. Boling: *Probably the reason your Sargon III will no longer boot when the modified F8 ROM is installed on the motherboard is that the program is performing a "checksum" of ROM memory. If the checksum does not match a specified value, the program knows nonstandard ROMs are present and it will not run.*

To get around this you might try reinstalling the autostart F8 ROM on the motherboard and installing the modified ROM in your Integer card. Then boot Sargon III with the Integer switch down. After the program has started, flip the Integer card switch up and then hit RESET. Hopefully you will end up in the monitor when the ":" or "-" key is hit. If this does not work, try removing your 16K RAM card from slot 0.

S-C Clarifies

It was a great pleasure to see Jeff Thomas' review of the S-C Macro Assembler in issue no. 7 of Hardcore COMPUTIST. We're always happy to hear from a satisfied programmer who appreciates our work.

I would like to correct one error in his article, and to explain why we use type "I" source files. First, the source code for the assembler is not provided as part of the package; it is available at extra cost. Registered owners of S-C Macro Assemblers Version 1.1 may purchase the completely commented source code on disk for $100. I know of no other professional quality assembler on the Apple that supplies source code for the assembler at any price.

About the type "I" files: This feature allows the assembler to masquerade as Integer BASIC, providing complete DOS transparency and permitting you to switch between the assembler and Applesoft with the INT and FP commands. It also allows DOS to handle the source files with fast LOAD and SAVE commands rather than by the much slower Text file routines. As Jeff mentioned, it is easy to assign distinctive filenames to your assembler source files. We usually use a prefix of "S." on our source code files, and "B." on the Binary object files.

That's all for now. Thanks for both of you fine magazines, and keep up the good work.

Bill Morgan
S-C Software Corporation
Dallas TX

Bill: *The error in the review was not Jeff's, but was due to an editorial insertion of ours. What we meant to say was that the code was available, not provided.*

More On RESETs

I found the article, "Whiz Kid", very interesting. I also liked the protection scheme mentioned. But there is one protection scheme I have not seen in any magazines. This protection scheme is to disable the Reset button of the Apple II, II+ and //e. I am a new subscriber to your magazine, so I'm not sure if you have published this protection scheme I'm about to mention.

In most programs on the market, if you hit the reset button it will either boot the disk or remain the same. If you hit the reset in Copy][Plus 4.3, the main menu will appear. If you hit it again, it will stay the same. I have done this in Applesoft, and have a list of both types of reset disable schemes. They are as follows:

```
POKE 1010,213:POKE 1011,203:
CALL -1169
```

The second disable scheme occured when I was entering a huge program. I placed the reset disable scheme on line 0, but I typed CALL -1168 by mistake. When I hit reset it booted the disk. This second reset scheme is as follows:

```
POKE 1010,213:POKE 1011,203:
CALL -1168
```

I hope that you place these protection schemes in your next issue of Hardcore.

P.S. I have one question. How do you get the red sphere in Zork II The Wizard of Frobozz?

Jeff Lucia
W. Caldwell NJ

Mod² ROMS

I thought Ernie Young's article on modified ROMs was one of your best ever. At the time I read it, I had coincidentally just obtained an EPROM Programmer, so I immediately burned a Super Saver EPROM and inserted it in my Franklin Ace 1000 (which, by the way, uses EPROMs for its ROM memory, eliminating the need for a conversion socket). After using it awhile I decided to make a couple of minor changes.

Upon powering up, I always want to boot a disk, so I might as well have the monitor do this automatically without having to hit the return key. This can be accomplished by a little vector swapping. Instead of passing control from the reset vector directly to the Super Saver routine and then to the routine at $FA62 if return is hit, we can go first to $FA62. If the computer decides this is a powerup, it will then boot the disk. If not, we can then substitute a jump to the Super Saver routine instead of the vector at $3F2. Finally, we complete the path by placing an indirect jump to the address at $3F2 in the Super Saver routine.

In the original Super Saver, the NMI and reset vectors were changed as follows:

$FFFA: CD FE CD FE

To swap vectors, the following three additional changes can be made:

1) $FFFC: 62 FA
2) $FAA4: FA FF
3) $FEE1: F2 03

The first change actually just restores the reset vector to its normal value in the autostart monitor. The second indirectly uses the new NMI vector to pass control from the normal autostart reset to the Super Saver. The third readjusts the destination of the Super Saver reset when the return key is hit. Thus, the new routine will be bypassed when the computer is first turned on, but will be entered if the reset key is pressed.

The second change involves the key that must be hit after the Super Saver subroutine has been entered. You'll recall that hitting reset causes the computer to freeze; what the computer does next depends on whether the minus sign, the colon, or the return key is hit. the pertinent section of the routine reads:

```
LDA KEYBOARD
CMP #'-
BEQ MONITOR
BCS NEW STUFF
JMP RESET
```

If the key pressed equals the minus sign, the monitor is entered. However, the BCS NEW STUFF means we will branch to NEW STUFF not only if the colon key is hit, but if any key with an ASCII value greater than 2D (the minus sign) is hit. This includes all the normal characters on the keyboard, greatly increasing the possibility of accidentally clobbering a program when using reset for its normal purpose. A more specific test would be:

```
LDA KEYBOARD
CMP#'-
BEQ MONITOR
CMP#':
BEQ NEW STUFF
JMP RESET
```

This routine is slightly longer than the original, but there is still room for it. We do, however, have to adjust our jump over the CRMON routine ($FEF6-FEFC, which must remain intact). Once this is done, the modified hexdump is as follows:

```
$FECD: 2C 00 C0
$FED0: 10 FB 8D 10 C0 AD 00 C0
$FED8: C9 2D F0 7D C9 3A F0 03
$FEE0: 6C F2 03 BA 8E 01 29 A0
$FEE8: 00 B9 00 00 99 00 20 B9
$FEF0: 00 01 4C FD FE EA 20 00
$FEF8: FE 68 68 D0 6C 99 00 21
$FF00: C8 D0 E6 84 3C 84 42 84
$FF08: 3E A9 09 85 3F A9 02 85
$FF10: 3D A9 22 85 43 20 2C FE
$FF18: 20 2F FB 20 58 FC 4C 59
$FF20: FF 60
```

Again, thanks to Mr. Young and Hardcore for a great article.

Anybody out there know the parameters for Flight Simulator II? Also, I'd like to see an article, should someone be inspired to write it, on the use of ROM expansion memory $C800-CFFF, with an eye towards its use in softkeys.

Paul Mindrup
Los Angeles CA

Super-Text On The //e

(Note:A 1" piece of 18 gauge solid copper wire with the insulation removed from both ends was attached to the following letter.)

Attached you will find a shift key modification kit for the Apple //e. I own both an Apple II+ and a //e. Using the Super-Text 40-80 column word processing program and desiring to use it with the //e at my office, I found that the shift key would not work. The fix, after calling the Muse Co., was to order their new "Professional" program because of various differences in the machines.

While waiting for the new unit, which I think was well worthwhile, I read about Visidex using a different paddle button to check for the shift key. So, I inserted the jumper wire between pins #2 and #4 in the game socket of the Apple //e and PRESTO!, it would now shift with the 40-80 program.

While the newer program does make better use of the //e, I have yet to find a problem with using the older one this way. I wonder how many other II+ programs, sitting on the shelf unused, might be put back to work with this modification by someone who now owns a //e?

Gerald P. Gibson
Dallas TX

Mommy, Will Kool-Aid Hurt My Disk?

Now that I'm a Hardcore COMPUTIST subscriber, I need some help from you. Would you be so kind as to help me out? My problem is in trying to make a backup of my young daughter's programs. They include: Story Machine, Face Maker, and Type Attack.

Maybe I missed how to do these in your past issues of Hardcore COMPUTIST. If so, please help me out.

Thank you for the information that I couldn't find anywhere else.

L.P. Williams
West Warwick RI

L. P.: Sorry, we don't have any sure-fire method of backing up any of the programs you mention. We have added them to our "Most Wanted List" in order to solicit information from our readers

A Few Suggestions

I enjoy your magazine very much. I especially appreciate your concern (HC#6, p.4) for us beginners. I especially like the items presented in the INPUT column in reply to readers' letters. However, a few items were omitted that would have helped us novices. For example, on Page 5 Clay Harrell explained why he made the change to DOS at $B942 (and I really appreciate this and his other hints), but Dan Lui on Page 6 made some similar changes with no explanation. Very confusing. And this reduces the value of the information. We see how to use the softkey for Donkey Kong (which I don't happen to have), but without an explanation of why he made the changes, we don't see how to apply the technique. I feel the editor should have seen to it that Lui's response be expanded or added to using your own explanations.

I hope you keep publishing Advanced Playing Techniques. I think the idea is great; occassionally I have one of the games mentioned.

Which brings me to an important idea for you. As you continue to publish softkeys and APT's, it is difficult for us to keep track of what software is covered in which issue of Hardcore or CORE. How about publishing an index? Maybe in the last issue every year like Consumer Reports and others.

I enjoyed the Modified ROMs article. But it would have helped readers a lot if Mr. Young had referenced some other articles that have been published recently on the subject ,as he left out some critical information: he never told us how to transfer the memory of the computer to the EPROM!! If you, or he had referenced the article in the November 1983 issue of Computers and Electronics, the readers might have had a hint. Byte magazine has also published on burning EPROMs.

Another fact that would help a lot is a source of inexpensive EPROM burners. I'm sure there are several suppliers of Taiwan EPROM burners around who will supply them for around $60. Otherwise, the burners are so expensive and out of reach of most of your readers, that the article is of little value.

Keep on publishing.

Ken Burnell
Adh-Dhahran
Kingdom of Saudi Arabia

Some Common Questions

I received my first issue of your magazine and was very pleased with the content. Starting a new subscription is sometimes like starting to read a new book on page 100 and wondering what you have missed. Reading your magazine makes me wonder what I have missed in the previous issues. This leads me to the following questions.

1. Are back issues available?
2. If so, do you identify the contents of the back issues?
3. If available, what is the cost and how do I order?
4. Is the CHECKSOFT program in a back issue?
5. Is information available on DEMUFFIN PLUS in back issues?

In Issue #7, the lead paragraph of the MOST WANTED LIST indicated that you have received numerous softkeys for "Sensible Speller". This program frustrates me to no end. I have boot traced it from $800 through $200 and $300 through $400. Because of the self-modifying code in $300, I have only been able to list $400, but cannot "RUN" it without losing control of the program. Any help would be appreciated. Locksmith 5.0 (revision F), still will not copy it.

A note on the MOST WANTED LIST #10, "The Accountant" by Decision Support Software, purchased in November 1983 (version 5), has no protection and comes with an errata sheet telling the user to ignore the "Game Port" hardware installation. My version can be copied with COPYA.

A note on the "Tax Advantage" 1983 by Continental Software. The only protection used is a changed routine as follows: CALL -151, then type B993:01. This branches around the I/O ERROR and allows the read. I then use FID to transfer the files to a previously INIT-ed disk with HELLO deleted. After all of the files have been transferred to the new disk, I wrote the following HELLO program, to boot the disk.

```
NEW
10 REM BOOT TAX ADVANTAGE
20 D$ = CHR$(4)
30 PRINT D$; "RUN TTA.PROG."
40 END
SAVE HELLO
```

I modified the print program to read my clock card so that my printouts were dated and therefore, not confusing when I was doing what-ifs.

Keep up the good work and let me know how to obtain the previously published info on how to modify "PROTECTED" software.

Robert. C. Taylor
Annapolis MD

Mr. Taylor: You can find information on back issues of Hardcore COMPUTIST on the inside back cover of this issue. We have not yet published a cumulative index of our magazines so unfortunately there is no easy way to tell what information is contained in a specific issue.

The Checksoft and Checkbin programs were first published in Hardcore COMPUTIST no.1 and are also reprinted in "The Best of Hardcore." Instructions on creating DeMuffin Plus were reprinted in Hardcore COMPUTIST no. 8 on page 15.

Last, but not least, see page 9 of this issue for Cris Rys' article on how to back up Sensible Speller 4.0.

Screenwriter II, V. 2.2

Concerning the backup procedure for Screenwriter II, Version 2.2: IT WORKS! (Ref. HC#7, pg. 4)

Dr. James N. Snaden
Newington CT

DEAR AUTHOR:

Hardcore COMPUTIST welcomes articles on a variety of subjects and would like to publish well-written material on the following:

- **Softkeys**
- **Utilities**
- **Hardware Modifications**
- **Advanced Playing Techniques**
- **DOS modifications**
- **Product reviews**
- **Adventure Tips**
- **Original programs of interest**
- **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. Also enclose a double-spaced hardcopy manuscript. Submissions will be mailed back if adequate return packaging is included.

Hardcore COMPUTIST pays on acceptance. Rate of payment depends on the amount of editing necessary and the length of the article (between $10 for a short softkey and $50 per typeset page for a full-length article.) For softkeys, please enclose the original commercial disk for verification). We guarantee the disk's return.

Softkey Publishing buys all rights and one-time reprint rights on general articles, and exclusive rights on programs. We may make alternate arrangements 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, P. O. BOX 44549, TACOMA WA 98444.

READERS' SOFTKEY & COPY EXCHANGE

Backing-UP VISIDEX

By Anthony L. Barnett

Visicorp
2895 Zanker Road
San Jose, CA 95134
$250.00

Requirements:
48K Apple
One disk drive with DOS 3.3
Visidex
One blank disk
COPYA

Visidex is a key-word index program which has some data storage and retrieval procedures which I have not seen in any other program for the Apple. Its retrieval of data by keyword is very fast.

In Hardcore COMPUTIST No. 3 (page 6) in my softkey for Visitrend/Visiplot, I explained the problems in obtaining backups from Visicorp when overseas. I notice in Hardcore COMPUTIST No. 5 that Bob Bragner describes similar problems in his softkey for Visifile. The same problems apply to Visidex.

Though I have been able to copy Visidex using the long list of Locksmith parameters described in Hardcore Vol. 1 No. 3, I decided that an unprotected backup would be more convenient. For one thing, the disk could also be used to store other files instead of being wasted on one 17K file.

During the course of my examination of the Visidex disk, I discovered that unless the DOS from the original disk is present, Visidex data disks may become corrupted. I also noticed that apart from the dummy serial number, there appeared to be only two very short binary files, namely VISIDEX and VISIO.8. It turns out that VISIDEX is a loader file which uses an RWTS routine to load the whole program into memory.

After some experimentation, I found the following procedure to work.

1) Use COPYA to make a copy of Visidex.
2) **UNLOCK** and **DELETE** the file named VISIDEX on the copy.
3) Boot the copy and wait for it to produce the FILE NOT FOUND message.
4) Insert the original and

BLOAD VISIDEX

5) With the original still in drive 1, enter the monitor with

CALL -151

6) From the monitor type

60A3:69 FF

7) Type

6000G

8) When the drive stops, you should still be in the monitor. Remove the original and insert the copy.
9) Type

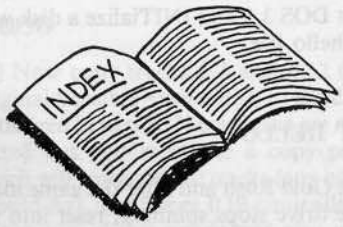
INIT VISIDEX

10) When initialization is complete, type

DELETE VISIDEX

11) Type

BSAVE VISIDEX,A$803,L$4404



The disk so produced, if booted, will run Visidex.

Steps 1 to 3 enable you to get Visidex's DOS into memory.

Step 4 gets the program loader into memory.

Steps 5 to 7 cause a jump to the monitor before the Visidex program is run. The technique here is the same as used with boot code tracing. At $60A2 is the jump out from the loader to the Visidex program at $803. Step 6 alters this to a jump to $FF69 which is the monitor entry address.

During the course of the load, the file VISIO.8 is loaded at $4C00. The essence of this file is 6 NOP's and one RTS. Thus, the effective program begins at $803 and ends at $4C06.

Step 9 is necessary because a DISK FULL message is received if you attempt to BSAVE the program at this point. Be careful not to INIT the original. It is a good idea to keep a write protect tab on original program disks.

Step 10 prevents a FILE TYPE MISMATCH error when BSAVEing the program.

The only difference between this version and the original that I can detect is the response to a RESET. In the original, this usually took you to a "BOOT FROM SLOT 6" message. In the copy, RESET dumps you back to Applesoft.

For those of you who want to make patches to the program, be warned. The program makes extensive use of "funny jumps", i.e. pushing an address onto the stack and then doing an RTS to get there.

Now if anyone knows how to backup DB Master Version 4...

Softkey For MUSIC CONSTRUCTION SET

By Jim Waterman

Electronic Arts
2755 Campus Drive
San Mateo, CA 99403
$40.00

Requirements:
Apple II Plus, or compatible
Music Construction Set
Bit-copy program such as Locksmith 4.1 or Copy II+ ver 4.1
One blank disk

1) Use your bit-copier to copy tracks $Ø-$22 from the Music Construction Set onto a blank disk.

2) Boot a normal 3.3 disk.

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

LOAD A4

4) Enter the monitor and change the bytes at $913A-$913B to NOP's

CALL -151
913A:EA EA

5) Save A4 with the changes you have made

BSAVE A4,A$4AØØ,L4B6Ø

6) The Music Construction Set may now be copied with any normal copy program such as COPYA.

Deprotecting GOLD RUSH

By Clay Harrell

Sentient Software
P.O. Box 4929
Aspen, CO 81612
$34.95

Requirements:
48K Apple II with at least one DOS 3.3 disk drive
Old style F8 monitor ROM or Super Saver ROM

Blank disk
GOLD RUSH

It seems that there are many "maze" games flooding the Apple market, now. Most have pretty much the same theme of eating all the dots before you get destroyed. I do enjoy these games, but variety is the spice of life!

Gold Rush uses the "avoid the bad guys" theme without the maze. It is entertaining, fun, and different from other chase 'em games. Being intrigued by the game, I wanted to learn more about it...

Gold Rush boots fairly quickly and is a "single load" game, which means it does not require any other data from the disk after the initial load. These types of programs are candidates for BLOADable files. When booting the disk, an Applesoft prompt appears at the bottom left of the screen which means there is some sort of modified DOS present. Disks like these are excellant candidates for deprotection with Super IOB.

The"Swap" controller is probably the most versatile since one doesn't usually have to figure out the protection used when using it. With this in mind, we merely boot Gold Rush, reset into the monitor and move RWTS from $B800-BFFF down to $1900 where the standard Super IOB swap operates. This is done from monitor with the command

1900<B800.BFFFM

Now boot a 48K slave disk and run Super IOB.SWAP. Remember that booting a slave disk only destroys memory from $00-8FF and $9600-BFFF, so this keeps our Gold Rush RWTS safe at $1900-20FF.

After making the copy, our new disk is quite CATALOGable. Also, the disk is now deprotected and completely copyable with COPYA and runs fine (providing the boot file name is "BOOT" instead of the normal "HELLO"). But remember, this is a single load game, and as I mentioned before, a good candidate for a BLOADable file.

Typing

CATALOG

reveals one file named "BOOT". You can BRUN this file and the game will load and execute just fine. But we want this game in a file that we can FID to another disk or BLOAD. To start the analysis, type

BLOAD BOOT

Now enter the monitor with

CALL -151

and type

AA60.AA73

The first two bytes listed are the length of the last BLOADed file, and the last two bytes listed are the address the last BLOADed file was loaded at. These are in "bassackward" order, with the low byte first and the high byte last (in other words, 00 03 means $0300, or 88 45 means $4588). We can see from this listing that the file "BOOT" loads at $300 and is fairly short (less than $100 bytes long).

If you examine the code at $300, you will find that Sentient is using second stage DOS to load in the game from $800 to $95FF, and then jumps to $B00 to start the game.

Sentient Software RWTS is not as friendly as the normal RWTS. They load $1B with the track number, $1C with the sector number and $1F with the top page to load to. Then they jump subroutine to $B7B5 to do the load. Look at the code and try to understand it (If you cannot, well, forget it).

After some groaning and grunting, we now know that Gold Rush loads from $800 to $95FF and starts at $B00. With this necessary information in hand, here are the exact steps for deprotecting Gold Rush:

1) Boot DOS 3.3 and INITialize a disk with a null hello file.

FP

INIT HELLO

2) Boot Gold Rush and after the game loads and the drive stops spinning, reset into the monitor

3) Move page eight out of the way so we can boot

6400<800.8FFM

4) Boot the disk you just initialized

C600G

5) Enter the monitor

CALL -151

6) Move $800 to its original location

800<6400.64FFM

7) Alter the program so that it JuMPs to the entry point

7FD:4C 00 0B

8) Tell DOS that we can BSAVE more than 121 sectors

A964:FF

9) Save Gold Rush

BSAVE GOLD RUSH,A$7FD,L$8E04

Gold Rush is now deprotected into a BRUNable file that you can FID to any disk.

Short Softkey For VISITERM

By B. Baker

Visicorp
2895 Zanker Road
San Jose, CA 95134
$100.00

Requirements:
Apple II family with 48K
One blank disk
A sector editing program

1) Copy the disk, using COPYA
2) Use the sector editor to read track $15, sector $0E of the backup disk.
3) Change byte $DF from $B0 to $90 and write the sector back to the disk.

Deprotecting COSMIC COMBAT

By Clay Harrell

Cosmic Combat
Highland Computer Services
14422 SE 132nd
Renton, WA 98056

Requirements:
48K Apple with at least
one DOS 3.3 disk drive
Old style F8 monitor ROM or
Super Saver ROM
Blank disk
Cosmic Combat
Super IOB with Swap Controller

Cosmic Combat is a cute shoot-'em-up with three levels of varied graphics. The game uses standard XDRAW graphics that all we wish we could use better than we do now!

In order for us to examine the code that makes Cosmic Combat tick and learn from it, it is necessary for us to deprotect it.

Cosmic Combat uses a modified DOS for its protection. We know this since when we boot the game, the Applesoft prompt appears at the bottom left of the screen, and the boot sounds 'normal' (listening to a program boot can sometimes reveal quite a bit about the protection used!)

The first thing I always try when deprotecting a program that uses a modified DOS is to defeat the DOS error checking routines (from normal DOS) and try CATALOGing the disk. To do this, enter the monitor with CALL -151 and change byte $B942 from 38 to 18. What this does is defeat all the DOS error checking routines that Apple so carefully put in for us. If you do this with Cosmic Combat, you will be disappointed when you type "CATALOG" because all sorts of garbage will appear on the screen.

Continued on page 31

Sensible Speller Softkey

By Cris Rys

Sensible Speller Version 4.Ø
Sensible Software
6619 Perham Drive West
Bloomfield, MI 48Ø33
(313) 399-8877
$125.00

Requirements:
Apple II,II+
16K RAM CARD
Blank Disk Sensible Speller V 4.Ø
Super IOB or any DOS track copy program

Sensible Speller version 4.Ø is a very useful dictionary program which has a vocabulary of over 8Ø,ØØØ words. The softkey of this program incorporates a very useful utility of the language card. This utility is customizing the monitor routines so when you press the RESET key, you arrive in the monitor. Unfortunately, this was the only way I could get a deprotected version of the program. If you don't have a language card (separate from the motherboard) you won't be able to use the following procedure.

1) With the computer off, put your 16K card in slot one.
2) Boot up the DOS 3.3 master disk.
3) Enter the monitor

CALL-151

4) Modify DOS so that the SPELLER.LOADER program will work.

```
B639:  AD 81 C0 AD 81 C0 4C B3 08
B6B3:  A0 00 B9 00 D0
B6B8:  99 00 D0 C8 D0 F7 EE B7
B6C0:  08 EE BA 08 AD BA 08 D0
B6C8:  EC AD 80 C0 A9 07 8D 00
B6D0:  02 4C 00 B7
```

5) Initialize the disk you want Sensible Speller on with this modified DOS

INIT HELLO

6) Insert a different disk and key in the SPELLER.LOADER program on page 1Ø.
7) Save this program in the event of error

BSAVE SPELLER.LOADER,
A$B7ØØ,L$9A

8) Type in this short machine language program which calls the RWTS

8Ø3:A9 B7 AØ E8 4C B5 B7

9) Tell the RWTS that we wish to write track zero, sector one from page $B7

B7EB:ØØ ØØ Ø1
B7FØ:ØØ B7 ØØ ØØ Ø2

1Ø) Insert the disk you initialized in step 5 and write the sector

8Ø3G

11) Now copy tracks 2-3 and 6-22 of your original Sensible Speller disk to the disk you initialized in step 5. These tracks are not protected but you must use a copy program which will copy specific tracks (any bit copier should do). The Super IOB controller listed at the end of this article will also work.

Here is the routine I mentioned at the beginning of this softkey. Sensible Speller, like many other programs, checks for a RAM card in slot zero. Because several programs (such as Sensible Speller) only check slot Ø, this technique may be used with other programs as well.

12) Boot a normal DOS disk and enter the monitor

CALL-151

13) Write enable the language card

CØ91 CØ91

14) Copy the monitor into the language card

F8ØØ<F8ØØ.FFFFM

15) Read and write enable the language card

CØ93 CØ93

16) Type in the SPELLER.SAVER program from page 1Ø and save it on the same disk you saved SPELLER.LOADER on

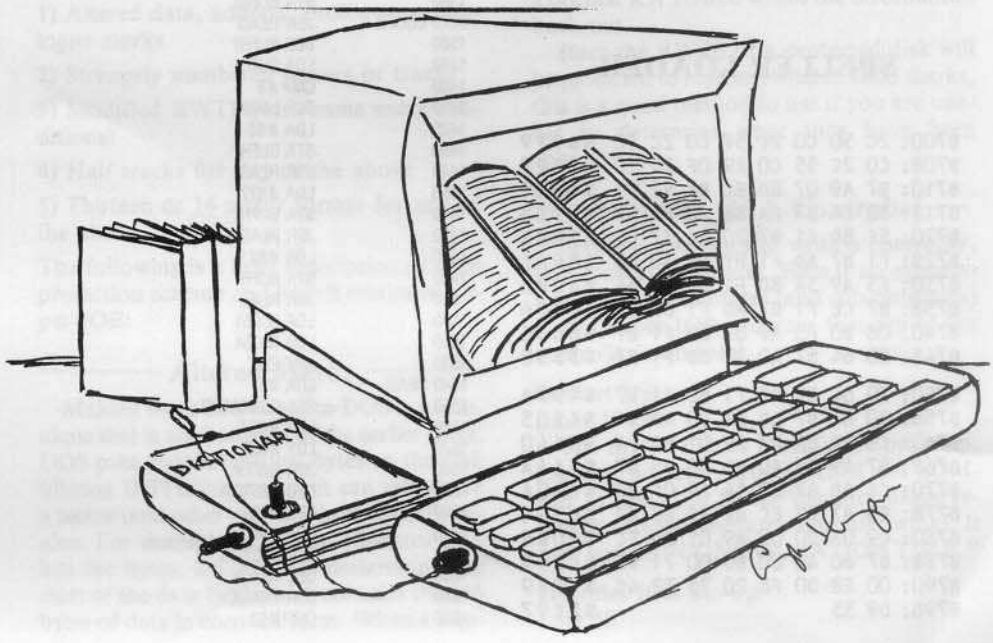
BSAVE SPELLER.SAVER,
A$DØØØ,L$6Ø

17) Type a routine to save page zero into page $71 when RESET is hit

FA62:A2 ØØ B5 ØØ 9D ØØ 71
E8 DØ F8 4C 59 FF

18) Read enable and write protect the language card RAM

CØ9Ø



19) Insert your original Sensible Speller disk and boot it

C600G

20) When the menu appears and the disk drive stops running, press RESET. You should now be in monitor.
21) Read enable the language card in slot 1

C090

22) Insert the disk you want Sensible Speller on and invoke the SPELLER.SAVER program

D000G

The disk drive will start whirling and after a while the cursor will come back. You're done! Now may be a good time to put the language card back in slot zero, but it is not necessary to do this to run this program. Don't forget to turn the computer off first.

SPELLER.CON

```
1000 REM SENSIBLE SPELLER HELPER

1010 TK = 2:ST = 0:LT = 35:CD = W
     R
1020 T1 = TK: GOSUB 490
1030 GOSUB 430: GOSUB 100:ST = S
     T + 1: IF ST < DOS THEN 1030

1040 IF BF THEN 1060
1050 ST = 0:TK = TK + 1 + 2 * (TK
     = 3): IF TK < LT THEN 1030
1060 GOSUB 490:TK = T1:ST = 0
1070 GOSUB 430: GOSUB 100:ST = S
     T + 1: IF ST < DOS THEN 1070

1080 ST = 0:TK = TK + 1 + 2 * (TK
     = 3): IF BF = 0 AND TK < LT
     THEN 1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT : PRINT "DONE
     WITH COPY": END
```

SPELLER.LOADER

```
B700: 2C 50 C0 2C 57 C0 2C 52  $D9F9
B708: C0 2C 55 C0 A9 0F 8D ED  $C1F9
B710: B7 A9 07 8D EC B7 A2 01  $642C
B718: 8E EA B7 CA 8E F0 B7 A9  $4728
B720: 5F 8D F1 B7 20 64 B7 CE  $1BBD
B728: F1 B7 AD F1 B7 C9 40 B0  $89AF
B730: F3 A9 3A 8D F1 B7 20 64  $8CB6
B738: B7 CE F1 B7 AD F1 B7 C9  $21C6
B740: 08 B0 F3 A9 03 8D F1 B7  $860E
B748: 20 64 B7 A9 77 8D F1 B7  $3558

B750: 20 64 B7 A9 71 8D F1 B7  $4976
B758: 20 64 B7 AD 51 C0 AD 54  $A103
B760: C0 4C 8A B7 A9 01 8D F4  $6640
B768: B7 A9 B7 A0 E8 20 B5 B7  $366A
B770: CE ED B7 10 14 A9 0F 8D  $FA0A
B778: ED B7 CE EC B7 AD EC B7  $C5F4
B780: C9 03 D0 05 A9 01 8D EC  $E0B8
B788: B7 60 A2 00 BD 00 71 95  $CB95
B790: 00 E8 D0 F8 20 75 32 4C  $1FE9
B798: D9 33                    $DC97
```

SPELLER.SAVER

```
D000: A9 0F 8D ED B7 A9 05 8D  $5713
D008: EC B7 A2 01 8E EA B7 CA  $48AB
D010: 8E F0 B7 A9 3A 8D F1 B7  $2A32
D018: 20 3A D0 CE F1 B7 AD F1  $6C61
D020: B7 C9 08 B0 F3 A9 03 8D  $0858
D028: F1 B7 20 3A D0 A9 77 8D  $1E6E
D030: F1 B7 20 3A D0 A9 71 8D  $E7BE
D038: F1 B7 A9 02 8D F4 B7 A9  $906A
D040: B7 A0 E8 20 B5 B7 CE ED  $7EAB
D048: B7 10 14 A9 0F 8D ED B7  $6F4D

D050: CE EC B7 AD EC B7 C9 03  $9EA3
D058: D0 05 A9 01 8D EC B7 60  $37FA
```

SPELLER.LOADER SOURCE CODE

```
1000 * ------------------------------
1010 * SPELLER.LOADER
1020 * THIS LOADS WHAT USED TO BE
1030 * PART OF SENSIBLE SPELLER
1040 * WHILE IT IS BOOTING
1050 * ------------------------------
1060
1070              .OR $B700
1080              .TF SPELLER.LOADER
1090
1100 DRIVE        .EQ $B7EA
1110 TRACK        .EQ $B7EC
1120 SECTOR       .EQ $B7ED
1130 BUFHI        .EQ $B7F1
1140 COMMAND      .EQ $B7F4
1150 RWTS         .EQ $B7B5
1160              BIT $C050
1170              BIT $C057
1180              BIT $C052
1190              BIT $C055
1200              LDA #$F
1210              STA SECTOR
1220              LDA #$7
1230              STA TRACK
1240              LDX #1
1250              STX DRIVE
1260              DEX
1270              STX BUFHI-1
1280              LDA #$5F
1290              STA BUFHI
1300 LOOP1        JSR READ
1310              DEC BUFHI
1320              LDA BUFHI
1330              CMP #$40
1340              BCS LOOP1
1350              LDA #$3A
1360              STA BUFHI
1370 LOOP2        JSR READ
1380              DEC BUFHI
1390              LDA BUFHI
1400              CMP #8
1410              BCS LOOP2
1420              LDA #$3
1430              STA BUFHI
1440              JSR READ
1450              LDA #$77
1460              STA BUFHI
1470              JSR READ
1480              LDA #$71
1490              STA BUFHI
1500              JSR READ
1510              LDA $C051
1520              LDA $C054
1530              JMP EXIT
1540 READ         LDA #1
1550              STA COMMAND
1560              LDA #$B7
1570              LDY #$E8
1580              JSR RWTS
1590              DEC SECTOR
1600              BPL RTS1
1610              LDA #$F
1620              STA SECTOR
1630              DEC TRACK
1640              LDA TRACK
1650              CMP #$3
1660              BNE RTS1
1670              LDA #1
1680              STA TRACK
1690 RTS1         RTS
1700 EXIT         LDX #0
1710 LOOP3        LDA $7100,X
1720              STA $0,X
1730              INX
1740              BNE LOOP3
1750              JSR $3275
1760              JMP $33D9
```

SPELLER.SAVER SOURCE CODE

```
1000 * ------------------------------
1010 * SPELLER.SAVER
1020 * THIS FILE SAVES THE SENSIBLE
1030 * SPELLER FROM MEMORY TO
1040 * VARIOUS SECTORS OF AN
1050 * UNPROTECTED DISK
1060 * ------------------------------
1070
1080              .OR $D000
1090              .TF SPELLER.SAVER
1100
1110 DRIVE        .EQ $B7EA
1120 TRACK        .EQ $B7EC
1130 SECTOR       .EQ $B7ED
1140 BUFHI        .EQ $B7F1
1150 COMMAND      .EQ $B7F4
1160              LDA #$F
1170              STA SECTOR
1180              LDA #$5
1190              STA TRACK
1200              LDX #1
1210              STX DRIVE
1220              DEX
1230              STX BUFHI-1
1240              LDA #$3A
1250              STA BUFHI
1260 LOOP         JSR WRITE
1270              DEC BUFHI
1280              LDA BUFHI
1290              CMP #$8
1300              BCS LOOP
1310              LDA #$3
1320              STA BUFHI
1330              JSR WRITE
1340              LDA #$77
1350              STA BUFHI
1360              JSR WRITE
1370              LDA #$71
1380              STA BUFHI
1390 WRITE        LDA #2
1400              STA COMMAND
1410              LDA #$B7
1420              LDY #$E8
1430              JSR $B7B5
1440              DEC SECTOR
1450              BPL RTS1
1460              LDA #$F
1470              STA SECTOR
1480              DEC TRACK
1490              LDA TRACK
1500              CMP #$3
1510              BNE RTS1
1520              LDA #1
1530              STA TRACK
1540 RTS1         RTS
```

Dear Subscribers:

CHECK YOUR LABEL. Your subscription may expire with this issue.
You may renew your subscription using the form on page 23.

Super IOB

By Ray Darrah

Requirements:
An Apple][Plus
Disks that need to be modified

Editors note: Although "Super IOB" has appeared in "The Best of Hardcore Computing," the text and program that follows have been updated. The main difference is the addition of a subroutine to the program but subtle changes have been made in the article as well.

As Hardcore COMPUTIST readers may recall, the IOB program is a simple BASIC program that performs softkeys. IOB stands for Input Output control-Block. It is a list of parameters used by the Read Write Track Sector (RWTS) subroutine.

In the course of time, HARDCORE COMPUTING (old series) and Hardcore COMPUTIST have published several IOB programs (or IOB modifications). These were useful not only for copying different types of disks but for configuring the program to different machines.

Presented here is an advanced version of the original IOB program. We're calling it "Super IOB." Included are the most useful subroutines from all the previous IOB programs. Some new features:

1) The controller isn't spread throughout the program.

2) Half tracks can be accessed.

3) Super IOB is self-configuring.

4) Incorrectly numbered tracks can be copied.

5) The controller performs sector modifications DURING the copy process.

6) A range of seven tracks are read at one time to cut down the disk swaps on single drive systems.

7) Super IOB can do everything MUFFIN PLUS and DEMUFFIN PLUS can.

8) Automatic error trapping is now included.

Using The Program

Start by entering the Applesoft listing, then

SAVE SUPER IOB

Next, enter the hexdump and

BSAVE IOB.OBJ0,A$300,L$5C

A third file is required in order to copy disks that have been protected with a 13 sector format.

RWTS.13

To read the protected DOS 3.2 disks, Super IOB uses an image of the 3.2 RWTS. By performing a swap of the image with the RWTS currently in memory, diskettes with different formats can be accessed.

Use BOOT13 from the system master disk to get DOS 3.2 into your 3.3 machine.

Once DOS 3.2 is booted up, all you have to do is BSAVE the RWTS.

BSAVE RWTS.13,A$B800,L$800

What It Does

Super IOB de-protects disks by pushing the RWTS to its upper-most limits. Because of this, it only works on disks with sectors somewhat resembling normal DOS. Before a disk can be "Softkeyed", the protection scheme must be determined. The easiest way to do this is to use a program (like "The CIA", "Bag of Tricks" or "DiskView") which allow you to discover the difference between normal sectors and the ones on the intended disk.

Once the protection has been discovered, all that needs to be done is the insertion of a controller program (lines 1000 through 9999) into Super IOB. Here is a list of the protection schemes Super IOB was designed to Softkey:

1) Altered data, address, prologue, or epilogue marks

2) Strangely numbered sectors or tracks

3) Modified RWTS (with same entry conditions)

4) Half tracks for any of the above

5) Thirteen or 16 sector format for any of the above

The following is a brief description of each protection scheme and how it relates to Super IOB:

Altered Marks

Making mark alterations to DOS is a technique that is used on a lot of the earlier disks. DOS puts certain reserved bytes on the disk (during INITialization) so it can tell where a sector (and other valuable information) begins. For example, a normal 16-sector disk has the bytes: D5 AA AD designating the start of the data field which contains the 256 bytes of data in encoded form. When a standard RWTS tries to find a sector, it looks for these marks. If they are not found (either because they don't exist or they have been changed to something else) DOS returns with the dreaded I/O ERROR.

The sequences of the four reserved-byte marks (start of address, end of address, start of data, end of data) are handled by subroutines in Super IOB. These subroutines simply change the marks the RWTS looks for or modify the RWTS so that it doesn't look for them at all (depending upon the mark).

Strangely Numbered Sectors

On some disks the numbers which tell the RWTS what sector is currently passing under the read/write head, have been tampered with. These disks are easily softkeyed with Super IOB. The controller simply reads using the strange sector numbers.

"Super IOB deprotects disks by pushing the RWTS to its uppermost limits. Because of this, it only works on disks with sectors somewhat resembling normal DOS. Before a disk can be "Softkeyed", the protection scheme must be determined."

This works because the RWTS compares the sector number found on the disk with the one the controller is looking for (even if it is higher than 15). Later, when writing, standard sector numbers are used, thus deprotecting the disk!

Modified RWTS

The disk-protectors will often rearrange and/or modify the standard RWTS subroutine. When this happens, all one has to do is to make a controller program which reads, using the strange RWTS and then swaps with a normal RWTS and writes the information back out.

Since the RWTS of a protected disk will be modified to read any altered DOS marks, this is a good method to use if you are unable to determine what they have been changed to.

Anatomy Of A Controller

Before we attempt to write a controller, let's look at its format. Here is an explanation of the subroutines (and sub-programs) in the Super IOB program that are at the controller's disposal.

1) Start Up
Lines 10-60

The first few lines identify the program. Line 60, however, sets HIMEM and LOMEM so that they fit the memory usage requirements (see memory map following). It then goes to "CONFIGURATION TIME."

2) Initial IOB Set-Up
Line 80

This subroutine is normally GOSUBed via "TOGGLE READ / WRITE." Its purpose is to reset the buffer page and set the drive number, slot number and volume number to the disk to be accessed next.

3) R/W Sector
Line 100-110

This subroutine is GOSUBed directly from the controller. It reads or writes (depending upon CD) at the specified track and sector.

4) Move S Phases
Lines 130-140

Moves the disk drive head by the number of phases specified by S; one phase equals one half-track. It is capable of moving in either direction up to 128 phases (or 64 tracks). When moving the head, this routine doesn't let the RWTS know that the head has been moved. Therefore, this subroutine makes it possible to copy disks that have track mismarkings. Care should be taken when moving a great number of phases that PH + S isn't greater than 255 or less than 0, otherwise an error will occur.

5) Ignore Checksums & End Marks
Line 170 (16 sector RWTS)
Line 270 (13 sector RWTS)

These routines do a few POKEs into their corresponding RWTS. The final result is that the RWTS no longer looks for epilogue marks or checksums when searching for a sector.

6) Altered Address Marks
Line 190 (16 sector RWTS)
Line 290 (13 sector RWTS)

These modify the RWTS (via POKE) so that it looks for a different sequence of address prologue marks. The decimal values of the marks to look for should be stored in DATA statements in the "DATA FOR MARKS" area.

7) Altered Data Marks
Line 210 (16 sector RWTS)
Line 310 (13 sector RWTS)

These are the same as the previous subroutine except for DATA prologue marks.

8) Normalizer
Lines 230-240 (16 sector RWTS)
Lines 330-340 (13 sector RWTS)

This restores the values in the RWTS subroutine that are messed up by the three previous routines. This routine should be called just before writing, when using only one RWTS (assuming of course that one of the previous routines was called before reading).

9) Exchange RWTS
Line 360

This is the standard swap RWTS's routine. It exchanges the RWTS at $1900 with the one at $B800, which is the normal residing place for an RWTS. To tell the swap routine (which is invoked by a CALL 832) what to exchange, a few POKEs must be executed. They are:

POKE 253, start of first location
POKE 255, start of second location
POKE 224, number of pages (a standard RWTS is eight pages long)

10) Format Disk
Lines 380-410

Formats the target disk. This is meant to be used before the softkey operation begins (and is GOSUBed by "CONFIGURATION TIME") but can be called by the controller should the need arise.

11) Print Track & Sector #
Line 430

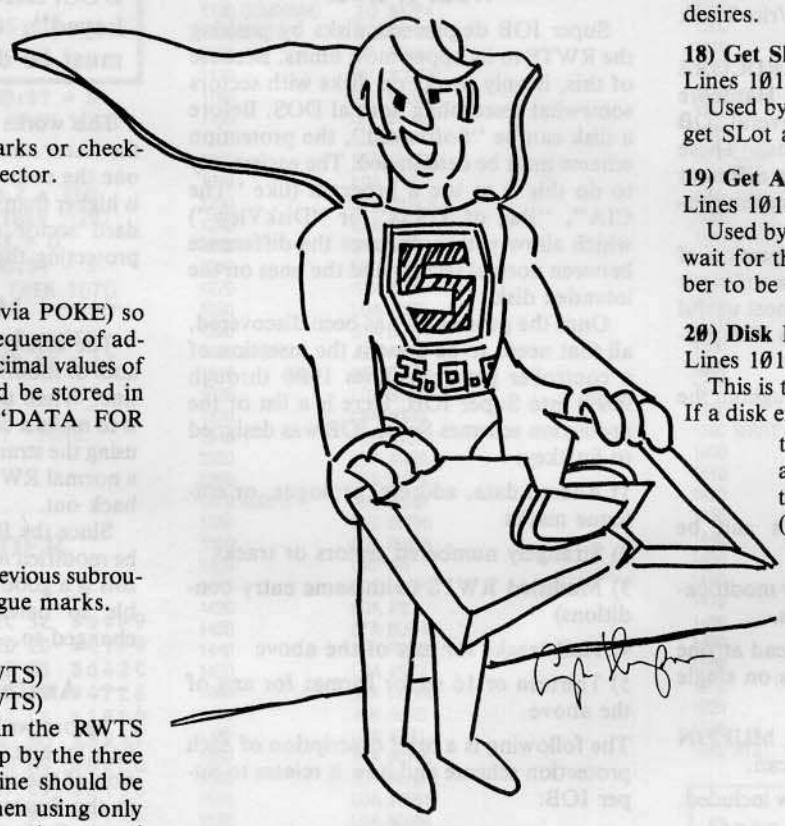
This is the subroutine that puts the current track and sector number at the top of the screen during the softkey operation.

12) Center Message
Line 450

Centers a message (contained in A$) at the current VTAB position and RETURNs.

13) Print Message & Wait
Line 470

This routine uses "CENTER MESSAGE" to print the intended message at a VTAB of 11 and then it prints "PRESS ANY KEY TO CONTINUE." After this, it waits for a key to be pressed and RETURNs.



14) Toggle Read/Write
Lines 490-530

This routine toggles the state of CD (from ReaD to WRite and vice versa) and prints the current mode in flashing letters at the very top of the screen. In addition, if the user has only one drive, it asks him to swap disks. It then exits via "INITIAL IOB SETUP," thus making the sector buffer ready for the next operation.

15) Ignore Unreadable Sectors
Lines 550-590

If the controller should pay no attention to unreadable sectors then somewhere in the begginning of it should be an 'ONERR GOTO 550. This is used usually with RWTS.13 (since DOS 3.2 sectors are unreadable until they have been written to) but can be used with any disk that has unreadable sectors which should be ignored.

16) Controller
Lines 1000-9999

These are the line numbers set aside for the controller. This area should have all of the controller and subroutines (sector edits and the like). Before using this, please see the memory map that follows.

17) Configuration Time
Lines 10000-10090

This routine asks the user which slots and drive numbers to use for the various disks. It also formats the target disk if the user so desires.

18) Get Slot & Drive #
Lines 10110-10130

Used by "CONFIGURATION TIME" to get SLot and DriVe information.

19) Get A Key
Lines 10150-10170

Used by "GET SLOT AND DRIVE#" to wait for the appropriate drive or slot number to be typed.

20) Disk Error
Lines 10190-10270

This is the normal error-trapping routine. If a disk error occurs, this routine will print the error message, otherwise it will assume that the error is in the controller and the program will crash (CALL 834).

21) Data For Marks
Lines 62010-63999

These line numbers should contain the appropriate data (if any) required for any altered mark routine.

Note: in the above line number description, the line numbers consisting of REMs have been ommitted. They may be excluded (although it is not recommended) when typing in the program.

Now that you have an idea of the subroutines, take a look at the following variables and note how they relate to them. While examining the table on the next page, it would be a good idea to observe the BASIC that makes up the previously listed subroutines. This will give you a better idea of how things are accomplished in Super IOB.

Table 1

A - general temporary usage, scrambled by "MOVE S PHASES."

A$ - holds message to pass to the user via "CENTER MESSAGE" and "PRINT MESSAGE AND WAIT," scrambled by "TOGGLE READ / WRITE."

A1,A2,A3 - scrambled by any "ALTERED ADDRESS MARKS" or "ALTERED DATA MARKS" routine, they are READ from DATA statements and POKEd into the appropriate RWTS subroutine to change the marks it looks for.

B$ - altered only during configuration.

BF - buffer full, holds the status of the sector buffer, set to 1 if the buffer is either full or empty and to 0 if neither; changed only by "R/W SECTOR."

BUF - buffer location, holds the address where the RWTS is expecting to find the page number of the sector; used by "INITIAL IOB SETUP" and "R/W SECTOR." A (PEEK(BUF)-1)*256 will return the address of byte zero in the last read sector.

CD - command code, used by the controller and "TOGGLE READ / WRITE," holds the current RWTS command code; only POKEd in by "INITIAL IOB SETUP" (see RD, WR, and INIT)

CMD - Command code location, holds the address where the RWTS is expecting to find the previously stated command code; used by "INITIAL IOB SETUP." A POKE CMD,CD will change the IOB command.

D1 - drive 1, set during configuration to the drive number of the source drive; used by "TOGGLE READ/WRITE".

D2 - drive 2, same as above except for target drive.

DOS - Disk Operating System, the number of sectors to read or write; initialized to 16.

DRV - drive location, holds the address where the RWTS is expecting to find the drive number of the drive to be accessed; used by "INITIAL IOB SETUP" to change the IOB drive number. A PEEK(DRV) will return the drive last accessed.

DV - current drive, used by "INITIAL IOB SETUP," 'TOGGLE READ/WRITE" and "MOVE S PHASES;" holds the drive number of the drive to be accessed next.

ERR - error code, used by "DISK ERROR" to determine the error that has just occured.

INIT - initialize command code, a CD = INIT will set the command code to format the diskette.

IO - Input/Output location, normally holds a 768 (set during configuration); CALLed by "R/W SECTOR" to induce the RWTS subroutine. To use a relocated RWTS, the controller must have a IO = IO + 42 statement.

MB - maximum buffer page, holds the last page of memory for the sector buffer; used by "R/W SECTOR," initialized (during configuration) to 151 and should be changed to 130 only when a 13-sector disk is read or written.

OVL - old volume location, a PEEK(OVL) will return the volume number of the previously accessed (via "R/W SECTOR") diskette.

PH - current phase, if "MOVE S PHASES" is referenced (by the controller), this variable must contain the disk arms' current phase number (PH = 2*TK).

RD - read command code, a CD = RD will set the command to read the disk.

S - step, used to tell "MOVE S PHASES" how may phases to step through (-120 to 120).

S1 - slot 1, set to the slot number of the source drive during configuration; used by "TOGGLE READ/WRITE."

S2 - slot 2, same as above except for target drive.

SCT - sector number location, holds the address where the RWTS is expecting to find the sector to be accessed; used by "R/W SECTOR" to tell the RWTS which sector is to be read or written. A PEEK(SCT) will return the last accessed sector number.

SLT - slot number location, holds the address where the RWTS is expecting to find the slot number of the disk to accessed next; used by "INITIAL IOB SETUP." A PEEK(SLT) will return the last accessed disk's slot number.

SO - slot number, used by "TOGGLE READ/WRITE" and "INITIAL IOB SETUP;", holds the slot number of the disk to be accessed next.

ST - sector number, used by the controller to tell "R/W SECTOR" what sector number is to be read or written next.

TK - track number, used by the controller to tell "R/W SECTOR" what track is to be accessed next.

TRK - track number location, holds the memory location where the RWTS is expecting to find the track to be accessed. A PEEK(TRK) will return the last accessed track number.

VL - volume number, used by the controller to tell "TOGGLE READ / WRITE" (which passes it to "INITIAL IOB SETUP") the volume number of the disk to be accessed next.

VL$ - altered only by "FORMAT DISK."

VOL - volume number location, holds the memory location where RWTS is expecting to find the volume to be accessed. A PEEK(VOL) will return volume number last used by the controller.

WR - write command code. A CD = WR will set the command to write.

Memory Usage

Before actually looking at some controllers, let's say a few words about memory usage.

Following, is a memory allocation table for the various parts of Super IOB. It is extremely important to stay within the boundaries when writing a controller. Otherwise, horrible things might happen (the least of which would be the production of an incorrect copy).

Table 2

$0800.$18FF (2048-6399) intended for the Applesoft part of Super IOB

$1900.$20FF (6400-8447) space allocated for a moved RTWS (RWTS.13 or other)

$2100.$26FF (8448-9983) BASIC variable space

$2700.$96FF (9984-38655) used for the sector buffer

First, notice the amount of space available for the BASIC program. The Super IOB program as listed (with all REMs) ends about 1200 bytes short of the final designated location. This means that the controller (and all DATA statements) must fit into this 1K area. In view of the space requirement, the end of program should be checked by typing

PRINT PEEK(175) + PEEK(176)*256

before a new controller is used.

If it has exceeded the 6399 limit, I suggest DELeting all subroutines not referenced by the controller and all REM lines until it fits within the allocated space.

However, if the program does NOT use a relocated RWTS, then the extra 2K allocated for an RWTS can be used for the BASIC. In this situation, the end of the program should only be checked with very long controllers, since 3K ought to be enough for any softkey operation.

Second, observe the 1534 bytes for variables. This should be enough space for the simple softkey procedure. It is impossible to allocate more memory for variables and use a relocated RWTS file. If you find that you need more memory and the program does not use RWTS.13 or some other moved RWTS, then the LOMEM: 8448 statement in line 60 may be ommitted. This will allocate what isn't used (by the BASIC program) of the 2K area reserved for the relocated RWTS as variable space.

Never omit the "HIMEM:" statement! This could cause variables to overflow into the sector buffer, thus making a faulty copy.

With all this new knowledge, we are finally ready to scrutinize some sample controller programs. Keep in mind that protection schemes can be used with one another. Therefore, a more sophisticated controller for Super IOB will probably be required for most softkeys. Even so, developing new controllers isn't difficult.

Standard Controller

```
0000 REM STANDARD CONTROLLER
1010 TK = 0:ST = 0:LT = 35:CD = WR
1020 T1 = TK: GOSUB 490
1030 GOSUB 430: GOSUB 100:ST = S
     T + 1: IF ST < DOS THEN 1030
1040 IF BF THEN 1060
1050 ST = 0:TK = TK + 1: IF TK <
     LT THEN 1030
1060 GOSUB 490:TK = T1:ST = 0
1070 GOSUB 430: GOSUB 100:ST = S
     T + 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 : PRINT "DONE
     WITH COPY": END
```

Here is how the standard controller works:

Unique Variables

The following variables are used by the controller exclusively. Other variables used by the controller are for interaction with various subroutines in Super IOB.

LT - this variable holds the last track to be accessed (it is the last track plus one). For example, if line 1010 were to have an LT = 15 (instead of LT = 35) then it would only copy tracks 0-14.

T1 - holds the track number (TK) for the transition of read to write and vice versa.

Line Explanation

1000 - identifies controller.

1010 - initializes variables.

TK = 0 - sets the starting track to zero.

ST = 0 - sets the starting sector to zero.

LT = 35 - sets the last track to 34.

CD = WR - sets command code to write.

1020 - The read routine. It begins by saving the current track number and then gets the source disk.

1030 - prints the current track and sector number, reads in the sector and increments the sector number. If it is less than DOS (in this case 16), then it reads another sector.

1040 - if the sector buffer is full, it goes to the write routine.

1050 - resets the sector number to zero and increments the track number. If it is not past the last track, it reads the new track.

1060 - this is the beginning of the write routine. It gets the write drive and starts at the previously saved track (T1), sector zero.

1070 - prints the current track and sector number, writes the sector to the disk and increments the sector number. If it is not finished with this track, it writes another sector.

1080 - resets the sector number and increments the track number. If the sector buffer isn't empty and it's not past the last track, it writes another track.

1090 - if it is not done duplicating the disk (i.e, not past last track), it reads some more tracks.

1100 - tells user that everything is OK and that the disk is copied.

Even though this controller only copies normal DOS 3.3 disks, I recommend saving it anyway. This controller is the basic (pun intended) building block for more complex controllers.

Load the original Super IOB program

LOAD SUPER IOB

Type in the controller listed above.Save this new program

SAVE IOB.STANDARD.CON

You now have the capability (I'm sure you did before) to copy a regular diskette. Because you probably don't think this is so exciting, we'll move on to the de-protection of Castle Wolfenstein. I chose this game because its controller is a simple example of what a few modifications to the standard controller can accomplish.

Swap Controller

```
1000 REM SWAP RWTS CONTROLLER
1010 TK = 3:ST = 0:LT = 35:CD = W
     R
1020 T1 = TK: GOSUB 490: GOSUB 36
     0
1030 GOSUB 430: GOSUB 100:ST = S
     T + 1: IF ST < DOS THEN 1030
1040 IF BF THEN 1060
1050 ST = 0:TK = TK + 1: IF TK <
     LT THEN 1030
1060 GOSUB 490:TK = T1:ST = 0: GOSUB
     360
1070 GOSUB 430: GOSUB 100:ST = S
     T + 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 "EVERYTHING O.
     K. DOS NOT COPIED": END
10010 PRINT CHR$ (4)"BLOADRWTS,
     A$1900"
```

This controller is the controller that swaps RWTSs before reading and writing. It can be used for a great many things. By modifying this one slightly we can copy Castle Wolfenstein.

Castle Wolfenstein Controller

```
1000 REM CASTLE WOLFENSTEIN CONT
     ROLLER
1010 TK = 3:ST = 0:LT = 35:MB = 1
     30:CD = WR:DOS = 13
1020 T1 = TK: GOSUB 490: GOSUB 36
     0: ONERR GOTO 550
1030 GOSUB 430: GOSUB 100:ST = S
     T + 2: IF ST < DOS * 2 THEN
     1030
1040 IF BF THEN 1060
1050 ST = 0:TK = TK + 1: IF TK <
     LT THEN 1030
1060 GOSUB 490:TK = T1:ST = 0: GOSUB
     360
1070 GOSUB 430: GOSUB 100:ST = S
     T + 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 "EVERYTHING O.
     K. DOS NOT COPIED": END
10010 IF PEEK (6400) < > 162 THEN
      PRINT CHR$ (4)"BLOAD RWTS.
     13,A$1900"
```

Castle Wolfenstein uses "Strangely Numbered Sectors" as its protection scheme. Luckily, they aren't so strange that a complex algorithm is needed to calculate the next number. Instead, they are merely even-numbered DOS 3.2 sectors (0-24).

When 13-sector DOS gets these sector numbers, it doesn't accept them and returns with I/O error. But the 13-sector RWTS doesn't care about the actual number on the sector, as long as it matches up with the sector number you want to access. Thus, all one has to do is read with the strange sector numbers and write with the normal ones.

Here is a line-by-line explanation of the differences that make this controller successful:

1000 - identifies controller.

2010 - start at track three (bypass DOS tracks) and set MB and DOS to their 13-sector values.

1020 - since we want to use RWTS.13 to read with, swap it in.

1030 - counts from 0 to 24 by two's.

1060 - swaps the normal RWTS back into its original location for the write ahead.

1100 - tells the user that the copy has no DOS on it.

10010 - BLOADs the 13-sector RWTS at $1900.

As noted in line 10010, once the copy has been made there will be no DOS on the de-protected version. This isn't a problem as long as you don't boot with it.

Super IOB BASIC program

```
10 REM ************************
20 REM **       SUPER IOB      **
30 REM **    BY RAY DARRAH     **
40 REM ************************
50 REM SET HIMEM BELOW BUFFER AND
   SET LOMEM ABOVE THE BLOADED RWTS
```

Continued on page 22

ProDOS To DOS:

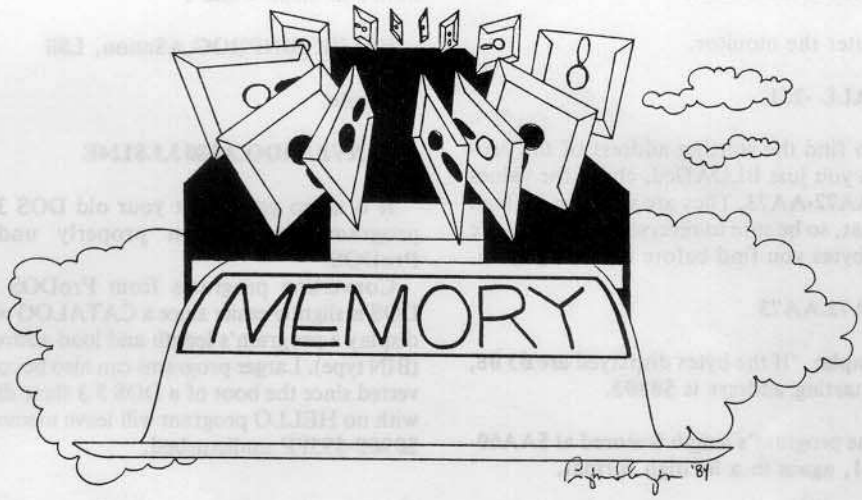
Single Drive Conversion Technique

By Jimmy Eubanks Jr.

Requirements:

64K Apple II+, IIe, etc.
One disk drive
DOS 3.3 slave disk
with null HELLO program
PRODOS Disk

*Editor's Note: One of the frustrations I encountered when first experimenting with Apple ProDOS was the inability of the DOS-ProDOS Conversion Program (DUCK) to transfer files between the two operating systems with only one disk drive. The ProDOS User's Manual inexcusably neglects to mention this limitation, and after several unsuccessful attempts at file transfer I was directing every expletive I could think of at a certain Fortune 500 company with headquarters in Cupertino, CA. Fortunately for single drive ProDOS users, Jimmy Eubanks has come up with a straightforward technique that will transfer **most** Applesoft and Binary programs between the two operating systems.*



This technique will allow those ProDOS users with only one disk drive to transfer Applesoft BASIC files and Binary files between ProDOS and DOS 3.3 disks in either direction. This is accomplished by loading the program from the source disk, then hiding it in an area of memory that is not disturbed by the boot process. Once the destination operating system has been booted, the program can be restored to its proper memory location and saved.

For the boot of a DOS 3.3 slave disk, all memory between $900 and $95FF is left untouched so files up to 193 sectors (86 blocks) in length can be transferred from ProDOS to DOS. ProDOS, however, does not tiptoe through memory quite so lightly during its boot. The only large portion of memory that is left undisturbed by a ProDOS boot is $5C00-$95FF. This limits the size of file that can be converted from DOS to ProDOS to those with 59 or fewer sectors (30 blocks).

To Transfer Applesoft BASIC Files From DOS To ProDOS:

1) Boot up DOS 3.3.

PR#n (n being the slot your disk interface is in)

2) Load the program you want to convert.

LOAD BASICPROG

3) Enter the monitor.

CALL -151

4) Find the highest memory location of the program now in memory by checking the values at $AF and $B0. These will be listed with the high order byte last, so reverse the order of the the two values you find.

AF.B0

Example: If the bytes displayed are AA 1B then the end of the program is at $1BAA.

5) You now need to calculate the length of the program and add it to $5C00 to determine where the end of the relocated program will be. To get the length, subtract $800 from the hex value found in step 4. This is the Applesoft program's length. Add this value to $5C00.

Example: For a program whose end is at $1BAA

$1BAA-$0800 = $13AA - program length
$5C00 + $13AA = $6FAA - end of relocated program

6) Move the program to $5C00.

5C00<800.(value found in step 4)M

Example: For a program whose end is at $1BAA

5C00<800.1BAAM

7) Boot a ProDOS Disk.

n^{CTRL}P (n being the slot your disk interface is in.)

8) Move the program back to its original location using the end of relocated program address found in step 5.

800<5C00.(address found in step 5)M

Example: If the end of the relocated program is at $6FAA

800<5C00.6FAAM

9) Restore the end of program pointers at $AF-BO to the values you found there in step 4.

AF: ll hh

Example: For a program whose original ending address is at $1BAA

AF:AA 1B

10) Hit the RESET key to re-enter Applesoft

11) Save your program.

SAVE BASICPROG

To Transfer Binary Files From DOS 3.3 To ProDOS

1) Boot up a DOS 3.3 disk.

PR#n (n being the slot your disk interface is in)

2) BLOAD the file you want to transfer.

BLOAD BINPROG

3) Enter the monitor.

CALL -151

4) To find the starting address of the program you just BLOADed, check the values at $AA72-AA73. They are stored in lo/high format, so be sure to reverse the order of the two bytes you find before recording them.

AA72.AA73

Example: If the bytes displayed are 03 08, the starting address is $0803.

5) The program's length is stored at $AA60-AA61, again in a lo/high format.

AA60.AA61

Example: If the bytes displayed are 4E 12, then the program's length is $124E bytes.

6) The end of the program at its original and relocated addresses also needs to be calculated. To do this add the length found in step 5 to the load address to find the original ending address. Likewise, add the length to $5C00 to find the relocated ending address.

Example: If the program's original address is $0803 and its length is $124E

$0803 + $124E = $1A51 - end of original program
$5C00 + $124E = $6F4E - end of relocated program

7) Hide the program at $5C00.

5C00 < starting address(step 4).ending address(step 6)M

Example: For a program which starts at $0803 and ends at $1A51

5C00 < 803.1A51M

8) Boot a ProDOS disk.

n^{CTRL}P (n being the slot your disk interface is in)

9) Move the program back to its original location using the relocated ending address calculated in step 6.

starting address(step 3) < 5C00.relocated ending address(step 6)M

Example: For a program whose relocated ending address is at $6F4E

803 < 5C00.6F4EM

10) Hit the RESET key to re-enter Applesoft

11) BSAVE your program using the proper address and length parameters you determined in steps 4 and 5

BSAVE BINPROG,A$nnnn, L$ll

Example:

BSAVE FIDO,A$803,L$124E

It is up to you to get your old DOS 3.3 programs to function properly under ProDOS.

Converting programs from ProDOS to DOS is slightly easier since a CATALOG will display a program's length and load address (BIN type). Larger programs can also be converted since the boot of a DOS 3.3 slave disk with no HELLO program will leave memory $0900-$95FF undisturbed.

Converting Applesoft Programs From ProDOS To DOS 3.3

1) Boot up ProDOS

PR#n (n being the slot your disk interface is in)

2) Load the program you want to convert.

LOAD BASICPROG

3) CATALOG the disk and write down the decimal file length that is displayed for the program you wish to convert. Convert this value to hexadecimal and add $0801 to it. Also add the hex file length to $4001 to come up with the end of the relocated program.

Example: If the length displayed by CATALOG is 4540.

4540(decimal) = $116C(hexadecimal) - program length
$0801 + $116C = $19BD - end of program
$4001 + $116C = $516D - end of relocated program

4) Enter the monitor.

CALL -151

5) Move the program to $4000.

4000 < 800.(end of program value found in step 3)M

Example:

4000 < 800.19BDM

6) Boot a DOS 3.3 slave disk with a "null" HELLO program.

n^{CTRL}P (n being the slot your disk interface is in.)

7) Move the program back to its original location using the end of relocated program address you found in step 5.

0800 < 0900.(relocated end of program value found in step 3)M

Example:

800 < 4000.516DM

8) Restore the end of program pointers at $AF-BO so they are correct. You found these values in step 3. Do not forget the pointers are in a lo/high format.

Example: If the original end of the program was at $19BD type:

AF: BD 19

9) Hit the RESET key to re-enter Applesoft

10) Save your program.

SAVE name of program

Converting Binary Programs From ProDOS To DOS 3.3

1) Boot up a ProDOS disk.

PR#n (n being the slot your disk interface is in)

2) CATALOG the disk and write down the load address and length that is displayed for the program you wish to convert. Convert the length to hexadecimal and add it to $0900 to come up with end of the relocated program.

Example: If the program length displayed is 2048

2048(decimal) = $0800(decimal)
$0800 + $0900 = $1100 - end of relocated program

3) BLOAD the program at $0900

BLOAD PROGRAM,A$900

4) Enter the monitor.

CALL -151

5) Boot a DOS 3.3 slave disk with a null HELLO program

Continued on page 18

READER QUESTIONNAIRE

Dear READER:

The editors at Hardcore COMPUTIST are in the process of planning a regular column highlighting Hardware Projects which can be completed by readers who possess a basic understanding of electronics.

To help us provide you with the kinds of articles and columns that you want, please complete the following questions and return this form along with your name and address to:

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

For those readers who feel a Hardware Column WOULD NOT BE OF INTEREST to them...Please respond! We need your answers and comments to assure an accurate sampling of reader opinion.

My age group is:

- ☐ 13 and under
- ☐ 14-24
- ☐ 25-34
- ☐ 35-44
- ☐ 44-65
- ☐ 65 and Over

Sex

- ☐ Male
- ☐ Female

Level of Education

- ☐ High School graduate
- ☐ Bachelor's degree
- ☐ Master's
- ☐ Doctorate
- ☐ Other_____

I would be interested in reading a Hardware Project column in future issues of Hardcore COMPUTIST.

- ☐ Definitely
- ☐ Maybe
- ☐ Probably Not

Projects I would be interested in would be in a cost range of:

- ☐ Under $25
- ☐ $25-75
- ☐ $75-150
- ☐ Steak and lobster for 10 or more

My strongest area of skill is:

- ☐ Programming
- ☐ Both
- ☐ Electronics
- ☐ Neither

Hardcore COMPUTIST publishes articles that are useful to me.

- ☐ Always
- ☐ Sometimes
- ☐ Not usually

Programming and softkey techniques are carefully explained and provide all the information I need to complete a project.

- ☐ Always
- ☐ Sometimes
- ☐ Not usually

I would like to see the following kinds of articles in future issues:(Check all boxes which apply)

- ☐ Deprotecting business programs
- ☐ Deprotecting applications & utilities programs
- ☐ Deprotecting games
- ☐ Reviews
- ☐ Hardware and software modifications
- ☐ Bit-copy parameters
- ☐ APT's
- ☐ Adventure Tips
- ☐ CORE
- ☐ Other_____

How many others, besides yourself, read your copy of Hardcore COMPUTIST?

- ☐ one person
- ☐ 2-4
- ☐ 5 or more

After reading each issue, do you

- ☐ Give it to a friend
- ☐ Add it to your reference library
- ☐ Use it to line the cat box
- ☐ Other

Do you often purchase products advertised in Hardcore COMPUTIST?

- ☐ Yes
- ☐ No

Approximately how many hours do you spend each week using your computer?

- ☐ under 5
- ☐ 5-10
- ☐ 10-20
- ☐ over 20 (before, during, and after dinner; before bedtime, instead of bedtime)

Which Apple computers do you own or use?
(Check all boxes which apply)

- ☐ Apple][
- ☐ Apple //e
- ☐ Apple][Plus
- ☐ Apple //c
- ☐ Apple compatible_____

Which Operating System(s) do you use?

- ☐ 3.3
- ☐ ProDOS
- ☐ 3.2
- ☐ Apple Pascal

How many disk drives do you own?

- ☐ one
- ☐ hard disk
- ☐ two
- ☐ not enough
- ☐ RAM

Which of these accessories do you own?
(Check all boxes which apply)

- ☐ modem
- ☐ language card
- ☐ printer
- ☐ modified ROM
- ☐ copycard
- ☐ 80-column card
- ☐ ROM card
- ☐ Other _____

In the use of computers, I think of myself as a

- ☐ Novice (Which button do I push?)
- ☐ Definite Intermediate (Sure. I can mess things up with real self-confidence)
- ☐ Full-time Hacker (Shave my beard? What beard?)

Comments

.....

.....

Name _____

Address_____

City _____ **St** _____ **Zip** _____

Continued from page 16

n^{CTRL}P (n being the slot your disk interface is in)

6) Move the program back to its original location using the relocated ending address calculated in step 2.

starting address<0900.relocated ending address(step 2)M

Example: If the CATALOG display showed a load address of $2000 and a length of $0800 type:

2000<0900.1100M

7) Hit the RESET key to re-enter Applesoft.

8) BSAVE your program with the address and length parameters which the ProDOS CATALOG displayed

BSAVE PROGRAM,A$nnnn, LSll

Using ProDOS On A Franklin Ace

The March, 1984 issue of Apple Assembly Lines contained a short article containing instructions on how to successfully boot ProDOS on a Franklin Ace computer. This method involved NOPping two bytes in the ProDOS system file after it had been loaded into memory. However, this method would not work for the ProDOS file dated 1-JAN-84.

The ProDOS system file contains a checksum-like subroutine which returns a value of $0C if a genuine Apple is detected and a $00 otherwise. If a non-Apple is detected ProDOS will just hang up and not load in the BASIC.SYSTEM interpreter. A disassembly of this routine from the 1-JAN-84 ProDOS file looks like:

```
2639-18        CLC
263A-AC 31 26  LDY  $2631
263D-B1 0A     LDA  ($0A),Y
263F-29 DF     AND  #$DF
2641-6D 31 26  ADC  $2631
2644-8D 31 26  STA  $2631
2647-2E 31 26  ROL  $2631
264A-C8        INY
264B-CC 34 26  CPY  $2634
264E-D0 ED     BNE  $263D
2650-98        TYA
2651-0A        ASL
2652-0A        ASL
2653-0A        ASL
2654-0A        ASL
2655-A8        TAY
2656-4D 31 26  EOR  $2631
2659-69 0B     ADC  #$0B
265B-D0 03     BNE  $2660
265D-A5 0C     LDA  #$0C
265F-60        RTS
2660-A9 00     LDA  #$00
2662-60        RTS
```

In order for this routine to always return with a value of $0C, the branch to the code which loads the accumulator with $00 (LDA #$00) needs to be removed. This can be done by replacing the BNE $2660 (D0 03) instruction with two NOP's (EA EA).

The easiest way to do this is to use a sector editor and zap the change directly to the disk. Any sector editor can be used on a ProDOS disk because the formatting has not been changed from DOS 3.3. On my copy of the ProDOS USERS DISK the change I had to make was to bytes $5B and $5C of track $01, sector $09. I changed them from D0 03 to EA EA and rewrote the sector.

If you have a sector editor with search capability, such as ZAP, you should search for a byte sequence of 69 0B D0 03, since the sequence of D0 03 is a fairly common one.

Once you have made this change, ProDOS should boot and operate on Franklin's and other Apple compatibles that have their monitor ROM routines in the proper locations.

Bugs in Hardcore COMPUTIST no.8

In our last issue there was an error in step 12 of the article "Breaking Windows:Softkey for Legacy of Llylgamyn" on page 10. The last byte listed in that step should be ED not FD. Step 12 should therefore read:

12) Start editing at byte $15, entering the following bytes

D0 16 EA AD 2D 00 CE FB 00 D0 F8 AD DE 00 A9 01 48 A5 01 48 A5 00 48 60 A9 00 F0 ED

Another detail which might cause problems for some readers is in the box on page 15 which contains instructions for creating DeMuffin Plus. In step 6 there should be an asterisk (*) following the ^{CTRL}Y. Step 6 should read:

6) Tell the monitor what is being moved and where it is going

1900<B800.BFFF ^{CTRL}Y

Most Wanted List

Reader response to our "Most Wanted List" has been very favorable. We have received softkeys for a number of programs previously in our list and lots of votes for programs to add to the list. We will be publishing the softkeys we have received just as soon as they have been evaluated and edited by our staff.

So, keep those votes and softkeys coming.

If there is a program that you have been pulling your hair out trying to back-up, let us know about it.

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

If you know how to de-protect, unlock or modify any of these programs, we encourage you to help other Hardcore COMPUTIST readers and earn some extra money at the same time. Be sure to 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. Type Attack
Sirius Software
4. DB Master 4.0
Stoneware, Inc.
5. Time Is Money
Turning Point
6. Julius Erving and Larry Bird Go One on One
Electronic Arts
7. Visiblend
Micro Lab
8. Cut And Paste
Electronic Arts
9. Dollars And Sense
Monogram
10. Word Juggler
Quark, Inc.
11. Catalyst
Quark, Inc.
12. Rocky's Boots
The Learning Company
13. PFS Graph
Software Publishing Corp.
14. HAMSOFT
Kaltronics
15. The Statistics Series
Human Systems Dynamics
16. Millionaire
Blue Chip Software
17. Facemaker
Spinnaker
18. Story Machine
Spinnaker

CORE Word Search Generator

By Barry Palinsky

Requirements:
An Apple][plus or compatible
48K RAM
At least one disk drive
Printer

When for some reason my computer becomes unusable (my wife is increasing the size of her phone list or my children spill Pepsi on the motherboard while shooting alien foes) I often find myself looking for entertainment. One thing I have discovered that can provide hours of fun is curling up in front of the television set with a good word search puzzle.

After solving many of these mindbending puzzles, I began thinking how enjoyable it might be to create some word searches of my own. So for a while I was spending my free time scribbling word searches. The best part of making my own searches was that I could pick words that interested (or excited) people.

Although it was enough to make searches using graph paper, I wanted an easier faster way to make them. This is why I made The CORE Word Searcher. If the idea of making word searches for your family or friends sounds fun, I strongly recommend this program.

Typing it In

The first step to creating your own word searches is to key in the program beginning on this page and SAVE it with:

SAVE WORD SEARCHER

Next, type in the hexdump following and save this one by typing:

BSAVE WORD SEARCHER.OBJ, A$300,L$AE

How To Use It

When you RUN the program, the first thing you will be asked is which slot your printer is in. Most printers are in slot 1 and this is probably what you should type. If you do not wish to use your printer, you may type a zero (Ø) indicating the screen as your printer. As with most of the prompts in this program, if you answer them illegally, the program will merely ask you the same question until you give it an acceptable answer.

The next question concerns how many characters per line your printer has. In standard mode (usually 10cpi) this is either 80 or 85. If you wish the program to set up the printer in a non-default mode, you will have to append some printer codes to line 680. If you do this, be sure save this new version of The CORE Word Searcher (preferably under another name).

Next, you will be prompted with DISK ACCESS? (Y/N). If you press "Y", you will find yourself in the disk access subroutine which is explained later.

Any other key will be considered a negative response and therefore will place you at the next input which is "COLUMNS IN WORDSEARCH." This is the number of horizontal characters in the wordsearch. Valid numbers are between 5 and 38 inclusive.

After that you will be asked the number of rows in the wordsearch. This is the number of characters placed vertically in the search. Valid numbers for this are between 5 and 23 inclusive.

The last phase before actually editing the word search is to enter the words to be searched for. The maximum number of words allowed is 75. As each word is typed in, it is placed in two arrays. One array is arranged by length of the word and the other is arranged alphabetically. Because of this, word entry may slow down as you approach the 75 word limit. If you type a word with a space in it (ex. NEW JERSEY) the space will not appear in the word search but it will appear in the word list.

Editing a WordSearch

While editing a word search, the word you're currently dealing with is highlighted by INVERSE text. There are a variety of commands available that affect this word. They are:

I J K M - These move the word in the usual ESCape directions.
< - > - The left and right arrow keys rotate the word counter clockwise and clockwise respectively.
N L - These keys place the word in the search where it is and begins editing the (N) next or (L) last word. **Note:** Words are edited according to length, longest word first, shortest word last.
G - The "G" key grabs a word (for editing) specified by you.

By pressing the "**P**" key, you enter the printout subroutine. The first prompt of this routine is "ANSWER SHEET OR WORD SEARCH?". Pressing "A" will cause a printout of the answer sheet. Pressing "W" will cause a printout of the actual word search (with random letters placed where no words are hidden). The next prompt is "SINGLE SPACED OR DOUBLE SPACED?". If you choose single spaced, the wordsearch rows will be printed one after another. On the other hand, if you choose double spaced, a blank line will be inserted between each row.

Next, the printout routine asks you the title. This is followed by the prompt "READY THE PRINTER." You must now adjust the printer and then press a key to get the printout.

The "**A**" key is meant for adding words to the word list.

Pressing "**E**" will invoke the edit words routine. When prompted for a word, you should type in the word you wish to change or delete. If you hit RETURN without typing a word, the alphabetical list of words will be displayed with duplicates highlighted. You may pause the list by pressing CTRL-S and you may terminate the listing by pressing RETURN.

The "**ESC**" key is used primarily to exit the program. It also allows two other options. You may go back to the wordsearch which will act just like you didn't press ESC or you may start the program over losing any unsaved work.

Pressing "**D**" or answering "Y" to the third prompt will invoke the disk access subroutine. This routine will let you do just about any DOS command with slot, drive and volume parameters. In addition, the LOAD and SAVE commands are intercepted before they get to DOS. Instead of LOADing and SAVEing BASIC programs they now load and save a word search (with any filename you wish).

```
10 REM /\/\/\/\/\/\/\/\/\/\/\/\
20 REM \                      /
30 REM /     CORE PRESENTS    \
40 REM \                      /
50 REM /                      \
60 REM \     WORD SEARCHER    /
70 REM /                      \
80 REM \/\/\/\/\/\/\/\/\/\/\/\/
90 REM
100 GOSUB 1540: GOSUB 1030: GOTO 300
110 REM POKE IN COORDINATES
120 POKE 252,W%(Z,WO): POKE 253,
    W%(O1,WO): RETURN
130 REM ERASE CURRENT WORD
140 GOSUB 120: POKE Z,64: CALL ML:
    RETURN
150 REM POKE IN STATS OF CURRENT
    WORD
160 FOR A = O1 TO LEN (W$(Z,WO)):
    POKE 735 + A, ASC (
    MID$(W$(Z,WO) + CHR$ (Z),A))
170 NEXT : POKE 249, LEN (W$(Z,WO)):
    POKE O1,W%(TW,WO): RETURN
180 REM MOVE IN A DIRECTION
190 GOSUB 140:XS = Z:YS = Z: ON A
    GOTO 200,210,220,230
200 YS = - O1: GOTO 240
210 XS = - O1: GOTO 240
220 XS = O1: GOTO 240
230 YS = O1
240 W%(Z,WO) = W%(Z,WO) + XS: IF
    W%(Z,WO) > = GX THEN W%(Z,WO) =
    LX
250 IF W%(Z,WO) < LX THEN W%(Z,WO) =
    GX - O1
260 W%(O1,WO) = W%(O1,WO) + YS: IF
    W%(O1,WO) > = GY THEN W%(O1,WO)
    = LY
270 IF W%(O1,WO) < LY THEN W%(O1,WO)
    = GY - O1
280 GOTO 320
```

```
290 REM MANIPULATE WORD SEARCH
300 GOSUB 1690: HOME :XS =
    W%(Z,01):W%(Z,01) = Z: GOSUB
    420:W%(Z,01) = XS:WO = 01
310 GOSUB 160: IF W%(Z,WO) = Z THEN
    W%(Z,WO) = 19:W%(01,WO) = 11
320 GOSUB 1220
330 WAIT - 16384,128: GET A$
340 FOR A = 01 TO LEN (K$): IF A$ < >
    MID$ (K$,A,01) THEN NEXT : GOTO
    330
350 ON A GOTO
    190,190,190,190,530,550,1350,4
    70,450,840,800,390,370,1710
360 REM ROTATE WORD
370 GOSUB 140:W%(TW,WO) = W%(TW,WO)
    + 01: IF W%(TW,WO) > 7 THEN
    W%(TW,WO) = Z
380 GOTO 400
390 GOSUB 140:W%(TW,WO) = W%(TW,WO)
    - 01: IF W%(TW,WO) < Z THEN
    W%(TW,WO) = 7
400 POKE 01,W%(TW,WO): GOTO 320
410 REM PRINT WORD SEARCH
420 CALL 903: FOR A = 01 TO NW: IF
    W%(Z,A) = Z THEN NEXT : RETURN
430 WO = A: GOSUB 160: POKE Z,96:
    GOSUB 120: CALL ML:A = WO: NEXT
    : RETURN
440 REM TRY FOR NEXT WORD
450 XS = 01: GOTO 480
460 REM TRY LAST WORD
470 XS = - 01
480 GOSUB 1250:WO = WO + XS: IF WO >
    NW THEN WO = 01
490 IF WO < 01 THEN WO = NW
500 GOSUB 160: IF W%(Z,WO) = Z THEN
    W%(Z,WO) = 19:W%(01,WO) = 11:
    GOTO 320
510 XS = W%(Z,WO):YS = WO:W%(Z,WO) =
    Z: GOSUB 420:WO = YS:W%(Z,WO) =
    XS: GOSUB 160: GOTO 320
520 REM ADD WORDS
530 GOSUB 1030: GOTO 300
540 REM PRINTER PRINTOUT
550 GOSUB 1250: GOSUB 1300
560 HOME : PRINT TAB( 15)"PRINT
    OUT": VTAB 4
570 INVERSE : PRINT "A";: NORMAL :
    PRINT "NSWER SHEET OR ";
580 INVERSE : PRINT "W";: NORMAL :
    PRINT "ORD SEARCH?":XS = Z
590 VTAB 23: PRINT "TO LEAVE PRESS
    ";: INVERSE : PRINT "ESC";:
    NORMAL : PRINT "APE"
600 WAIT - 16384,128: GET A$: IF A$
    < > "W" AND A$ < > "A" THEN 300
610 VTAB 4: CALL 64578: PRINT
    "ANSWER SHEET":YS = Z: IF A$ =
    "W" THEN XS = 01: VTAB 4: PRINT
    "WORD SEARCH "
620 VTAB 7: INVERSE : PRINT "S";:
    NORMAL : PRINT "INGLE SPACED OR
    ";
630 INVERSE : PRINT "D";: NORMAL :
    PRINT "OUBLE SPACED?"
640 WAIT - 16384,128: GET A$: IF A$
    < > "S" AND A$ < > "D" THEN 560
650 VTAB 7: CALL 64578: PRINT
    "SINGLE SPACED": IF A$ = "D"
    THEN YS = 01: VTAB 7: PRINT
    "DOUBLE SPACED"
660 VTAB 12: CALL 64578: PRINT:
    INPUT "TITLE=>";T$: HOME
670 PRINT "READY THE PRINTER": GET
    A$:WO = 01: IF X * TW < = PX THEN
    W=TW
680 PR# PR: REM PRINTER SETUP
690 IF T$ < > "" THEN PRINT SPC( (PX
    - LEN (T$)) / TW)T$: PRINT M$;
700 FOR A = 01 TO Y: IF YS THEN PRINT
    M$;
710 PRINT SPC( (PX - X * WO) / TW);:
    FOR B = 01 TO X: IF XS = Z THEN
    730
720 IF MID$ (WS$(A),B,01) = "." THEN
    PRINT CHR$ ( RND (01) * 26 +
    65);: GOTO 740
730 PRINT MID$ (WS$(A),B,01);
740 IF WO = TW THEN PRINT " ";
750 NEXT : PRINT M$;: NEXT : PRINT
    M$;
760 FOR A = 01 TO NW STEP TW: PRINT
    SPC( PX / 4)A") "W$(01,A)" ";
770 IF NW > (A) THEN PRINT SPC( PX *
    5 / 8 - LEN (W$(01,A))- LEN (
    STR$ (A)) - TW - PX / 4)A + 01")
    "W$(01,A + 01)
780 NEXT : PRINT M$: PR# Z: GOTO 560
790 REM GRAB WORD
800 GOSUB 1250: POKE 34,23: VTAB 24:
    INPUT "GRAB WORD=>";W$: HOME :
    POKE 34,Z
810 FOR A = 01 TO NW: IF W$(Z,A)< >
    W$ THEN NEXT : PRINT G$;: GOSUB
    140: GOTO 320
820 WO = A: GOTO 500
830 REM EDIT WORDS
840 GOSUB 1250: HOME : PRINT SPC(
    4)"EDIT WORDS (TYPE 'X' TO
    RETURN)": POKE 34,01
850 PRINT : INPUT "WORD =>";W$:
    PRINT : IF W$ = "X" THEN TEXT :
    GOTO 300
860 IF NW = Z THEN TEXT : GOSUB
    1030: GOTO 300
870 IF VAL (W$) < > Z THEN W$ =
    W$(01, VAL (W$))
880 IF W$ = "" THEN 990
890 FOR A = 01 TO NW: IF W$(Z,A)< >
    W$ THEN NEXT : PRINT "?WORD NOT
    FOUND"G$: GOTO 850
900 XS = A: FOR A = 01 TO NW: IF
    W$(01,A) < > W$ THEN NEXT
910 YS = A: FOR A = YS + 01 TO
    NW:W$(01,A - 01) = W$(01,A):
    NEXT :W$(01,NW) = ""
920 FOR A = XS + 01 TO NW:W$(Z,A-
    01) = W$(Z,A):W%(Z,A - 01) =
    W%(Z,A):W%(01,A - 01) = W%(01,A)
930 W%(TW,A - 01) = W%(TW,A): NEXT
    :W$(Z,NW) = "":W%(TW,NW) =
    TW:W%(Z,NW) = Z:W%(01,NW) = Z
940 INVERSE : PRINT "C";: NORMAL :
    PRINT "HANGE OR ";: INVERSE :
    PRINT "D";: NORMAL : PRINT
    "ELETE"
950 WAIT - 16384,128: GET A$: IF A$
    < > "C" AND A$ < > "D" THEN 950
960 VTAB ( PEEK (37)): CALL 64578:
    IF A$ = "D" THEN NW = NW - 01:
    PRINT "DELETED": GOTO 850
970 INPUT "NEW WORD=>";W$: IF LEN
    (W$) > X OR LEN (W$) > Y OR LEN
    (W$) > 20 THEN 970
980 GOSUB 1140: GOTO 850
990 FOR A = 01 TO NW: IF W$(01,A) =
    W$(01,A - 01) THEN FLASH
1000 PRINT A") "W$(01,A): NORMAL :
    IF PEEK ( - 16384) < > 141 THEN
    NEXT
1010 POKE - 16368,Z: GOTO 850
1020 REM INPUT WORDS
1030 HOME : PRINT SPC( 5)"WORD ENTRY
    (TYPE 'X' TO EXIT)": VTAB 4
1040 IF NW THEN PRINT "ERASE WORDS
    IN MEMORY (Y/N) Y" CHR$ (8);:
    GET W$: IF W$ < > "N" THEN GOSUB
    1660:NW = Z
1050 VTAB TW: CALL 64578: VTAB 4:
    POKE 34,TW
1060 PRINT : INPUT "WORD =>";W$
1070 IF LEN (W$) > X OR LEN (W$) > Y
    OR LEN (W$) > 20 THEN PRINT
    G$"TOO LONG": GOTO 1060
1080 IF W$ = "X" THEN 1120
1090 IF W$ = "" THEN 1060
1100 NW = NW + 01:XS = Z: GOSUB 1140
1110 IF NW < 75 THEN 1060
1120 TEXT : RETURN
1130 REM FIND A PLACE FOR THE WORD
1140 FOR A = 01 TO NW: IF W$ >
    W$(01,A) THEN NEXT :W$(01,NW) =
    W$: GOTO 1160
1150 FOR B = NW TO A STEP -
    01:W$(01,B) = W$(01,B - 01):
    NEXT :W$(01,A) = W$
1160 A$ = "": FOR A = 01 TO LEN (W$):
    IF MID$ (W$,A,01) < > " " THEN
    A$ = A$ + MID$ (W$,A,01)
1170 NEXT : FOR A = 01 TO NW: IF LEN
    (A$) < LEN (W$(Z,A)) THEN NEXT
    :A = NW: GOTO 1200
1180 FOR B = NW TO A STEP -
    01:W$(Z,B) = W$(Z,B -
    01):W%(TW,B) = W%(TW,B - 01)
1190 W%(Z,B) = W%(Z,B - 01):W%(01,B)
    = W%(01,B - 01): NEXT
1200 W$(Z,A) = A$:W%(Z,A) =
    Z:W%(01,A) = Z:W%(TW,A) = TW:
    RETURN
1210 REM HIGHLIGHT THE CURRENT WORD
1220 GOSUB 120: POKE Z,Z: CALL ML:
    GOSUB 120
1230 POKE Z,96: INVERSE : CALL ML:
    NORMAL : RETURN
1240 REM DROP WORD
1250 IF PEEK (Z) = 255 THEN 1270
1260 POKE Z,128: CALL ML: IF PEEK
    (Z) = 255 THEN 1280
1270 PRINT G$;: POP : GOTO 330
1280 POKE Z,96: GOSUB 120: CALL ML:
    RETURN
1290 REM CONVERT SCREEN IMAGE
1300 VTAB 24: HTAB 01: PRINT "ONE
    MOMENT PLEASE";
1310 POKE 01,TW: POKE 249,X: FOR A =
    01 TO Y: POKE 252,LX
1320 POKE 253,LY + A - 01: POKE Z,Z:
    CALL ML:WS$(A) = ""
1330 FOR B = 01 TO X:WS$(A) = WS$(A)
    + CHR$ ( PEEK (703 + B)): NEXT :
    NEXT : RETURN
1340 REM DISK ACCESS
1350 HOME : PRINT SPC( 5)"DISK
    ACCESS (TYPE 'X' TO EXIT)": POKE
    34,01: VTAB 3
1360 PRINT : INPUT "COMMAND=>";A$:
    IF PEEK (512 + LEN (A$)) THEN
    VTAB PEEK (37): CALL 64578
1370 W$ = "": FOR A = 512 + LEN (A$)
    TO 767: IF PEEK (A) THEN W$ = W$
    + CHR$ ( PEEK (A)): NEXT
1380 T$ = CHR$ (4): ONERR GOTO 1520
1390 IF LEFT$ (A$,4) < > "SAVE" THEN
    1440
1400 A$ = RIGHT$ (A$, LEN (A$) - 4):
    PRINT T$"OPEN"A$W$: PRINT
    T$"DELETE"A$
1410 PRINT T$"OPEN"A$: PRINT
    T$"WRITE"A$: PRINT NW","X","Y
1420 FOR A = 01 TO NW: FOR B = Z TO
    TW: PRINT CHR$ (63 + W%(B,A));
1430 NEXT : PRINT
```

```
    ","W$(Z,A)","W$(O1,A): NEXT :
    GOTO 1480
1440 IF LEFT$ (A$,4) < > "LOAD" THEN
    1490
1450 GOSUB 1670:A$ = RIGHT$ (A$, LEN
    (A$) - 4): PRINT T$"VERIFY"A$W$:
    PRINT T$"OPEN"A$
1460 PRINT T$"READ"A$: INPUT NW,X,Y:
    FOR A = O1 TO NW: INPUT
    A$,W$(Z,A),W$(O1,A)
1470 FOR B = Z TO TW:W%(B,A) = ASC (
    MID$ (A$,B + O1)) - 63: NEXT :
    NEXT : GOSUB 1680
1480 PRINT T$"CLOSE": GOTO 1360
1490 IF LEFT$ (A$,O1) = "B" THEN
    POKE 222,11: GOTO 1520
1500 IF A$ = "X" THEN POKE 216,Z:
    TEXT : GOTO 300
1510 PRINT T$A$W$: GOTO 1360
1520 CALL 932: PRINT : PRINT "ERROR
    #" PEEK (222)G$: GOTO 1360
1530 REM INITIALIZE VARIABLES
1540 A = O1 = X = Y = Z = TW = LX =
    LY = GX = GY = NW = B = PR = PX =
    WO = ML = XS = YS
1550 A$ = "":K$ = "IJKMAPDLNEG" +
    CHR$ (8) + CHR$ (21) + CHR$
    (27):G$ = CHR$ (7):M$ = CHR$
    (13)
1560 DIM W$(1,75),WS$(24),W%(2,75)
1570 O1 = 1:Z = O:TW = 2:WO = O1:ML =
    768
1580 IF PEEK (768) < > 6 THEN PRINT
    CHR$ (4)"BLOAD WORD
    SEARCHER.OBJ,A$300"
1590 WS$(Z) = ".": FOR A = O1 TO
    5:WS$(Z) = WS$(Z) + WS$(Z): NEXT
1600 TEXT : HOME : NORMAL : PRINT
    SPC( 5)"CORE WORD SEARCH
    EDITOR/CREATOR": VTAB 4
1610 PRINT : INPUT "PRINTER SLOT
    =>";PR: IF PR > 7 OR PR < Z THEN
    1610
1620 PR = INT (PR): PRINT : INPUT
    "CHARACTERS PER LINE =>";PX: IF
    PX < 40 OR PX > 255 THEN 1620
1630 PRINT : PRINT "DISK ACCESS?
    (Y/N) N" CHR$ (8);: GET A$: VTAB
    PEEK (37): CALL 64578: IF A$ =
    "Y" THEN POP : GOTO 1350
1640 PRINT : INPUT "COLUMNS IN
    SEARCHWORD =>";X: IF X > 38 OR X
    < 5 THEN 1640
1650 PRINT : INPUT "ROWS IN
    SEARCHWORD =>";Y: IF Y > 23 OR Y
    < 5 THEN 1650
1660 X = INT (X):Y = INT (Y): FOR A =
    O1 TO Y:WS$(A) = LEFT$
    (WS$(Z),X): NEXT
1670 FOR A = O1 TO NW:W$(O1,A) =
    "":W$(Z,A) = "":W%(Z,A) =
    Z:W%(O1,A) = Z:W%(TW,A) = TW:
    NEXT
1680 LX = 20 - INT (X / TW):GX = LX +
    X:LY = 11 - INT (Y / TW):GY = LY
    + Y
1690 POKE 254,LX: POKE 255,LY: POKE
    250,GX: POKE 251,GY: RETURN
1700 REM EXIT?
1710 VTAB 24: HTAB O1: INVERSE :
    PRINT "E";: NORMAL : PRINT "XIT
    ";: INVERSE : PRINT "C";: NORMAL
1720 PRINT "LEAR WORK AREA ";:
    INVERSE : PRINT "B";: NORMAL :
    PRINT "ACK TO SEARCHWORD";
1730 WAIT - 16384,128: GET A$: IF A$
    < > "E" AND A$ < > "C" AND A$ < >
    "B" THEN 1730
1740 HTAB O1: VTAB 24: CALL 64578:
    IF A$ = "C" THEN RUN
1750 IF A$ = "B" THEN 330
1760 REM HBAYR RRAADY YDAARR RYABH
1770 TEXT : HOME : END
```

Word Searcher Example

```
P X S G G S N T B X H I
O A I D W I J R O J N U
A N I R E L B B I N T N
O S R E T E M A R A P B
K O E L S P C K E L Y H
X F I L E U A P P S R Z
K T P O C D H K U J C T
I K O R T K P Y S A N H
C E C T O O C V R Z E B
F Y T N R J T A M R O F
T F I O M G S I R Y A J
N R B C P P F N X T Q B
```

1) A P T
2) BIT COPIER
3) CONTROLLER
4) DISK
5) ENCRYPT
6) FORMAT
7) NIBBLE
8) PARAMETERS
9) SECTOR
10) SOFTKEY
11) SUPER IOB
12) TRACK

Word Searcher Hexdump

```
0300:06 00 A2 FF B0 6B E8 E4  $2C33
0308:F9 90 01 60 A5 FD 20 C1  $44EE
0310:FB A4 FC 20 4E 03 A4 01  $8A5A
0318:18 B9 44 03 65 FD 85 FD  $1037
0320:C5 FB B0 16 C5 FF 90 12  $E833
0328:18 C8 C8 B9 44 03 65 FC  $AE47
0330:85 FC C5 FA B0 04 C5 FE  $2800
0338:B0 CC E8 E4 F9 B0 04 A9  $C7F9
0340:FF 85 00 60 FF FF 00 01  $8E34
0348:01 01 00 FF FF FF 24 00  $E824

0350:30 0E B1 28 29 7F 70 04  $0F64
0358:9D C0 02 60 9D E0 02 60  $DBB0
0360:70 05 BD C0 02 50 03 BD  $71E4
0368:E0 02 09 80 25 32 91 28  $CFC5
0370:60 86 00 E8 E4 F9 B0 F8  $18D0
0378:BD C0 02 C9 2E F0 F4 DD  $9EC5
0380:E0 02 F0 EF 86 00 60 A5  $478D
0388:FA 85 21 A5 FF 48 20 C1  $C2D3
0390:FB A9 AE A4 FE 20 A0 FC  $C2C6
0398:68 69 00 C5 FB 90 EE A9  $C58D

03A0:28 85 21 60 68 A8 68 A6  $4407
03A8:DF 9A 48 98 48 60        $6D45
```

Source Code

```
1000* ------------------------------
1010 * WORD SEARCHER
1020 *  MACHINE LANGUAGE STUFF
1030 * ------------------------------
1040
1050 LX          .EQ $FE          LEAST MOST X
COORDINATE AVAILABLE
1060 LY          .EQ $FF          LEAST MOST Y
COORDINATE AVAILABLE
1070 GX          .EQ $FA          GREATEST XCOO
RDINATE AVAILABLE
1080 GY          .EQ $FB          GREATEST YCOO
RDINATE AVAILABLE
1090 X           .EQ $FC          STARTING AND
CURRENT X COORDINATE
1100 Y           .EQ $FD          STARTING AND
CURRENT Y COORDINATE
1110 DIR         .EQ 1            DIRECTION TO
LOAD/SAVE
1120 DUR         .EQ $F9          NUMBER OF TIM
ES TO MOVE
1130 FUNC        .EQ $0           FUNCTION TO P
ERFORM, B6=LOAD/SAVE, B5=PRIMARY/SECONDA
RY BUFFER, B7=VERIFY?
1140 INVFLG      .EQ 50           FLAG THAT TEL
LS WHETHER TEST IS INVERSE OR NOT
1150 BASCALC     .EQ $FBC1        ROUTINE THAT
CALCULATES TEXT SCREEN ADDRESS
1160 BASL        .EQ $28          WHERE THE SCR
EEN ADDRESS IS STORED
1170 PRIMARY     .EQ $2C0         PRIMARY BUFFE
R
1180 SECONDARY   .EQ $2E0         SECONDARY BUF
FER
1190 CLEOL2      .EQ $FCA0        ROUTINE THAT
STORES A ROW OF PERIODS
1200
1210             .OR $300
1220             .TF WORD         SEARCHER.OBJ
1230
1240             ASL FUNC         VERIFY?
1250             LDX #$FF         BEG. OFFSET
1260             BCS VERIFY       YES
1270
1280 NXT1        INX              STARTING1
1290             CPX DUR          DONE?
1300             BCC DO1          NOPE!
1310             RTS
1320
1330 DO1         LDA Y
1340             JSR BASCALC      GET ADDR
1350             LDY X            CH
1360             JSR MOVER        LD/SV PR/SC
1370
1380             LDY DIR          INC/DEC
1390             CLC              C=0!
1400             LDA SPEEDS,Y     Y=Y+YS
1410             ADC Y
1420             STA Y
1430             CMP GY
1440             BCS RTS2         STILL IN?
1450             CMP LY
1460             BCC RTS2
1470             CLC              C=0!
1480             INY              XS OFFSET
1490             INY
1500             LDA SPEEDS,Y     X=X+XS
1510             ADC X
1520             STA X
1530             CMP GX           STILL IN?
1540             BCS RTS2
1550             CMP LX
1560             BCS NXT1
1570 RTS2        INX
1580             CPX DUR          FINISHED
1590             BCS RTS1         YES, IGNORE
1600             LDA #$FF
1610             STA FUNC         TELL BASIC
1620 RTS1        RTS
1630
1640 SPEEDS      .HS FFFF0001010100FFFFFF
1650
1660 MOVER       BIT FUNC         LD OR SV?
1670             BMI LOADER       LOAD!
1680
1690             LDA (BASL),Y     SAVE SCRN
1700             AND #$7F         ASCII CNVRT
1710             BVS SECND.S
1720             STA PRIMARY,X
1730             RTS
1740 SECND.S     STA SECONDARY,X
1750             RTS
1760
1770 LOADER      BVS SECND.L
1780             LDA PRIMARY,X
1790             BVC LOAD
1800 SECND.L     LDA SECONDARY,X
1810 LOAD        ORA #$80         NORMALIZE
1820             AND INVFLG       INVERSE?
1830             STA (BASL),Y
1840 RTS3        RTS
1850
1860 VERIFY      STX FUNC         RETURN CODE
1870 .1          INX              NEXT1
1880             CPX DUR          DONE?
1890             BCS RTS3         YES!
1900             LDA PRIMARY,X    GET ONE
```

Continued on page 30

Continued from page 14

```
60 LOMEM: 8448: HIMEM: 9983: GOTO
    10010
70 REM INITIAL IOB SETUP
80 POKE BUF,39: POKE DRV,DV: POKE
    VOL,VL: POKE SLT,SO * 16: RETURN
90 REM R/W SECTOR
100 BF = 0: POKE TRK,TK: POKE
    SCT,ST: POKE CMD,CD: CALL IO:
    POKE BUF, PEEK (BUF) + 1: IF
    PEEK (BUF) = > MB THEN BF = 1
110 RETURN
120 REM MOVE S PHASES
130 POKE 49289 + SO * 16 + DV,0:
    POKE 49289 + SO * 16,0: A = PH -
    INT (PH / 4) * 4: POKE 1144,128
    + A: POKE 811,128 + S + A: POKE
    813,SO * 16: CALL 810: POKE
    49288 + SO * 16,0: PH = PH + S:
    IF PH < 0 THEN PH = 0
140 RETURN
150 REM 16 SECTOR RWTS ALTERATIONS
160 REM IGNORE CHKSUM & END MARKS
170 POKE 47405,24: POKE 47406,96:
    POKE 47497,24: POKE 47498,96:
    RETURN
180 REM ALTERED ADDRESS MARKS
190 READ A1,A2,A3: POKE 47445,A1:
    POKE 47455,A2: POKE 47466,A3:
    RETURN
200 REM ALTERED DATA MARKS
210 READ A1,A2,A3: POKE 47335,A1:
    POKE 47345,A2: POKE 47356,A3:
    RETURN
220 REM NORMALIZER
230 POKE 47405,208: POKE 47406,19:
    POKE 47497,208: POKE 47498,183:
    POKE 47445,213
240 POKE 47455,170: POKE 47466,150:
    POKE 47335,213: POKE 47345,170:
    POKE 47356,173: RETURN
250 REM 13 SECTOR RWTS ALTERATIONS
260 REM IGNORE CHKSUM & END MARKS
270 POKE 47530,24: POKE 47531,96:
    POKE 47438,24: POKE 47439,96:
    RETURN
280 REM ALTERED ADDRESS MARKS
290 READ A1,A2,A3: POKE 47478,A1:
    POKE 47488,A2: POKE 47499,A3:
    RETURN
300 REM ALTERED DATA MARKS
310 READ A1,A2,A3: POKE 47368,A1:
    POKE 47378,A2: POKE 47389,A3:
    RETURN
320 REM NORMALIZER
330 POKE 47530,208: POKE 47531,183:
    POKE 47438,208: POKE 47439,19:
    POKE 47478,213
340 POKE 47488,170: POKE 47499,181:
    POKE 47368,213: POKE 47378,170:
    POKE 47389,173: RETURN
350 REM SWAP RWTS AT $1900 WITH THE
    ONE AT $B800
360 POKE 253,25: POKE 255,184: POKE
    224,8: CALL 832: RETURN
370 REM FORMAT DISK
380 A$ = "VOLUME NUMBER FOR
    COPY=>254": HOME: GOSUB 450:
    HTAB32: INPUT "";VL$: VL = VAL
    (VL$): IF VL$ = "" THEN VL = 254
390 IF VL > 255 OR VL < 0 THEN 380
400 POKE CMD,INIT: SO = S2: DV = D2:
    A$ = "INSERT BLANK DISK IN SLOT
    " + STR$ (S2) + ", DRIVE " + STR$
    (D2): GOSUB 470
410 GOSUB 80: HOME: A$ =
    "FORMATING": FLASH: GOSUB 450:
    NORMAL: CALL IO: VL = 0: RETURN
420 REM PRINT TRACK & SECTOR#
430 VTAB 3: HTAB 10: PRINT
    "TRACK=>"TK SPC( 2)"SECTOR=>"ST
    SPC(2): RETURN
440 REM CENTER MESSAGE
450 HTAB 21 - LEN (A$) / 2:
    PRINTA$;: RETURN
460 REM PRINT MESSAGE AND WAIT
470 HOME: VTAB 11: GOSUB 450:
    VTAB13: A$ = "PRESS ANY KEY TO
    CONTINUE": GOSUB 450: WAIT
    -16384,128: GET A$: RETURN
480 REM TOGGLE READ/WRITE
490 CD = (CD = 1) + 1: IF CD = RD
    THEN A$ = "INSERT SOURCE DISK.":
    SO = S1: DV = D1: GOTO 510
500 A$ = "INSERT TARGET DISK.": SO =
    S2: DV = D2
510 IF D1 = D2 AND S1 = S2 THEN GOSUB
    470: HOME
520 VTAB 1: HTAB 1: PRINT SPC(39);:
    FLASH: A$ = "READING": IF CD =
    WR THEN A$ = "WRITING"
530 GOSUB 450: NORMAL: GOTO 80
540 REM ONERR IGNORE UNREADABLE
    SECTORS
550 CALL 822:ERR = PEEK (222): IF
    ERR > 15 AND ERR < 254 THEN POKE
    216,0
560 IF (ERR = 255 OR ERR = 254) AND
    CD < > RD THEN 10230
570 IF ERR > 15 THEN RESUME
580 PRINT CHR$ (7);: POKE BUF, PEEK
    (BUF) + 1: IF PEEK (BUF) = > MB
    THEN BF = 1
590 RETURN
10000 REM CONFIGURATION TIME
10010 REM BLOAD RWTS HERE
10020 IF PEEK (768) * PEEK (769) =
    507 THEN 10060
10030 HOME: A$ = "* SUPER IOB *":
    GOSUB 450: PRINT: PRINT: A$ =
    "CREATED BY RAY DARRAH": GOSUB
    450
10040 VTAB 10: A$ = "INSERT SUPER
    IOB DISK": GOSUB 450: PRINT:
    PRINT: PRINT: A$ = "PRESS ANY
    KEY TO CONTINUE": GOSUB450: WAIT
    - 16384,128: GET A$
10050 PRINT: PRINT CHR$ (4)"BLOAD
    IOB.OBJ0,A$300"
10060 TK = ST = VL = CD = DV = SO:RD
    = 1:WR = 2:INIT = 4: ONERR GOTO
    10220
10070 IO = 768: SLT = 779: DRV =
    780: VOL = 781: TRK = 782: SCT
    =783: BUF = 787: CMD = 790: OVL
    =792
10080 HOME: DOS = 16:MB = 151
10090 VTAB 8: PRINT: A$ =
    "ORIGINAL": S2 = 6: D2 = 1:
    GOSUB 10140: S1 = S2: D1 = D2
10100 PRINT: PRINT: PRINT: D2 =(D2 =
    1) + 1: A$ = "DUPLICATE
": GOSUB 10140
10110 A$ = "FORMAT BACK UP FIRST? N"
    + CHR$ (8): HOME: VTAB12: GOSUB
450: GET A$: IF A$ = "Y" THEN GOSUB
    380
10120 HOME: A$ = "INSERT DISKS IN
    PROPER DRIVES.": GOSUB 470:
    HOME: GOTO 1000
10130 REM GET SLOT AND DRIVE#
10140 GOSUB 450: PRINT: PRINT: PRINT
    TAB( 10)"SLOT=>"S2
    SPC(8)"DRIVE=>"D2;
10150 HTAB 16: B$ = "7": GOSUB
    10180: S2 = VAL (A$)
10160 HTAB 32: B$ = "2": GOSUB
    10180: D2 = VAL (A$): RETURN
10170 REM GET A KEY
10180 GET A$: IF (A$ < "1" OR A$ >
    B$) AND A$ < > CHR$ (13) THEN
    10180
10190 IF A$ = CHR$ (13) THEN A$ =
    CHR$ ( PEEK ( PEEK (40) + PEEK
    (41) * 256 + PEEK (36)) - 128)
10200 PRINT A$;: RETURN
10210 REM DISK ERROR
10220 ERR = PEEK (222): IF ERR >15
    AND ERR < 254 THEN POKE216,0:
    CALL 822
10230 IF ERR = 254 THEN PRINT "TYPE
    AGAIN PLEASE:": PRINT: RESUME
10240 IF ERR = 255 THEN STOP
10250 IF ERR = 0 THEN A$ =
    "INITIALIZATION ERROR"
10260 IF ERR = 1 THEN A$ = "WRITE
    PROTECTED"
10270 IF ERR = 2 THEN A$ = "VOLUME
    MISMATCH ERROR"
10280 IF ERR = 4 THEN A$ = "DRIVE
    ERROR"
10290 IF ERR = 8 THEN A$ = "READ
    ERROR"
10300 VTAB 20: GOSUB 450: PRINT CHR$
    (7): END
62000 REM DATA FOR MARKS
```

CHECKSUMS

```
10    -  $BADD      410   -  $9A03
20    -  $9B13      420   -  $FF36
30    -  $4D3B      430   -  $713A
40    -  $AD92      440   -  $0A35
50    -  $C899      450   -  $76B5
60    -  $1FBA      460   -  $51E2
70    -  $0061      470   -  $CCA2
80    -  $835F      480   -  $7AD0
90    -  $E171      490   -  $EEB8
100   -  $ADOE      500   -  $3A54
110   -  $57B6      510   -  $5FC8
120   -  $8472      520   -  $D7BE
130   -  $617E      530   -  $A4CF
140   -  $0F1F      540   -  $1447
150   -  $F1B3      550   -  $0523
160   -  $C59A      560   -  $DD07
170   -  $6DEC      570   -  $EAC1
180   -  $56EA      580   -  $AB8E
190   -  $D2AC      590   -  $62FA
200   -  $1EEF    10000   -  $ADC8

210   -  $C7D5    10010   -  $074E
220   -  $7B7E    10020   -  $E36A
230   -  $F7E4    10030   -  $A272
240   -  $596A    10040   -  $DE1E
250   -  $50B9    10050   -  $32F3
260   -  $7DB0    10060   -  $58BE
270   -  $AD47    10070   -  $2C16
280   -  $E373    10080   -  $0328
290   -  $4B8B    10090   -  $E82F
300   -  $FFE7    10100   -  $6FD0
310   -  $4DD1    10110   -  $60F3
320   -  $4DA3    10120   -  $9CCC
330   -  $C76F    10130   -  $BBE7
340   -  $01F0    10140   -  $29D9
350   -  $F0AE    10150   -  $1F41
360   -  $5452    10160   -  $81FC
370   -  $C2A5    10170   -  $03AA
380   -  $8A57    10180   -  $4592
390   -  $65AE    10190   -  $AF74
400   -  $15FA    10200   -  $B55E
```

```
10210 -   $E996  10260 -   $120B
10220 -   $D7C9  10270 -   $E779
10230 -   $6FB9  10280 -   $A46B
10240 -   $7772  10290 -   $57BD
10250 -   $C5EC  10300 -   $636F
                 62000 -   $2462
```

Source Code

```
1000 * ---------------------------------
1010 * Super IOB machine routines
1020 *
1030 * BY RAY DARRAH
1040 * ---------------------------------
1050
1060 RWTS.B800   .EQ $03D9          ENTRY POINT T
O RWTS @$B800
1070 INVOKERROR  .EQ $D412          ROUTINE THAT
CAUSES BASIC TO DO THE ERROR CONTAINED IN X
1080 RWTS.1900   .EQ $1E00          ENTRY POINT T
O THE RWTS AT $1900
1090 SEEKABS     .EQ $B9A0          ENTRY POINT T
O THE SEEKABS ROUTINE AT $B800
1100 BAS.ERR     .EQ 222            ;BASIC ON ERR
 ERROR CODE
1110 SWFRM       .EQ $FC            ;EXCHANGE FR
OM PARAMTER
1120 SWTO        .EQ $FE            ;EXCHANGE RW
TS 'TO' PARAMETER
1130 PAGES       .EQ $E0            ;NUMBER OF PA
GES OF MEMORY TO EXCHANGE
1140             .OR $0300          STARTS AT PAG
E THREE
1150             .TF  IOB.OBJ0
1160
1170 * ---------------------------------
1180 *              CALL RWTS          *
1190 * ---------------------------------
1200
1210 IO          LDA /TABLETYP      ENTRY POINT
FOR CALING THE RWTS THROUGH BASIC
1220             LDY #TABLETYP      A,Y POINT TO
 THE IOB TABLE
1230             JSR RWTS.B800      GO TO THE RW
TS AT $B800
1240             BCS DOS.ERR        IF THE CARRY
SET THEN CAUSE BASIC ERROR
1250             RTS                OTHERWISE, AL
L IS WELL SO RETURN
1260 TABLETYP    .HS 01             TYPE OF TABLE
 (1=IOB)
1270 SLT         .HS 60             SLOT
1280 DRV         .HS 01             DRIVE
1290 VOL         .HS 00             VOLUME
1300 TRK         .HS 00             TRACK
1310 SCT         .HS 00             SECTOR
1320 DCTPTR      .DA DCT            POINTER TO TH
E DEVICE CHARACTERISTICS TABLE
1330 BUFFERLO    .HS 00             ALWAYS MAKE
LSB OF BUFFER POINTER ZERO!
1340 BUF         .HS 27             SECTOR BUFFE
R PAGE POINTER
1350 NOTHING     .HS 00             NOT USED
1360 BYTCOUNT    .HS 00             BYTE COUNT FO
R PARTIAL SECTOR (0=256 BYTES)
1370 CMD         .HS 00             COMMAND COD
E (0=SEEK)
1380 RWTS.ERR    .HS 00             ERROR CODE T
HA THE RWTS.B800 RETURNS WITH
1390 OVL         .HS 00             VOLUME NUMB
ER OF LAST ACCESSED DISK
1400 OLDSLT      .HS 60             SLOT PREVIOUS
LY ACCESSED
1410 OLDDRV      .HS 01             DRIVE PREVIOU
LSY ACCESSED
1420 DCT         .HS 00             DEVICE TYPE O
F DEVICE CHARACTERISTICS TABLE
1430 PHASES      .HS 01             PHASES-1 PER
TRACK, (0 OR 1)
1440 MOTORCNT    .HS EFD8           MOTOR-ON TIM
E COUNT
1450 DOS.ERR     LDA RWTS.ERR       DOS HAS HAD A
N ERROR, GET THE ERROR CODE
1460             LSR                DIVIDE IT BY 16
1470             LSR
1480             LSR
1490             LSR
1500             TAX                TRANSFER IT T
O X SO BASIC WLL INDUCE THE FALSE ERROR CODE
1510             JMP INVOKERROR     CAUSE A BAS
IC ERROR
1520
1530 * ---------------------------------
1540 *           MOVE THE DISK ARM      *
1550 * ---------------------------------
1560
1570 MOVPHASES   LDA #$00           ROUTINE TO SE
T UP THE REGISTERS BEFORE CALLING SEEKABS
1580             LDX #$00           X AND A HAVE
DUMMY NUMBERS THAT WILL BE POKED INTO BY
 "MOVE S PHASES"
1590             JMP SEEKABS
1600
1610 * ---------------------------------
1620 *       CAUSE ERROR IN CONTROLLER  *
1630 * ---------------------------------
1640
1650 BASICERR    LDX BAS.ERR        BASIC HAS MAD
E AN ERROR SO CAUSE THE ERROR NUMBER AT 222
1660             JMP INVOKERROR
1670 * ---------------------------------
1680 *            POP OFF RETURN       *
1690 * ---------------------------------
1700 POP         PLA                ROUTINE TO PO
P OFF ONE RETURN (BASIC) ADDRESS
1710             TAY
1720             PLA
1730             LDX BAS.ERR+1      GET WHAT THE
 STACK WOULD BE IF THE GOSUB WASN'T THERE
1740             TXS                PUT THAT AS
THE STACK POINTER
1750             PHA
1760             TYA                RESTORE THE
LAST RETURN ADDRESS
1770             PHA
1780             RTS
1790
1800 * ---------------------------------
1810 *          EXCHANGE RWTS's         *
1820 * ---------------------------------
1830
1840             LDY #0             ;ZERO LSB's
1850             STY SWFRM          ;AND HAVE Y
AT ZERO FOR START
1860             STY SWTO
1870 MOVE.PAGE   LDA (SWFRM),Y      ;GET A BYTE
1880             PHA                ;AND SAVE IT
1890             LDA (SWTO),Y       ;GET THE BYTE
 WHERE THE SAVED ONE GOES
1900             STA (SWFRM),Y      ;AND STORE I
T WHERE THE SAVED ONE WAS
1910             PLA                ;GET THE SAVE
D BYTE
1920             STA (SWTO),Y       ;AND STORE IT
 WHERE IT GOES
1930             INY                ;DONE WITH A
PAGE
1940             BNE MOVE.PAGE      ;NO KEEP WOR
KING ON IT
1950             INC SWFRM+1        ;GET NEXT MSB
1960             INC SWTO+1
1970             DEC PAGES          ;DECREMENT T
HE NUMBER OF PAGES TO MOVE
1980             BNE MOVE.PAGE      ;IF NOT DONE
, MOVE ANOTHER PAGE
1990             RTS                ;FINISHED, RTS
```

Super IOB HEXDUMP

```
0300: A9 03 A0 0A 20 D9 03 B0  $BD35
0308: 16 60 01 60 01 00 00 00  $9CF5
0310: 1B 03 00 27 00 00 00 00  $4320
0318: 00 60 01 00 01 EF D8 AD  $55A7
0320: 17 03 4A 4A 4A 4A AA 4C  $B42B
0328: 12 D4 A9 00 A2 00 4C A0  $8038
0330: B9 A6 DE 4C 12 D4 68 A8  $6E1C
0338: 68 A6 DF 9A 48 98 48 60  $FDD9
0340: A0 00 84 FC 84 FE B1 FC  $3777
0348: 48 B1 FE 91 FC 68 91 FE  $AAB9

0350: C8 D0 F3 E6 FD E6 FF C6  $921F
0358: E0 D0 EB 60              $3160
```

Softkey For Sierra On-Line Software

By Doni G. Grande & Clay Harrell

Screenwriter II V2.2 ($129.95)
The Dic*tion*ary ($179.95 for the pair)
Sammy Lightfoot ($29.95)
Time Zone V1.1 ($100.00)
Apple Cider Spider ($29.95)
Oils Well ($29.95)
Cannonball Blitz ($29.95)

Sierra On-line, Inc.
36575 Mudge Ranch Road
Coarsegold, CA 93614

Requirements:
Apple with 48K
COPYA
A Sector editing program such as Disk Zap
A blank disk for each of the above programs

The folks at Sierra On-line have come to a very realistic view on copy protection. Some of their earlier releases such as Lunar Leepers used elaborate protection schemes like spiral tracking (uses ¼ tracks spiraled along the disk). The problem with this kind of protection is that it is expensive, and it doesn't work on all flavors of Apples (//e, etc.). On-Line probably also had to send out to have the copies made because you can't exactly run COPYA to make thousands of commercial copies of a spiral disk.

With competition increasing, and the high awareness to cut internal costs, On-line has chosen not to use elaborate protection schemes on their newer releases (Oil's Well, Sammy Lightfoot, Screenwriter II, ver 2.2, etc.). Instead, they use a good basic scheme that will often deter those equipped with Locksmith or Nibbles Away, and even discourage those who try to deprotect their programs altogether. This protection scheme is interesting in that one can usually use COPYA to copy the disk, but the copy will not run. Somehow, the programs know that an original disk is not in the drive.

The basis of Sierra On-Line's protection scheme is called a *nibble count*. When the master disk is copied at On-line, the copy program counts the number of bytes on a certain track and then stores this value on the disk. This value is different for each copy of the program because miniscule changes in the disk drive speed can cause extra or fewer bytes to be put on a track when it is written. Normal DOS does not care about this, since it uses a certain byte (FF) as a filler. When a disk is booted, it reads that track and compares the number of bytes read to the number stored on the disk. If they don't match, the program bombs out! Locksmith and several other bit copy programs allow you to do nibble counting when making a copy. However, if you have ever attempted this, you know how very difficult it is and how long it takes to get a reliable copy. Though the disk drive speed is highly regulated, it only takes a fraction of a turn to make a difference of a few bytes on the track.

The best way around the problem is to find the protection code and disable it. Fortunately for us, Sierra On-line uses the same technique on a number of their disks, so if you figure out how to copy one, you can copy a number of them. If you are interested in how they implement nibble counting, read on. Otherwise, skip the next section and go on to the step-by-step "cookbook" instructions.

Real Men Write Self-Modifying Code

The protection code used by Sierra On-line is a great example of hiding the true function of machine code! Read on, and you will see examples of code within code within code. A typical disassembly of the protection code might start out:

```
0900-  CE 03 09   DEC $0903
0903-  EF         ???
0904-  03         ???
0905-  09 AD 28   ORA #$AD
0907-  28         PLP
0908-  09 49      ORA #$49
```

It looks like there really isn't anything there. But notice that the first instruction decrements the very next byte! If we look at this same code after the first statement executes, we see:

```
0900-  CE 03 09   DEC $0903
0903-  EE 03 09   INC $0903
0906-  AD 28 09   LDA $0928
0909-  49 8A      EOR #$8A
090B-  D0 01      BNE $090E
090D-  20 8D 28   JSR $288D
0910-  09 18      ORA #$18
```

It doesn't even look like the same code! Notice, too, that the "new" instruction at $0903 restores $0903 to its original value of $EF so that the program hides itself again. From there, a simple sequence of instructions first gets a byte from $0928 (which happens to be $60), EORs it with $8A (giving $EA) and then executes a branch instruction. As I have shown, the accumulator is not zero, so this branch is taken...*right into the middle of another statement!* $090E is in the middle of the following JSR! However, by listing from $090E, the following code is revealed:

```
090E-  8D 28 09   STA $0928
0911-  18         CLC
0912-  D0 01      BNE $0915
0914-  4C A0 29   JMP $29A0
0917-  98         TYA
```

Ah ha! There is really a STA hiding there, and it stores the accumulator back to $0928. Then a clear carry is done to set up the carry flag to be used in a moment. The branch is always done since the accumulator doesn't end up being zero (it is $EA, from above). Once again, this branch is to the middle of another instruction! Listing from $0915, you see the following:

```
0915-  A0 29      LDY #$29
0917-  98         TYA
0918-  90 01      BCC $091B
091A-  20 59 00   JSR $0059
091D-  09 99      ORA #$99
```

So, another bit of hidden code! The Y-register is loaded with the value $29, copied into the accumulator, and a branch is made (remember the carry flag that was cleared above?). Once again, the branch is to the middle of another instruction (are you beginning to get the picture?). Disassembling the program from $091B gives:

```
091B-  59 00 09   EOR $0900,Y
091E-  99 00 09   STA $0900,Y
0921-  C8         INY
0922-  D0 F3      BNE $0917
0924-  88         DEY
0925-  30 01      BMI $0928
0927-  4C EA E1   JMP $E1EA
```

Here at last is the core of the code (no pun intended). This section does an Exclusive OR of a memory location with the accumulator and then stores the number back into memory. The Y register (used as the index) is incremented and the branch to the top of the loop is taken until Y is zero (it counts from $29 to $FF). When the Y register is zero it is decremented, and since it is now negative, the branch is made into the middle of another instruction. You notice that this is at $0928, our old friend from above which was converted from a $60 (RTS) to an $EA (NOP). Once all the EOR's have been executed on memory from $0929 to $09FF, the following code is revealed.

```
0928-  EA         NOP
0929-  C8         INY
092A-  8C F4 B7   STY $B7F4
092D-  8D EC B7   STA $B7EC
0930-  A9 B7      LDA #$B7
0932-  A0 E8      LDY #$E8
0934-  20 D9 03   JSR $03D9
0937-  BD 89 C0   LDA $C089,X
093A-  A9 05      LDA #$05
093C-  8D 00 BB   STA $BB00
093F-  20 90 09   JSR $0990
0942-  10 01      BPL $0945
0944-  20 C8 C0   JSR $C0C8
0947-  30 5D      BMI $09A6
0949-  8C C0 90   STY $90C0
094C-  F8         SED
094D-  BD 8C C0   LDA $C08C,X
0950-  10 0A      BPL $095C
0952-  C9 C9      CMP #$C9
0954-  D0 0D      BNE $0963
0956-  BD 88 C0   LDA $C088,X
0959-  4C 00 09   JMP $0900
```

This is the start of On-line's infamous "nibble count" routine which reads track zero and counts the bytes on the track. If the count is not correct, then the program proceeds to wipe out the computer's memory. If the count is correct, a jump is made to $0900 which is the start of the entire routine. The EOR operations are performed all over again and the result is that the valid code is returned to its original, obscured state. Pretty sly those On-line people!

How does the routine ever stop? Remember that the program first EORed location $0928 ($60) with $8A and then stored the result ($EA) back into $0928, which was later branched to? The second time the program is executed, the contents of $0928 ($EA) are EORed with $8A, giving a value $60 (the original value) which is then put back into $0928. When the branch to $0928 is later executed, it encounters $60, which just happens to be the machine code for RTS!

As you can see, the people who write the protection schemes are clever. However, sometimes they are so clever that they fool themselves, and that is their downfall. Apparently, they believed that the above code was so clever, and that it was so well hidden that they relied totally on IT protecting their software. In other words, it is the only check done to see if the disk in the drive is an original. Also, probably since Sierra On-line publishes software written by a number of authors, this protection program is called as a stand-alone subroutine, returning no values! The result is that it can usually be easily defeated just by changing the first byte to $60 to keep it from executing at all!

I mentioned above that usually the protection scheme could be defeated by changing the first byte of the protection code to $60. One case where this won't work is in Time Zone. Time Zone does a checksum on the protection code several times before it actually executes the protection code (Sammy Lightfoot also does a checksum between loading each level after the first!). The checksum code for Time Zone, whose nibble count routine lives at $1700, looks something like the following:

```
          Initialize Pointer to $1700
50AC-  A9 00      LDA   #$00
50AE-  85 FE      STA   $FE
50B0-  A9 17      LDA   #$17
50B2-  85 FF      STA   $FF
          Initialize all Registers
50B4-  A9 00      LDA   #$00
50B6-  A0 00      LDY   #$00
50B8-  A2 00      LDX   #$00
          Ready Carry for Add
50BA-  18         CLC
          Add Memory Byte
50BB-  71 FE      ADC   ($FE),Y
          Multiply by 2
50BD-  0A         ASL
          Increment Index
50BE-  C8         INY
     Decrement Counter (256 Times)
50BF-  CA         DEX
          Loop until Done
50C0-  D0 F9      BNE   $50BB
     Compare Checksum to $1E
50C2-  C9 1E      CMP   #$1E
50C4-  EA         NOP
          Branch if Equal
50C5-  F0 03      BEQ   $50CA
     $09CB Contains JMP $09CB
50C7-  4C CB 09   JMP   $09CB
     Return to Calling Routine
50CA-  60         RTS
```

The above assumes the protection code is located at $1700, which it is in Time Zone. The easiest way to foil this checksum routine is to change the first byte of the nibble count routine to a $60 (RTS) and also to alter the next byte of the routine so that the checksums will be identical even though the code has been changed. By experimenting with the code listed above, it turns out that the second byte of the nibble count routine should be changed to a $E0. However, this will only work on Time Zone because the other programs seem to use a different checksum routine. For these other programs (Oils Well, Sammy Lightfoot, etc.) it turns out that the second byte needs to be changed to a $AD. Screenwriter II ver. 2.2 and The Dic*tion*ary do not have the checksum routines, so you can get away with just changing the first byte of the nibble count routine to a $60.

The Steps

In a step-by-step fashion, here is what you need to do to make a copyable version of many of Sierra On-line's programs:

1) COPYA the original disk.

RUN COPYA

2) Run your sector editor. Examine each sector for the values $CE $03. If you have an editor such as Disk Zap from Bag of Tricks, or Tricky Dick with The Tracer you can automatically search for this sequence. I have found that this sequence of bytes is usually at the very beginning of a sector.

Note: Sometimes this code appears more than once, as in Screenwriter II, so be sure to search the entire disk to find all occurences!

3) Change the $CE to $60 and the $03 to an $AD ($E0 for Time Zone) and rewrite the sector, and that's all there is to it!

Program	Trk	Sec	Byte	From	To
Screenwriter	1A	0E	00	CE	60
II	08	0F	00	CE	60
version 2.2	0C	0F	00	CE	60
	17	0F	00	CE	60
The Dic*tion*ary	10	0D	00	CE	60
Sammy Lightfoot	05	0E	00	CE	60
	05	0E	01	03	AD
Time Zone version 1.1	03	0F	00	CE	60
	03	0F	01	03	E0
Apple Cider Spider	12	01	00	CE	60
	12	01	01	03	AD
Oil's Well	10	0F	00	CE	60
	10	0F	01	03	AD
Cannonball Blitz	18	06	00	CE	60
	18	06	01	03	AD

An alternate method which worked for Sammy Lightfoot and Time Zone involves disabling the code which does a JSR to the nibble count routine. This technique may also work on other SOL disks which directly JSR or JMP to the nibble count routine. Apple Cider Spider and Oil's Well apparently do an indirect JMP or JSR to their nibble count routines so this alternate technique will not work on them.

Alternate Method

1) COPYA the disk.

RUN COPYA

2) Use a sector editor to search the disk for a byte sequence of $CE $03 which is usually found at the beginning of a sector.

3) When you find this sequence, look at the byte which lies eight bytes past the $CE. This byte is the high order byte of the address where the nibble count routine runs. Write down the value of this byte. For example, if the CE is found at the beginning of the sector and the eighth byte is 09, then the protection code is located at $0900. Now search the disk for a JSR $0900 (20 00 09). This will be the call to the protection code.

Note: There may be more than one call to this code, so be sure to search the entire disk!

4) Change the JSR XXXX you find to EA EA EA, rewrite the sector, and you are done!

For Time Zone and Sammy Lightfoot, here are the sector edits which are needed.

Program	Trk	Sec	Byte	From	To
Sammy Lightfoot	0D	00	9B	20	EA
	0D	00	9C	00	EA
	0D	00	9D	9E	EA
Time Zone version 1.1	03	0B	F0	20	EA
	03	0B	F1	00	EA
	03	0B	F2	17	EA

Hint: If you try the first method and everything seems to work fine up to a point and then the program just hangs, try the alternate method.

The Visible Computer: 6502
Software Masters
3330 Hillcroft Suite BB
Houston, TX 77057
$49.95

Requirements:
Super IOB program
Blank disk
The Visible Computer program

The Visible Computer: 6502 is an excellent educational program designed to teach 6502 machine language to those would-be Bill Budges among us. The program allows the user to step and trace through machine language programs, displaying the effects upon registers and memory in hi-res graphics. The package comes with very good documentation but unfortunately, the disk is copy-protected. With the following procedure The Visible Computer: 6502 can be copied and modified to work with normal DOS 3.3.

The Visible Computer uses a modified DOS as its primary form of protection. In addition to the DOS marks being altered and the encoding manner, most of the DOS commands have been obliterated. Because of a few mistakes by the creators of The Visible Computer, we can easily capture its strange RWTS and use it to copy the disk. Once a copy has been made, the main program needs a few modifications in order to work with DOS 3.3.

Start by turning on your computer. Next, place your original version of Visible Computer in the disk drive but DO NOT CLOSE THE DRIVE DOOR. Type

CTRL C

and shut the door. In a few seconds you will get a "Break"message and a cursor.

Right away I thought I was 'in' to the visible computer. But the RUN flag gets set during The Visible Computer's boot process. Therefore, anything you type will execute the program in memory (never to give control back to you).

This is where we take advantage of their first mistake. We type

FP

which clears the program in memory but more importantly, clears the RUN flag. Now you are 'in' to the Visible Computer. Sure the DOS commands have been obliterated, but you can still call them directly. Try a

CALL 42350

to see the CATALOG.

What we have to do now is save the RWTS portion of the DOS in memory so that Super IOB can use it. This is accomplished by the monitor command

1900<B800.BFFFM

Next, boot your Super IOB disk and save the RWTS with

BSAVE VISICOMP.RWTS,
A$1900,L$800

With this file, all you have to do is use the Super IOB controller from page 27.

Once the copy has been made, there are a couple of modifications and deletions that must be made to the main Applesoft program called VC so that it will work properly with DOS 3.3.

The first deletion is line 1 of the program. This line is a REM statement containing a CTRL M, a CTRL D and the DOS command FP. This has the effect that if the line 1 is listed, DOS will process the FP command and delete the program. This problem is easily rectified by deleting line 1.

The BSAVE command in the Visible Computer's DOS has been changed to "SOREN". The only place this command appears is in line 13910. Of course, the normal DOS 3.3 command "BSAVE" must be restored to this line.

Line 18000 determines whether the program can Read and Write to normal DOS 3.3 , depending upon the setting of the Visible Computer's "MASTER" command. This line contains two calls to The VC's modified DOS which need to be removed. With these two calls removed, the "MASTER" command will still function normally except that the program will not be able to read files from the original Visible Computer program disk.

Finally, and most importantly, line 20730 contains a CALL that will INIT the disk if the program is used under normal DOS. The INIT command has been removed from the Visible Computer's DOS, so the CALL is harmless until the program is run under DOS 3.3. This nasty little line must be completely deleted.

Here are the exact steps to go through to deprotect The Visible Computer.

Revealing The Visible Computer

1) Turn on the computer and tell it to break

CTRL C

2) Insert the Visible Computer disk and shut the drive door.
3) Clear the RUN flag

FP

4) Enter the monitor

CALL-151

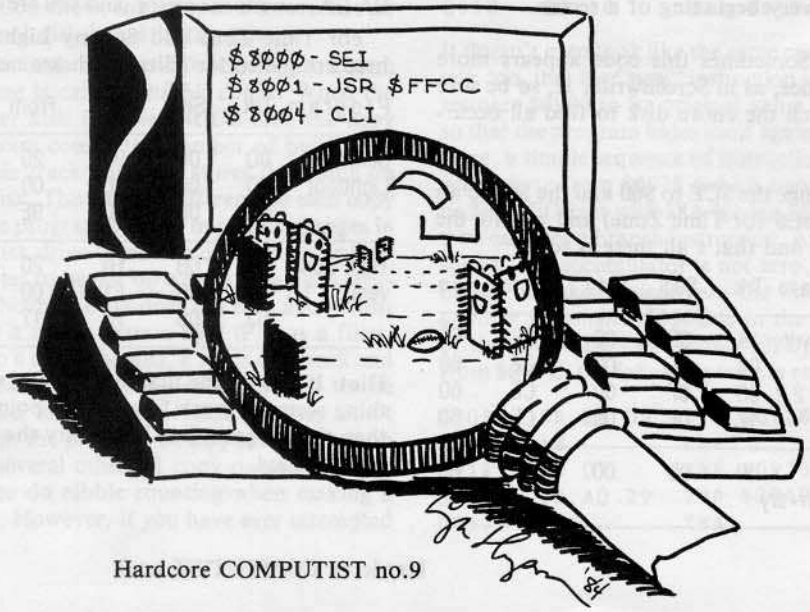
5) Move the RWTS to a safe place

1900<B800.BFFFM

6) Boot a 48K slave disk with a short HELLO program

C600G

7) Save the RWTS



Softkey For The Visible Computer

By Jared Block & Bob Bragner

BSAVE VISICOMP.RWTS, A$1900,L$800

8) Load Super IOB and type in the controller on this page.

SAVE SUPER IOB.VISICOMP

9) Copy the Visible computer

RUN

10) Load the main program from the copied disk

LOAD VC

11) Delete line 1 so the program can be listed normally

1

12) List line 13910.
It should read:

```
13910 VTAB 20: PRINT D$''SOREN''P2
      $Z$
```

Change this line so it has a normal BSAVE command

```
13910 VTAB 20: PRINT D$''BSAVE''P2
      $Z$
```

13) Line 18000 contains some CALLS to THE VC's DOS which must be removed. This line now reads:

```
18000 PV = T: CALL 47741 : IF P2$ =
      ''OFF THEN PV = F:CALL 47741: REM
        SET NRML,PRTCT DOS'S
```

Change this line so it reads;

```
18000 PV = T: IF P2$ = ''OFF'' THEN
      PV = F: REM SET MASTER ON/OFF
```

14) Line 20730 contains a CALL which will INIT normal DOS 3.3 disks.
It reads:

```
20730 CALL PEEK (40222) + PEEK
      (40223) * Q4 + 1
```

Be sure to get rid of it by typing

20730

15) If you wish you can also have the program set the RESET vector so that the program does not reRUN itself when the RESET key is pressed. For instance, if you want to enter the monitor when RESET is pressed change line 2 to read:

```
2 POKE 1010,89: POKE 1011,255:CALL
    -1169
```

16) Finally SAVE the modified program back to the copied disk

SAVE VC

As a CATALOG should reveal, this disk is completely broken and can be copied with COPYA or even FID. Frequent sessions with TVC are recommended for aspiring bit-brains. Good luck!

Visible Computer Controller

```
1000 REM SWAP CONTROLLER (VISIBL
     E COMPUTER)
1010 TK = 3:ST = 0:LT = 35:CD = W
     R
1020 T1 = TK: GOSUB 490: GOSUB 36
     0
1030 GOSUB 430: GOSUB 100:ST = S
     T + 1: IF ST < DOS THEN 1030

1040 IF BF THEN 1060
1050 ST = 0:TK = TK + 1: IF TK <
     LT THEN 1030
1060 GOSUB 490:TK = T1:ST = 0: GOSUB
     360
1070 GOSUB 430: GOSUB 100:ST = S
     T + 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 "EVERYTHING O.
     K. DOS NOT COPIED": END
10010 IF PEEK (6400) < > 162 THEN
      PRINT CHR$ (4)"BLOADVISICO
     MP.RWTS,A$1900"
```

The Visible Computer:6502
and
The Apple II-6502 Assembly Language Tutor

A review and comparison of two products which can help you learn the ins and outs of 6502 Assembly Language

Reviewed by Martin Collamore

The Visible Computer: 6502
Software Masters
3330 Hillcroft, Suite BB
Houston, TX 77057
$49.95

Introduction

The Visible Computer: 6502 (henceforth known as TVC) is a text and program designed to teach assembly programming. The text introduces an assembly command, then its execution can be viewed step-by-step on the simulator program's hi-res display.

So far as I can tell, it is the only product of Software Masters, Houston, Texas. TVC was written by Charles Anderson and retails for $49.95.

Hardware Requirements

To use TVC, you'll need a II+ or IIe and one disk drive. Apple II standards require a 16K RAM card or Applesoft ROM card. A printer is supported.

Documentation

The 141 page manual is spiral bound so that it lays flat (yeah!). The printing is good and the writing style is readable and amusing.

It begins with a good explanation of hex-dec-binary numbering and computer hardware configuration. After a session showing the operation of the simulator, you're off to explore the hows and whys of opcodes.

A command is first explained in the text. Then you load an example program to be run using the simulator. The text provides a blow by blow account as the simulator shuffles data between registers. The explanations, in most cases, are quite thorough.

The pace is slow at first, as it should be. By the time stack operations are explained, you've covered the basics and, believe it or not, you are somewhat comfortable with them. The final examples are fairly complex, but well documented.

Screen Display

Upon booting the TVC disk, you'll be confronted with just about all the information you'd ever need to follow an assembly program's execution. The hi-res screen displays the contents of all the microprocessor's registers as well as a bit by bit display of the processor status register. Also shown are four registers that perform scratch pad functions during the execution of a programming step.

A number of other informative tidbits are displayed: a message window to show what the TVC is doing (reading, fetching, writing, etc.), a status area to display the execution speed selected, a window for dissassembly of the lines as they are executed, a window to show the dissassembly of the next instruction, and a cursor for user input.

Working With TVC

First off, the program is fun to watch. The values actually scoot from one register to another as the code is executed. You can control how fast this takes place, pausing after each "microstep" if you wish. Optionally, you can view two different ranges of memory during a run to see what affect the program is having. This is nice when stack operations are demonstrated because you actually see the locations change. Unfortunately, the use of this option covers up some of the registers displayed, but the important ones are still visible.

More goodies are at your disposal. Some of these are: hex-dec-binary calculator/converter, editing of memory locations and register contents, and an erase which is used to clear the display and view a hi-res page.

Now a few gripes. TVC is written mainly in Applesoft and, hence, executes rather slowly. Speedy typists will probably will be annoyed. It takes 4 seconds to get the cursor back if TVC didn't understand your command. If you've erased the screen and want to restore the TVC display, you wait 15 seconds. The author believes that lower case letters make for better comprehension. I have a hard time with "1" and "l" or "b" or "6 ". The default is lower case, but you can change to upper if you so desire.

The documentation walks you through 24 examples demonstrating the arcane and mysterious 6502 opcodes. It's a lot of information and it takes a few "ah ha!"s to absorb some concepts. Watching how an instruction is executed really goes a long way in making it less painful. A picture IS worth a thousand words. I encountered only one minor bug. The documentation showed one type of indexed addressing, while the simulator used another. No big deal; it was fairly obvious. Obvious? Did I say that? Gosh, maybe this thing is actually teaching me something!

The author states that the two main reasons for learning assembly language are: 1) speed and 2) more speed. All that speed is needed for things like sorting, music, and animation. The last example programs put-it-all-together and demonstrate a bubble sort, a keyboard "organ", and a hi-res drawing routine. If you've made it that far and are able to follow these more complex programs, then you are on your way to becoming a competent assembly language programmer.

Conclusions

The purpose of TVC is to teach. It's also a debugging tool, albeit a somewhat slow one. Although all 151 opcodes are executed correctly, it is not an exact copy of a 6502 and may arrive at an answer in a manner different than the "real thing".

If you are serious about teaching yourself assembly language, TVC is a good place to start. For the most part, the docs are clear, concise, and understandable. The simulator program, while not a paragon of speed, is a terrific way to "see" how the innards of that little 6502 monster tick.

.....

Apple II- 6502 Assembly Language Tutor
Prentice-Hall
Englewood Cliff, NJ 07632
$34.95

Introduction

The Apple II-6502 Assembly Language Tutor (text and disk) is, obviously, a method teaching assembly language. Short, user entered programs are used to demonstrate 6502 microprocessor commands. The accumulator and the various register's contents are displayed as the program is executed.

Its author, Richard Haskill, is a professor of engineering and chairperson of the electrical and computer engineering group at Oakland University, Rochester, Michigan. Not surprisingly, I/O and hardware interfacing techniques are emphasized.

Published by Prentice-Hall, it lists for $34.95.

Hardware Requirements

The Assembly Language Tutor (ALT) manual states that a 48K Apple II or Apple II+ with either DOS 3.2 or 3.3 (one drive needed) is supported. I did not try the program under DOS 3.2. In the Tutor, control-S is used to single step through a program. This is familiar to those with Autostart ROM. For those of us with old monitor ROM, a copy of the Autostart F8 ROM is needed. This requires either an Applesoft firmware card or a language card which has been loaded with a copy of the Autostart F8 ROM and read enabled. The manual says it also supports a printer, but I wasn't able to check on that.

Documentation

The paperback manual is 234 pages and indexed. The text and schematics are clear, though some of the photos are of poor quality. While readable, the manual's writing style is reminiscent of a text book. Exercises are presented at the end of chapters, although no solutions are offered. This text book approach is both vice and virtue; Dr. Haskell leads you in a definite, bit-oriented direction.

By page 5, you've been presented with a 6502 microprocessor pinout and a bit-wise description of its data bus. After the usual hex-dec-binary explanations and the instruc-

tions for the use of the TUTOR disk program, there is a thorough discussion of binary coded decimal, two's complement, and signed number representation. By page 85, all of the assembly commands have been presented and, if you've entered them in, demonstrated. Not exactly bedside reading; however, a good text book is also a good reference book.

The next 47 pages contain descriptions of the Apple video displays. Subjects include: raster scan, TV RAM, screen soft switches, and shape tables. Example programs appear on nearly every page and the treatment is fairly detailed.

Given the contents of the next hundred pages, it's easy to see Dr. Haskell's true love: this is where we separate the hackers from the chip heads. Even if the thought of a soldering iron causes heart palpitations, you'll still find this part informative.

These last chapters deal with I/O hows and whys. Ever wanted a 6 channel A/D converter connected to your game I/O socket? Make a peripheral board to support ROMs? How about a parallel or serial interface? Well, don't expect a Heathkit approach, just expect clear schematics, circuit descriptions, chip data sheets, and assembly language driver programs. None of these circuits are especially complex in construction, but some experience with the care and feeding of chips would be helpful.

Screen Display

The display is fairly simple: the six 6502 internal registers contents (in binary and hex), a disassembled line of the next instruction to be performed, and a raw hex dump of any 88 continuous bytes. A cursor is used in the dump display to edit bytes. At the screen's bottom is a command line. Hitting the "/" key brings up 14 single letter options, rather like Visicalc. Each of the options provide up to 6 single letter suboptions. This allows flexibility, but keep your manual handy.

Working With The Assembly Language Tutor

Aside from getting used to the command line options, using the TUTOR program is simple. The program lies between $8000 to $93B2 (safe from DOS boots) plus some zero page locations. Applesoft uses some of these locations, so there is no graceful way to exit except to DOS or the monitior.

In addition to control-S, you can set up to four break points to halt program execution. Or you can let it go full tilt. For debugging programs, this is fine. But single stepping through a large program will require a lot of key strokes. A word of warning: since you have free access to memory, you can bomb the system if you're not careful.

One nice touch is the ability to calculate branching offsets. Say you have a branching instruction to a line 8 bytes backwards from the next line in the program. The tricky part is that a "-8" would have to be expressed in two's complement, "F8". The TUTOR will calculate these offsets for you.

Conclusions

I admit that I have chip head leanings, so I found the marriage of assembly level software and do-it-yourself firmware interesting. Some might not. Even so, The Assembly Language Tutor provides a lot of detailed information, not only on assembly language programming, but also on how your Apple communicates with the rest of the world. It is a valuable reference book for both programmers and technicians.

.....

The Visible Computer: 6502
vs.
The Assembly Language Tutor

In my opinion, The Visible Computer: 6502 and The Apple- 6502 Assembly Language Tutor are both excellent educational packages for those wishing to learn assembly langauge. If you are considering the purchase of either of these two packages you should be aware that the approach taken by the two authors differs considerably.

The Visible Computer: 6502's software is definitely the more polished and "user-friendly" (does this term really mean anything anymore?) of the two. The approach of TVC manual is also quite a bit more gradual and beginner oriented than that of ALT.

Because TVC is a 6502 simulation in software and operates in "microsteps", the novice gets to see the effect of each instruction on the CPU's registers. The ALT Tutor program whizzes along much faster and does not allow the user to view the effect of every instruction with single stepping. This gives TVC the edge in educational terms; however, the utility of TVC as a debugging tool for experienced programmers is reduced because of the simulator's slow speed. ALT's Tutor program, while not flashy, can serve as an assembly language development aid long after the intricacies of the language have been mastered.

The Assembly Language Tutor manual also contains a great deal of information that might come in handy for those wishing to design their own peripheral cards and/or write assembly language driver programs. TVC's manual does not contain any comparable material.

If you merely wish to learn assembly language programming and have no need or interest in paying homage to the God of Hardware, I would recommend The Visible Computer: 6502. It will simultaneously entertain and educate you.

However, if you wish to both learn assembly language and find out how your Apple and other computers communicate with the outside world, The Apple II- 6502 Assembly Language Tutor is the product of choice as long as you can put up with the author's textbook approach.

Whichever package you choose, you shouldn't be disappointed.

ADVENTURE TIPS

Pirate Adventure
Adventure International

To "get parrot", you must have some crackers with you.
Wait for the tide to come in before you set sail.
Who knows best how to sail a "pirate" ship?

Mask Of The Sun
Ultrasoft

Shoot the snake as soon as he rises.
Place the urn on the right pedestal.
In the room with the webs, search though the debris on the floor.
To find the "key" to the problem, look for a "weighty" solution.

Colossal Caves
Adventure International

You need a giant opener for a giant clam.
The troll is scared easily.
Look for the pirate's stronghold in the maze whre all the rooms are alike.

Transylvania
Penguin Software

"Get bottle" from the hut.
Vampires don't like religious objects.
You need flypaper to "catch flies".
What would a frog like to eat?

Death In The Caribbean
Microlab

If you've been there once already, the ring can take you there again.
Don't paddle across the river. Shovel your way across.
Try another route around the alligator.
He's just there to give you trouble.
Reach into your pocket and flick your BIC!

Zork II
Infocom Inc.

That bucket looks big enough to hold a person.
Don't just have your cake. Eat it, too!
A flask of water is just like a magnifying glass.
Getting out of the steel cage will have to be a "robotic" effort.

Continued from page 21

```
1910            CMP #46         '.'S MATCH
1920            BEQ .1
1930            CMP SECONDARY,X MATCH?
1940            BEQ .1
1950            STX FUNC
1960            RTS
1970
1980 CLR.WS     LDA GX          FAKE WIDTH
1990            STA $21         WNDWDTH1
2000            LDA LY          CV
2010 .1         PHA             SAVE CV
2020            JSR BASCALC     GET ADDR
2030            LDA #$AE        PERIODS
2040            LDY LX          CH
2050            JSR CLEOL2      STORE EM
2060            PLA
2070            ADC #$00        C=1
2080            CMP GY          WNDBTM
2090            BCC .1          NEXT LINE
2100            LDA #40         REAL WIDTH
2110            STA $21
2120            RTS
2130
2140 * ------------------------------
2150 * FIX BASIC ERRORS
2160 * ------------------------------
2170
2180            .HS 68A868A6
2190            .HS DF9A4898
2200            .HS 4860
```

Word Searcher Checksums

```
10   - $BADD
20   - $9B13
30   - $4D3B
40   - $AD92
50   - $C899
60   - $FF65
70   - $A3BF
80   - $A900
90   - $924D
100  - $111D
110  - $EB5D
120  - $F0BA
130  - $68D8
140  - $FB89
150  - $9FBB
160  - $04A0
170  - $F070
180  - $523B
190  - $7422
200  - $16F1
210  - $4298
220  - $696C
230  - $BC64
240  - $2967
250  - $1A5B
260  - $4724
270  - $1A41
280  - $9849
290  - $95E7
300  - $67DE
310  - $4D91
320  - $A6C0
330  - $776E
340  - $85F8
350  - $D21B
360  - $F790
370  - $EEA5
380  - $EE3E
390  - $E22E
400  - $F4C7
410  - $A30B
420  - $54EA
430  - $0CE0
440  - $5008
450  - $9375
460  - $B913
470  - $4D6D
480  - $FA27
490  - $AFCA
500  - $1574
510  - $ECEE
520  - $A14F
530  - $36E6
540  - $09CF
550  - $386B
560  - $E5BA
570  - $42AF
580  - $164F
590  - $216F
600  - $C111
610  - $10DD
620  - $5398
630  - $2A19
640  - $AFAD
650  - $A43B
660  - $6F9D
670  - $9FDD
680  - $2DAC
690  - $AA9B
700  - $667E
710  - $97EE
720  - $06E0
730  - $ADD2
740  - $2365
750  - $D40D
760  - $F7A0
770  - $B8F2
780  - $8B83
790  - $3A5C
800  - $7038
810  - $C83B
820  - $ECA4
830  - $19AD
840  - $297E
850  - $6560
860  - $A050
870  - $E2BE
880  - $74D7
890  - $484F
900  - $1B64
910  - $C5BD
920  - $2987
930  - $B6BD
940  - $D3D8
950  - $65F9
960  - $89B4
970  - $7EBD
980  - $8BA8
990  - $701F
1000 - $5AF8
1010 - $0259
1020 - $68E4
1030 - $95E4
1040 - $ED74
1050 - $DB45
1060 - $C633
1070 - $3F0A
1080 - $2E63
1090 - $7538
1100 - $1568
1110 - $650F
1120 - $1E0D
1130 - $D940
1140 - $360C
1150 - $D108
1160 - $BBFE
1170 - $1958
1180 - $4D87
1190 - $16E0
1200 - $913F
1210 - $1B5F
1220 - $34BF
1230 - $91D1
1240 - $C576
1250 - $A1B4
1260 - $B784
1270 - $89D9
1280 - $9D31
1290 - $8F46
1300 - $87C5
1310 - $FE41
1320 - $9484
1330 - $A91C
1340 - $6E8F
1350 - $DB8F
1360 - $9CDB
1370 - $64CA
1380 - $1FD0
1390 - $C6E7
1400 - $4038
1410 - $8824
1420 - $96E5
1430 - $DCC8
1440 - $E7D5
1450 - $E7E1
1460 - $FE71
1470 - $A4CE
1480 - $8915
1490 - $4134
1500 - $F2CE
1510 - $B308
1520 - $833D
1530 - $83A0
1540 - $F235
1550 - $987F
1560 - $B9A3
1570 - $CC93
1580 - $E5C9
1590 - $F0F4
1600 - $DAB2
1610 - $0490
1620 - $F8B7
1630 - $19EF
1640 - $CD46
1650 - $FA02
1660 - $9E10
1670 - $47BC
1680 - $E3C2
1690 - $F49C
1700 - $61D9
1710 - $4821
1720 - $A90A
1730 - $8FD0
1740 - $1B80
1750 - $7F65
1760 - $3426
1770 - $A510
```

Continued from page 8

Obviously, this was not the right course of action.

The next approach is to use a deprotection tool such as Super IOB. Super IOB's 'swap controller' allows us to read in the protected disk with the protected disk's DOS and write it back out to a normal DOS 3.3 disk. The controller starts reading the protected disk at track 3 (it does not copy the protected DOS on tracks Ø through 2) and copies up to track 34. But first we must capture the protected DOS's RWTS and store it at $19ØØ for Super IOB to use.

To do this, boot Cosmic Combat and when the BASIC prompt appears at the lower left of the screen, Reset into the monitor. Then move the RWTS portion of DOS down to $19ØØ with the monitor move command by typing 19ØØ<B8ØØ.BFFFM. Next boot a slave disk and run Super IOB.swap. The swap controller will use the foreign RWTS at $19ØØ to read the protect disk. Since we are booting a slave disk, memory from $9ØØ to $95FF will not be destroyed and hence our relocated RWTS will not be disturbed.

After using Super IOB to copy the disk, Cosmic Combat is deprotected and we can CATALOG our copy and BLOAD (or FID the files to another disk) and examine the code to our heart's desire!

With SUPER IOB.SWAP and Cosmic Combat in hand, here are the steps to deprotection:

1) Boot normal DOS 3.3 and initialize a disk with "HELLO" as the boot program by typing

INIT HELLO

2) Boot the COSMIC COBMAT disk and when the prompt appears at the bottom of the screen, Reset into the monitor.

3) Move the RWTS portion of DOS down to $19ØØ

19ØØ<B8ØØ.BFFFM

4) Boot the disk you Initialized in step 1

C6ØØG

5) Put in your Super IOB disk and use the version of Super IOB that has the swap controller

RUN SUPER IOB.SWAP

You may now run, CATALOG, or load any of the files from the backup disk or transfer them to another disk with FID. The start-up program is, of course, HELLO, which will run Cosmic Combat.

Do you need BACK ISSUES?

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

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

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

- Hardcore COMPUTIST #1 ..... $3.50 ☐
- Hardcore COMPUTIST #2 * ..... $3.50 ☐
- Hardcore COMPUTIST #3 * ..... $3.50 ☐
- Hardcore COMPUTIST #4 ..... $3.50 ☐
- Hardcore COMPUTIST #6 ..... $3.50 ☐
- Hardcore COMPUTIST #7 ..... $3.50 ☐
- Hardcore COMPUTIST #8 ..... $3.50 ☐
- CORE #1 Graphics ..... $4.50 ☐
- CORE #2 Utilities ..... $4.50 ☐
- CORE #3 Games ..... $4.50 ☐

* Limited supplies

SPECIAL OFFER!

Order these magazine/disk combinations and SAVE.

- CORE #1, Hardcore COMPUTIST #1, and Library Disk #1 all for only ..... $19.95 ☐
- CORE #1,2,3 ..... $10.00 ☐

Name _____

Address _____

City _____ St _____ Zip _____

Country _____

VISA/MC _____ Exp _____

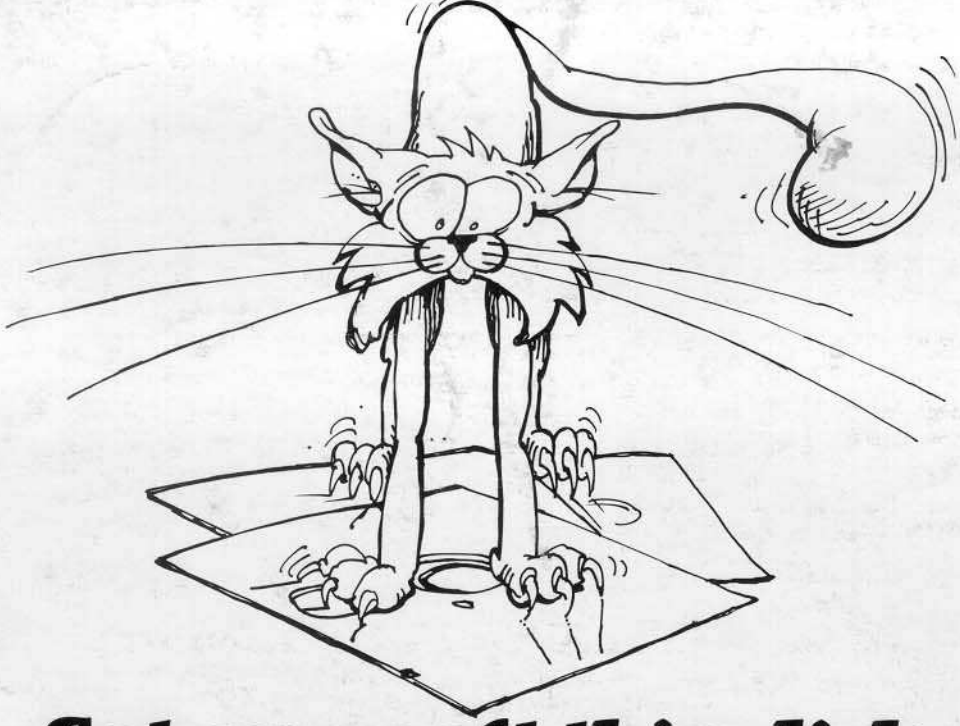
Signature _____

Send check or money order to:

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

Washington state residents add 7.8% sales tax. Foreign orders add 20% shipping and handling. U.S. funds only.

- Library Disk #5 ..... $9.95 ☐
  - Hardcore COMPUTIST #7:
    - Corefiler
    - Disk Directory Designer
  - Hardcore COMPUTIST #8:
    - Corefiler Formatter
- Library Disk #4 ..... $9.95 ☐
  - Hardcore COMPUTIST #6:
    - Modified ROMs
    - Crunchlist
    - Crucial Code Finder
- Library Disk #3 ..... $9.95 ☐
  - CORE Games issue:
    - Destructive Forces
    - Dragon Dungeon
- Library Disk #2 ..... $19.95 ☐
  - CORE Utilities issue:
    - Hi-Res Utilities
    - Dynamic Menu
    - Line Find
    - GOTO Replace
    - GOTO Label
    - Fast Copy
  - Hardcore COMPUTIST #3:
    - Map Maker
  - Hardcore COMPUTIST #4:
    - Ultima II Character Generator
- Library Disk #1 ..... $19.95 ☐
  - CORE Graphics issue:
    - Scruncher
    - Quick Draw
    - QD.Editor
    - Design Plus
    - Faster Shapes
    - Space Raid
  - Hardcore COMPUTIST #1:
    - Checksoft
    - Checkbin
  - Hardcore COMPUTIST #2:
    - Page Flipper
    - String Plotter
    - Wall Draw
- Disk Control ..... $15.00 ☐
  - Disk Edit
  - IOB
  - Menu
  - Disk View



Cat on a soft thin disk.

You need software insurance.

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

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

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

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

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

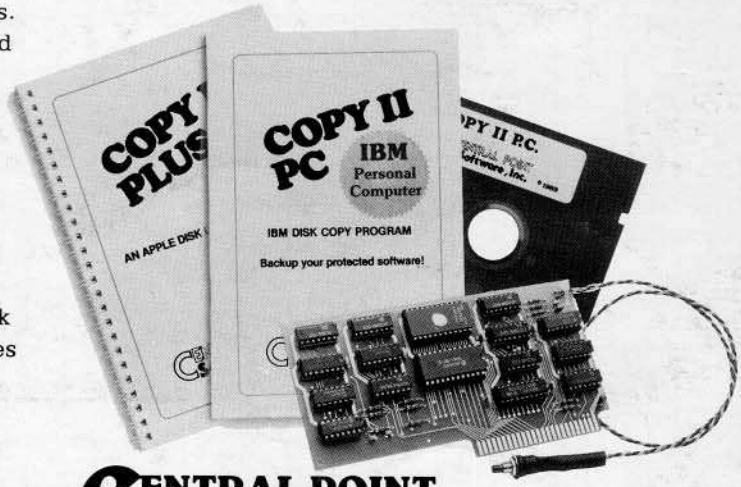
Important Notice: These products are provided for the purpose of enabling you to make archival copies only. Under the Copyright Law, you, as the owner of a computer program, are entitled to make a new copy for archival purposes only, and these products will enable you to do so.

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

Copy II PC (IBM)

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

We are the backup professionals. Instead of diluting our efforts in creating a wide variety of programs, we specialize in offering the very best in backup products. So, protect your software investment, and get surefire relief from scratchy disks.



CENTRAL POINT Software, Inc.

The Backup Professionals

To order, call 503/244-5782, 8:00-5:30 Mon.-Fri., or send your order to: Central Point Software, 9700 SW Capitol Hwy, Suite 100, Portland, OR 97219. **Prepayment is required. Please include $2 for shipping and handling ($8** outside U.S. or Canada).