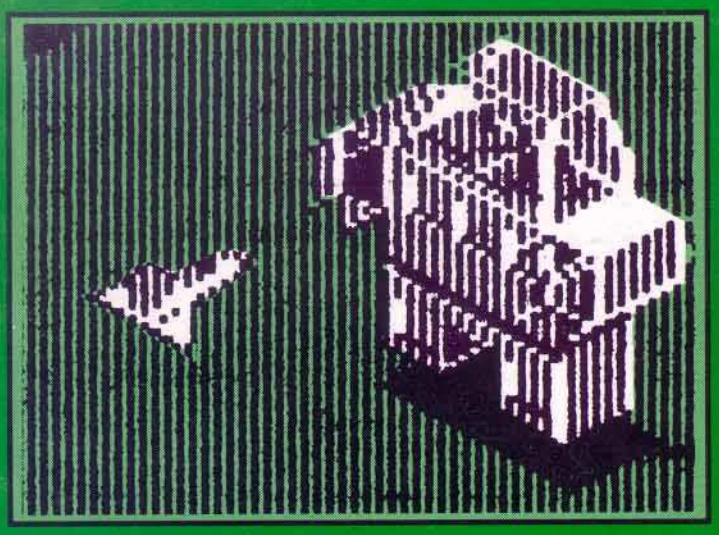
HOW TO BACKUP, UNLOCK, OR MODIFY COPY-PROTECTED SOFTWARE

Hardcore

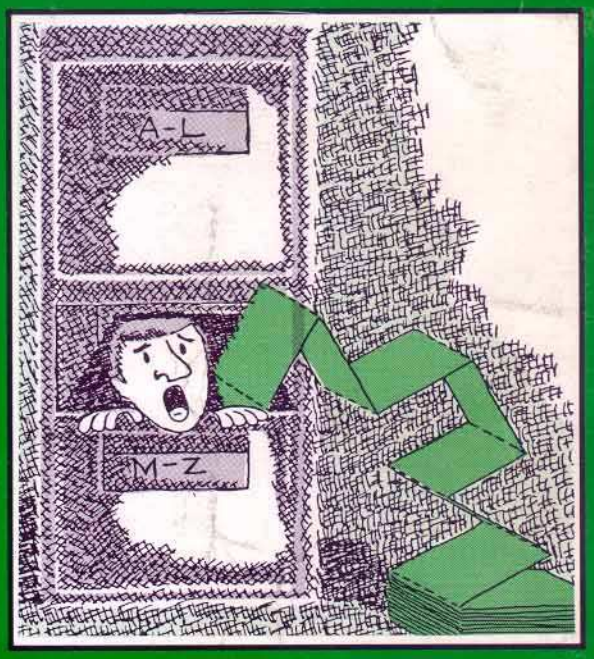
COMPUTIST

Volume 3 Number 3 $2.50



ZAXXON:

A Softkey for the Apple Version of the Arcade Classic.



COREFILER:

A Complete File Management System for the Home or Small Business

DISK DIRECTORY DESIGNER:

Create a New Look for your Disk's Catalog

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

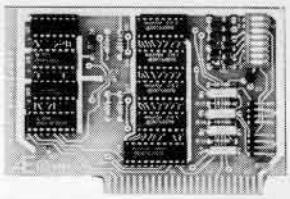
NEW 1984 DESIGN
An official PRO-DOS Clock

- 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

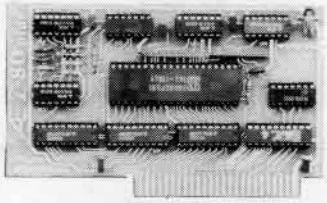
Super Music Synthesizer
Improved Hardware and Software



- 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	80 COLUMN OVERRIDE	INVERSE CHARACTERS
VIEWMASTER	169	YES	YES	YES	YES	YES	YES	YES	YES
SUPRTERM	MORE	NO	YES	NO	NO	NO	YES	NO	YES	YES
WIZARD80	MORE	NO	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

Hardcore

COMPUTIST

THIS ISSUE:

Hardcore
COMPUTIST

Publisher/Editor
Charles Haight

Business Manager
Ken Fields

Technical Editing
Gary Peterson

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

Programmer
Ray Darrah

Printing
Grange Printing, Inc.
Seattle, WA

Production & Graphics
Lynn Campos-Johnson

Publishing
SoftKey Publishing
P.O. Box 44549
Tacoma, WA 98444
USA

Circulation
Valerie Robinson

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. All subscription inquiries should be directed to: Hardcore COMPUTIST, Subscription 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.

We now offer surface rates for our foreign subscribers.
See pg. 31 for information.

INPUT INPUT INPUT

Some Inspiring Words

I am writing for a few reasons. First of all, I want to thank you for publishing so many articles on how to softkey disks. Even though I don't need to backup many of them, I still enjoy learning new tricks about the Apple. I wish there had been more new programs (i.e. more than zero) in no. 5, but I can't complain.

I would like to get one small bit of information from you: the address of A.P.P.L.E. (Apple Pugetsound Program Library Exchange). It was mentioned in a letter by Thomas E. Militello (pg. 4) and I've heard of it before, but can't seem to find the address anywhere. Even though I live in the same town as the inventor of Wizardry, there doesn't seem to be any Apple users group around, which means I'm fairly isolated.

An article I would love to see is one which would list a skeleton library of books on the Apple. Just a list of what books you have found most useful for BASIC, assembler, graphics, etc. I personally like the McGraw-Hill/Osborne book on Apple][BASIC, but have no idea what books are good for machine language. What book(s) are good on DOS layout, on the monitor, and on machine architecture would also be wonderful to know. I'd also love to see a simple list of what software tools people have found useful/essential for program development. Through talking with people, I have found out about GPLE now sold Beagle Bros. I wish I had known about this software months ago; it would have saved a lot of painful editing. Right now, I'd like to get a macro assembler, but have no idea where to go to look for a comparison of what's available. A little known book that I'd like to recommend is "Apple Graphics & Arcade Game Design" by Jeffrey Stanton, The Book Company. It's far and away the best book I've seen on the subject; unlike 95% of all other Apple graphics books, which may devote their last chapter to shape tables, this book gets into a lot of techniques mentioned in Hardcore COMPUTIST. After a third of the book, you're past shape tables and getting into machine language coding. Chapters cover hi-res screen architecture, bit-mapping, and arcade techniques such as page-flipping, steering and firing, collisions, explosions, and scrolling. The whole book centers around making a Defender-like arcade game (which is left unfinished as an "exercise for the reader"). The book is 288 pages with very little space wasted and a lot of usable assembly language routines.

You've probably seen "Enhancing Your Apple][, Vol. 1", by Don Lancaster. It's pretty interesting, but relies too heavily on hardware changes to get results. The enhancements are clever and cheap, but make software using them great for the hobbyist and that's all. His "Tearing into Machine Code" chapter looks pretty good; interesting methods to figure out how someone else's clever code works. So far, the part I've most enjoyed is the introduction, where he says something that you might quote in Hardcore:

"Undoing copy-protection is fun!"

'Not only is it fun, but cracking the uncopyable is about the most challenging and rewarding thing that you can possibly do with your Apple. And, the things you learn along the way are exactly the skills that you will need to become a really great programmer."

Inspiring words indeed!

Eric Haines
Ithaca NY

Mr. Haines: The address of Call A.P.P.L.E. is:

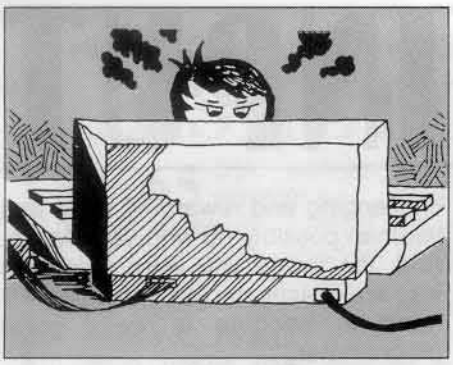
21246 68th Ave. S.
Kent, WA 98032
Phone (206) 872-2245

Books we have found useful include:

Beneath Apple DOS
By Don Worth and Peter Lechner
Quality Software

What's Where in the Apple
Micro Ink

Using 6502 Assembly Language
By Randy Hyde



Most Wanted List

We were overwhelmed with the number of Softkeys that were recently submitted to us for Sensible Software's "Sensible Speller" and Sir Tech's "Legacy of Llylgamyn." We would like to thank all of the people who responded to the input column of Hardcore COMPUTIST No. 6.

This has prompted us to regularly print a list of the most wanted softkeys. If there is a program that you have just been pulling your hair out over backing it up, let us know about it. Send your vote for the most wanted softkey to:

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

Below is the first list. If you know how to deprotect, 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. For a complete format description, ask for our Writers guide.

1. **Bank Street Writer**
   Broderbund
2. **PFS File / PFS Report**
   Software Publishing Corp.
3. **Zoom Graphics**
   Phoenix Software
4. **Flight Simulator II**
   Sub Logic
5. **Type Attack**
   Sirius Software
6. **DB Master 4.0**
   Stoneware, Inc.
7. **Ultima III**
   Origin Systems
8. **One on One**
   Electronic Arts
9. **Visiblend**
   Micro Lab
10. **The Accountant**
    Decision Support Software

Likewise, programs we would hate to be without include:

Bag of Tricks
Quality Software
$39.95

Copy II+ version 4.0 or above
Central Point Software
$39.95

Global Program Line Editor (GPLE)
Beagle Brothers
$49.95

The CIA
Golden Delicious Software
$65.00

The S-C Macro Assembler
S-C Software
$92.50

In addition to your recommendations we might add that all of Don Worth's other books (Micro Cookbook, The Hexadecimal Chronicles, The Incredible Secret Money Machine, etc) are also very good.

More on Screenwriter

I have had three different Screenwriter disks crash. A backup was purchased after the first crash. As a result of subsequent changes, I now possess one 2.0 version and one 2.2 version. I was excited when I saw the article ("Screenwriter II Softkey By Daniel Price in Hardcore COMPUTIST #5), and promptly verified the procedure on the 2.0 version. The 2.2 version could not be copied because it does not use the same file names. Furthermore, a portion of the catalog is unlistable.

The latter part of the article made reference to increasing the speed of booting Screenwriter II through the use of "Bag Of Tricks". For what it is worth, the 2.0 version boots from the menu in 14 seconds.

The faster booting rate of the 2.2 version is a distinct advantage and I would be very interested in seeing a comparable article entitled "Screenwriter II, Version 2.2, Softkey By Whoever Knows How".

Neil E. Walter
Rockville MD

Mr. Walter: Shortly after the softkey for Screenwriter 2.0 appeared in Hardcore COMPUTIST no. 5, one of our readers called and recommended the following procedure for backing up Screenwriter 2.2

1) Use COPYA to copy the disk.

2) Use some sort of disk scan utility (The Inspector, ZAP from Bag of Tricks, or The Tracer on The CIA disk) to search for a byte sequence of 20 00 7F.

3) This sequence should be found on track $0F, sector $07.

4) Change these bytes to EA EA EA and write the sector back to the disk.

We don't have Screenwriter 2.2 to try this out on, so please let us know whether or not this procedure works for you.

Egbugs

Thanks to Dr. Goldstein for his article "Cracking The Egbert II Communications Disk". I have already thanked you by sending more money.

However, there are a few errors (maybe mine). I did not use the information on DeMuffin Plus since I already had it. Between steps 6 and 7 (page 17) you need to add a step, *86000G. On page 18, check the name of your Combo Program. Mine was 10-83. Also, the value at *54BC was AB. I changed it to 20 as the article suggested.

On page 19 in the RTTY and ECW program, you should not use the reserved word. Use "A" only. My RTTY program does not receive correctly. It prints at an angle going up the screen. I'm still looking for this. Can anyone help?

Also, can anyone unlock Zoom Graphics?

Jim Willis West
Monroe LA

Mr. Willis:Unfortunately you are right; we did manage to introduce some bugs into Dr. Goldstein's softkey. Please see the corrections printed on page 30 of this issue.

We have amputated the toes of the person responsible for these bugs.

Hit List

Following are some ideas and thoughts regarding your publication:

It seems as though your two magazines are starting to appear on a more regular basis. This would seem to indicate that hopefully, we are over the hump and you are here to stay! Congratulations!

Also, the softkey articles in the latest issue seem to be more tutorial and step-by-step, so that those of us who

Continued on page 28

READERS' SOFTKEY & COPY EXCHANGE

Backing-Up Crush, Crumble & Chomp
By Jeff Rivett

Crush, Crumble, and Chomp
Automated Simulations (Epyx)
28035 Dorothy Drive
Agoura, CA 91301

Requirements:
Apple II Plus or compatible
Replay II card and utility disk with SOFTMOVE program
Crush, Crumble, and Chomp (Epyx)
An initialized DOS 3.3 disk with "HELLO" as the boot program
FID program from 3.3 System Master disk

Crush, Crumble, and Chomp is copy-protected only on areas of the disk that contain the two main Applesoft programs. The rest of the files, and there are a few of them, can be transferred to a copy disk using any file transfer program.

The main (copy-protected) files are called:

C^{CTRL}NCCMAIN
C^{CTRL}NCC

Run FID (or any other file transfer program) and copy all the files, except the two copy-protected ones, from the original to the blank initialized disk.

BRUN FID

After transferring the unprotected files to your initialized copy disk, boot the original Crush, Crumble and Chomp disk

PR#6

Make a REPLAY copy at the point where the set-up program starts and asks you the question "DO YOU WANT TO CONTINUE A SAVED GAME?".

Now, run "SOFTMOVE" from the Replay II utility disk. "SOFTMOVE" assumes that there was an Applesoft program in memory when you pushed the Replay button to make the copy. This Applesoft program is put back into memory intact, with a normal DOS resident instead of the copy-protected DOS.

You can now save the Applesoft file to the disk that contains all the files you have transferred. Save it with the name "C^{CTRL}NCC."

SAVE C^{CTRL}NCC

Boot the original again. This time let the set-up progress to the point where the play actually starts and then make another REPLAY copy. Repeat the "SOFTMOVE" procedure you used previously on the first protected program. Save this file as C^{CTRL}NCCMAIN.

SAVE C^{CTRL}NCCMAIN

That's all there is to it. These files can now be modified at will.

A last note: the "SOFTMOVE" program is a surprisingly useful utility. For instance, if you typed in a huge Applesoft program and forgot that there was no DOS in the machine, you might want to shoot yourself. With REPLAY II and "SOFTMOVE" you can make a REPLAY copy and have the file on disk in no time, almost as if DOS was in the machine all along.

Backing-Up Snake Byte
By Clay Harrell

Snake Byte
Sirius Software, Inc.
10364 Rockingham Drive
Sacramento, CA 95827

Requirements:
48K Apple, with old monitor F8 ROM
One disk drive, with DOS 3.3
Initialized 48K slave DOS 3.3 disk
Snake Byte

Snake Byte is a challenging game that requires a great deal of eye-hand coordination. But it is a lot of fun, and so is its deprotection!

In typical Sirius tradition, the disk is protected so that nibble copies are relatively difficult to make. But, there is another way! Snake Byte is not a very big program and to confirm this, Sirius has helped us out a great deal.

After booting the game and resetting into the monitor, the text page is covered with what appears to be the lowres version of the game. The author uses the text page for animation duties, so we know that we do not have to save that portion of memory ($400-800).

If we list memory from $800 to $1800, we find it all zeroes. Obviously, we do not have to save this since there is no valid code, there. Thanks Sirius!

Strolling down memory lane, we come to $8600 and notice the same phenomenon again! Sirius has blanked out memory from $8600 up to RWTS (which we don't need to save since no disk access is ever encountered after the game finishes its initial load). It is now obvious that we need to save $1800 to $85FF.

The question of the day then, is: why did Sirius/the author do this?

Anyway, figuring out what code to save was easy.

Now, where does the game start? Checking all the standard starting locations ($1800, $6000, etc.), nothing comes up. But if you are cunning and/or lucky, like me, you will find that $7697 seems to be the start of the game.

We now have all the information we need to deprotect Snake Byte.

Note that we really do not have to

save hi-res page one, since it is drawn on. For convenience' sake, however, we will.

Here's the exact procedure for the deprotection of Snake Byte:

1) Boot the Snake Byte disk

PR#6

2) After the drive stops, reset into the monitor.

3) Boot a 48K slave disk

PR#6

4) Enter the monitor

CALL-151

5) Enter the following code which turns on hi-res page two and waits for a keypress

17E9:AD 50 C0 AD 52 C0 AD 57 C0 AD 55 C0 AD 10 C0 AD 00 C0 10 FB 4C 97 76

6) Save the program

BSAVE SNAKE BYTE, A$17E9,L$6E19

Snake Byte Modifications

If you become totally frustrated with Snake Byte, here are some ways to beat the system.

Create this Applesoft program that runs the game:

```
5  POKE 1010,102: POKE 1011,213: POKE
     1012,112
10  IF A<>0 THEN 40
20  HOME: PRINT "AFTER SNAKE BYT
    E IS RUNNING PRESS RESET FOR
     MORE SNAKES OR FOR A DIFFE
    RENT STARTING LEVEL"
30  A=1: PRINT CHR$(4);"BRUN SNAKE BYTE"
40  HOME: VTAB 8: INPUT "HOW MAN
    Y SNAKES WOULD YOU LIKE?";SN$: SN
    =VAL(SN$):  IF  SN  >  2
    54 OR SN < 1 THEN 40
50  VTAB 10: INPUT "WHAT STARTING
     LEVEL?";SL$: SL = VAL(SL$)
    : IF SL > 21 OR SL < 1 THEN
     50
60  POKE 30382,SN
70  POKE 30360,SL
80  CALL 30359
```

Undocumented Feature

One last note on Snake Byte: Whenever the game asks: "HOW MANY PLUMBS?" you can touch the "C" key to change the keys used to direct the snake. Have fun!

Softkey For DB Master

By Dan Lui

DB Master
Stoneware, Inc.
50 Belvedere St.
San Rafael, CA 94901

Requirements:
Apple II, with 48K
One disk drive
DB Master (old version)
COPYA
The Inspector, or similar program
One blank disk

The old version of DB (Data Base) Master is protected very well. The disk contains three different protection schemes.

First, it uses half tracks from $6.5-$22.5. Second, the closing address and data marks have been changed from the normal DOS 3.3 $DE/$AA to $DF/$AA. Third, there is a nibble-like checking routine to check track 0.

Breaking the nibble-count scheme is the most difficult task of all.

The following procedure will unlock this program. It also works for DB's Utilities disk:

1) Put in the System Master and type:

LOAD COPYA

2) Add the following lines to COPYA:

```
199  GOSUB 400
248  GOSUB 420
259  GOSUB 420
400  POKE 47413,223: POKE 47423,1
     71: POKE 47505,223: POKE 475
     15,171
405  POKE 48351,201: POKE 48352,1
     2: POKE 48353,105: POKE 4835
     4,0: POKE 48355,24: POKE 483
     56,76: POKE 48357,107: POKE
     48358,190
410  POKE 48741,223: POKE 48742,1
     88: RETURN
420  POKE 48741,107: POKE 48742,1
     90: POKE 47413,222: POKE 474
     23,170: POKE 47505,222: POKE
     47515,170
425  POKE 48741,107: POKE 48742,1
     90: RETURN
```

3) Save the new COPYA in case of errors.

SAVE COPYADB

4) Execute the program

RUN

5) After the disk has been copied, use the Inspector, or some other sector editor to read and modify the following sectors:

Trk	Sct	Byte	From	To
0	3	$35	DF	DE
0	3	$3F	AB	AA
0	3	$91	DF	DE
0	3	$9B	AB	AA
0	E	$0A	A2	D0
0	E	$0B	00	12
1	F	$C7	A9	60
3	1	$3E	20	60

6) Write-protect disk before running.

The above procedure eliminates all three of the protection schemes Stoneware provided on the older version of the excellent DB Master.

Backing-Up Mouskattack

By Clay Harrell

Mouskattack
On Line Systems
36575 Mudge Ranch Rd
Coarsegoald, CA 93614

Requirements:
48K Apple
One disk drive, with DOS 3.3
DOS 3.3 System Master
Mouskattack
One blank, initialized 48K slave disk

Although Mouskattack is a rather old and not so thrilling maze game, it warrants discussion on deprotection methods and the use of DOS from protected programs.

Upon booting the Mouskattack disk, the prompt appears on the lower left side of the screen, indicating that a somewhat normal DOS is used by the program.

If you boot a normal DOS 3.3 disk, then put your Mouskattack disk in the drive and type CATALOG, a director does appear. You will not see any files, just copyright notices and names of the authors involved.

These, however, are files in the directory.

The disk seems almost unprotected and to confirm this, I made a copy with COPYA from the DOS 3.3 System Master. It reads the non-DOS tracks slowly. This is due to the sector skewing used by On-Line in hopes of a faster loading game and not due to the protection. However, the people at On-Line were not too successful in carrying out their intention and we shall see why, in a moment.

The first three tracks (the DOS tracks) read at the normal speed because they are normal DOS, just like those in your DOS 3.3 System Master.

After making my COPYA copy, I

used a Disk Editor to read in track 11, sector F, of the disk. This is the first sector of the catalog track.

On-Line was able to make the directory appear without file types and sector lengths using a simple technique. The first seven (or six) characters in the directory name are back spaces. Starting with byte $0E, change the first seven characters to anything but control characters, numbers or spaces, and you can load and examine the files like any other DOS 3.3 files.

Don't bother with the other file names, though. They are all blank files. The only one we are interested in is the first directory entry, MOUSKATTACK.

Snooping Through

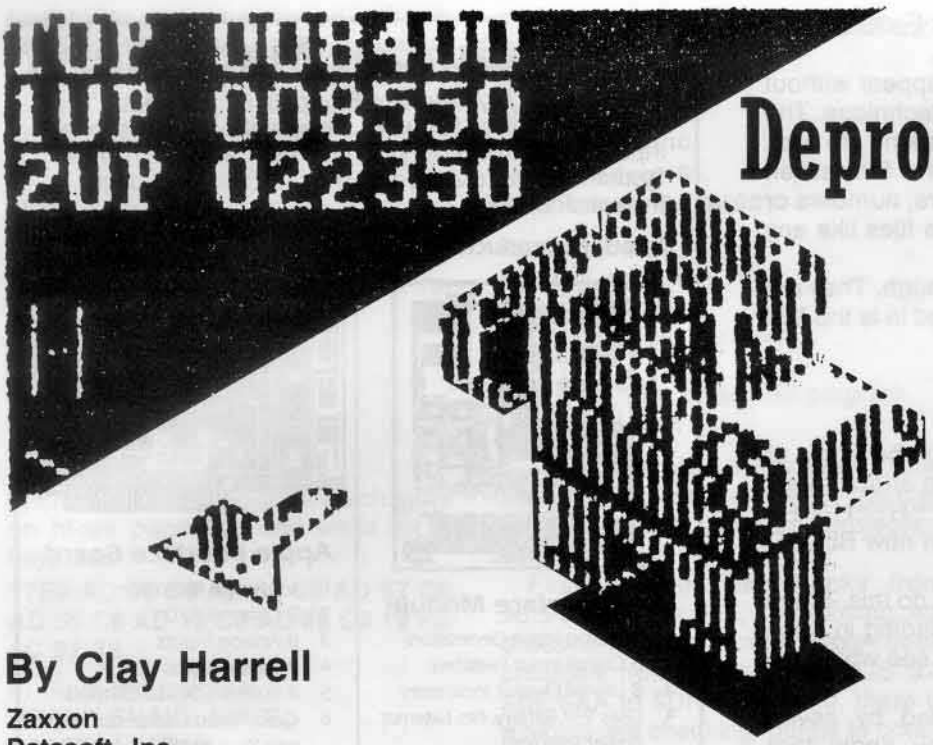
Now, get back into BASIC and catalog the copy of Mouskattack. The first entry of your catalog should have a binary file, four sectors in length, called CTRL-H CTRL-H CTRL-H CTRL-H CTRL-H CTRL-H CTRL-H CTRL-H MOUSKATTACK. You can now BLOAD this file and snoop through it.

At this, point you might ask how I knew to do this. Since Mouskattack has a normal DOS on it, by loading in track one, sector 9, with a sector editor, you can see which file is the boot file or Hello program.

Sure enough, Mouskattack is preceeded by seven CTRL-H's (i.e., backspaces). Therefore, you know that Mouskattack is the first file you should snoop through.

Now, to find out where this file is loaded, BLOAD the file CTRL-H CTRL-H CTRL-H CTRL-H CTRL-H CTRL-H CTRL-H CTRL-H MOUSKATTACK, enter the monitor and examine locations AA72.AA73.

Continued on page 30



Deprotecting Zaxxon

By Clay Harrell

Zaxxon
Datasoft, Inc.
16606 Schoenborn Street
Sepulveda, CA 91348
$39.95

Requirements:
48K Apple or Apple II Plus
One disk drive and DOS 3.3
COPYA from 3.3 System Master
A sector editor such as Zap
Two blank disks
Original Zaxxon disk

One of my favorite arcade games of all time is Zaxxon. The game was the first to use "true" three-dimensional graphic effects that played well. I've spent a great deal of time (and tokens) playing the game so naturally, when it was available for the Apple, I bought it.

The only complaint I have about Zaxxon is that there are only three planes given for each game. The arcade Zaxxon I had always played had five. I felt cheated! With this in mind, I decided to dig into the code that made Zaxxon fly.

(Please note that there has been a new release of Zaxxon from Datasoft. This is the one I will be primarily discussing, although the same protection was used on both versions. You can tell if you have the new version because it has an option to use the Mockingboard with it. If your Zaxxon boots and asks "Mockingboard in slot 4 Y/N?," you have the new version. In addition, I have seen two different protections used on the newer versions of Zaxxon. I will give details on the deprotection of all versions, including the older version of the game).

The first thing to notice is the boot of Zaxxon. Listen to your disk drive as the game boots and you can hear the drive arm swing out to an outside track and then swing back in and read the game in. This is what is loosely known as a "nibble count" or a "checksum" routine.

If a byte doesn't match the benchmark like it should when the outside track is read, the game will clear memory and reboot. Usually, the only thing involved in deprotecting a program that isn't a single load and that has a "nibble count" is to find the routine that does the check and jump around it. Usually, too, about the only way to find this routine is to (yech!) trace the boot.

Boot Code Tracing

Boot code tracing is a method of tracing how a program gets from your disk to memory. It does not magically happen all at once but in stages which we can trace and examine and, hopefully, understand. Hence, the name "boot code tracing."

The theory of boot code tracing suggests that you should follow the boot process one step at a time to see where it takes you, by altering the code to prevent it from running away from you. Yes, it is advisable that you understand assembly language, since the code that boots the disk is, in many cases, intentionally misleading.

This process is based upon the law that track zero, sector zero, must always be read by the disk controller card into page 8 ($800-8FF) of memory. After this, depending on the complexity of the protection, it is sometimes difficult to understand what goes on in the rest of the load.

In a normal slave disk boot, there are three stages of a boot, starting with the code at $C600-C6FF in the disk controller card. This controller card code loads in track $0, sector $0, into $800-$8FF, which, in turn, loads in track $0, sector $0 through sector $9, into $B600-BFFF. This new code loads in track $0, sector $C through track $2, sector $4, into $9D00-$B5FF and finally your Hello program is run. In summary:

Stage #	Code location	Final location	Jumps to
0	$C600-C6FF	$0800-08FF	$0801
1	$0801-08FF	$B600-BFFF	$B700
2	$B600-BFFF	$9D00-B5FF	RUN

Now, in order to change the code in

the boot so that it doesn't run away from us, we can either alter memory or alter (a copy of) the disk.

Check Routines

If you defeat the DOS (Disk Operating System) error-checking routines, you can copy the Zaxxon disk with COPYA. Of course, it won't run because of the "check" routine and because of some of the other incompatibilities with normal DOS 3.3, but you can read and write to the copy easily.

To defeat the error-checking, enter the monitor

CALL-151

and type:

B942:18

Whenever DOS encounters an error, it jumps to a routine at $B942 which sets the carry bit and returns. The carry bit is a flag to DOS that there was an error and directs it to stop whatever it was doing. It then prints out a worthless message to the user.

If you defeat this routine, you can fool DOS and read the entire Zaxxon disk. Then copy the disk and examine the data on it. So, put your DOS 3.3 System Master in the drive and

RUN COPYA

When the prompt for source drive is asked, hit CTRL C. This will break you into BASIC. Now, type the following:

CALL-151
3A1:18
302:17
35F:17
3D0G
70
RUN

This will make some changes to COPYA.OBJ0 so that it ignores errors and only copies tracks $0-$16. Line 70 of the Applesoft part of COPYA is also deleted to prevent loading in COPYA.OBJ0 and writing over the change just made. Now, select the desired drives and copy Zaxxon.

Note: Zaxxon only lives from track $0 to track $16 (tracks $0-$13 on some versions). If COPYA tries to read an unformatted track your drive will recalibrate ("grind") for every sector that the program tries to read. Do not interrupt this process. After the copy is made, make another copy of the duplicate of Zaxxon, which you had just made. Label the first copy WORK ZAX and the second copy COPYA ZAX.

Put these two copies aside for a moment. The next step is to trace the boot. First, enter the monitor

CALL-151

and clear out memory with this command:

800:0 N 801<800.95FFM

To start the boot you need to run the code in the disk controller card, but we stop it before it runs away. You cannot change the code in ROM (Read Only Memory), but you can copy it down to RAM (Random Access Memory) and change it. Use the following command from monitor to do this:

9600<C600.C6FFM

Having done this, you can change the boot code so that it loads in track $0, sector $0, but does not execute it. At location $96F8 you will see a JMP $801. This starts the next boot process (refer to table).

Next, change this to JMP $FF59, which will jump to the monitor. From the monitor, type:

96F8:4C 59 FF

Now, put the WORK ZAX disk in drive one and execute the code with:

9600G

The drive will spin, the Apple will beep and you will see the "*" prompt.

From the table, you can see that this code is loaded into $801. (To stop the drive from spinning, enter **C0E8** from the monitor. Also, to turn off the hi-res page, enter **C054** and **C051**). Now, examine the code at $801 with the command:

801L

Upon examining the code, you will find that it is fairly normal, with some exceptions:

```
0801- A5 27     LDA  $27       !------
0803- C9 09     CMP  #$09      !
0805- D0 18     BNE  $081F     !
0807- A5 2B     LDA  $2B       !  N
0809- 4A        LSR            !
080A- 4A        LSR            !  O
080B- 4A        LSR            !
080C- 4A        LSR            !  R
080D- 09 C0     ORA  #$C0      !
080F- 85 3F     STA  $3F       !  M
0811- A9 5C     LDA  #$5C      !
0813- 85 3E     STA  $3E       !  A
0815- 18        CLC            !
0816- AD FE 08  LDA  $08FE     !  L
0819- 6D FF 08  ADC  $08FF     !
081C- 8D FE 08  STA  $08FE     !
081F- AE FF 08  LDX  $08FF     !
0822- 30 15     BMI  $0839     !  D
0824- BD 4D 08  LDA  $084D,X   !
0827- 85 3D     STA  $3D       !  O
0829- CE FF 08  DEC  $08FF     !
082C- AD FE 08  LDA  $08FE     !  S
082F- 85 27     STA  $27       !
0831- CE FE 08  DEC  $08FE     !
0834- A6 2B     LDX  $2B       !
0836- 6C 3E 00  JMP  ($003E)   !
0839- EE FE 08  INC  $08FE     !------
083C- AD 55 C0  LDA  $C055     !TURN ON
083F- AD 50 C0  LDA  $C050     !HI-RES2
0842- AD 57 C0  LDA  $C057     !------
0845- A2 7D     LDX  #$7D      !???????
0847- 9A        TXS            !???????
0848- 4C B4 08  JMP  $08B4     !------
        :  :  :                !
        :  :  :                !
        :  :  :                !
```

```
08B4- A9 20     LDA  #$20     !
08B6- 85 1B     STA  $1B      !
08B8- AD 52 C0  LDA  $C052    !
08BB- A9 60     LDA  #$60     !
08BD- 8D 06 07  STA  $706     !-------
08C0- 6C F8 08  JMP  ($08FD)  !JUMP TO
                              !$8000
                              !-------
```

How it Works

The first part of the code, from $801 to $83B, is lifted verbatim from a DOS 3.3 Slave disk. This code loads in track $0, sector $0, through track $0, sector $09, into $7F00-$88FF. This is revealed from location $8FE, which is one higher than the first page loaded into. The byte at $8FF is one less than the number of sectors to be loaded.

The next piece of code turns on the hi-res screen. The last part of code, before the jump to $8B4, looks innocent, but really isn't. It loads the X-register with $7D and transfers it to the stack pointer.

To understand the implications of this, you must understand how the computer keeps track of its return position after an RTS (Return from Subroutine).

When the 6502 encounters a JSR (Jump to Sub-Routine), it stores the present address on the stack so it knows where to return when an RTS is encountered.

This can be used to obscure code from unwanted eyes. For example, say we want to go to $9600. You can load the stack with $95 and then $FF, by using the PHA op-code (Push Accumulator on Stack). When an RTS is encountered, the two last bytes are pulled from the stack, incremented by one (to $9600), and jumped to.

Alternatively, you can change where the pointer on the stack is pointing to and make it point to the desired location. Keep this in mind when you find an RTS.

At $8B4, a few zero page locations are loaded and then there is an indirect jump to $8000 (through $8FD). To see the code at $8000 load the next stage of the boot but stop it before it can execute. To do this, run your sector editor and change track $0, sector $0, byte $C0, to 4C 59 FF on the disk labeled WORK ZAX.

This will jump you to the monitor before it can execute the code at $8000. Now, write the sector back out to the disk and boot the disk. After a moment the computer will beep and stop in the monitor. You can now examine the code at $8000. Do this with the command:

8000L

and the code will look like this:

```
8000- A0 09     LDY  #$09     !-------
8002- A2 00     LDX  #$00     !
8004- 8A        TXA          ! M
8005- EE 0D 80  INC  $800D   ! E  M
8008- EE 10 80  INC  $8010   ! M  O
800B- BD 18 7F  LDA  $7F18,X ! O  V
800E- 9D 7E 00  STA  $007E,X ! R  E
8011- CA        DEX          ! Y
8012- D0 F9     BNE  $800B   !
8014- 88        DEY          !
8015- D0 EE     BNE  $8005   !-------
8017- 60        RTS          !JUMP
                             !THRU
                             !STACK
                             !-------
```

Notice the RTS at $8017. Remember: you jumped (JMP), not jumped sub-routine (JSR) to all the routines, so there is nothing on the stack to return to!

Well, yes there is! The memory move relocates the memory at $7F19 through $8718, down to $7F through $087E. This moves memory across page $1, which is the stack!

Remember too, that the stack pointer is set to $7D in Boot1. After the memory move, an RTS is executed. The stack is pointing at $17D, which is now $07 and $65 after the memory move. This will be the next jump (plus one) for the final stage of the boot!

The manufacturer, Datasoft, has added a final bit of protection in that the next jump is across the text page which, of course, changes when you exit the program in any manner. But you can simply move memory to, say, $107E and examine it to see what the next load does.

To do this, change $8010 to 10 and execute the memory move. Do this as follows:

8010:10 N 8000G

Now, $1766 is equivalent to $766. So type:

1766L

and examine the code. It should look as follows:

```
1766- A5 2B     LDA  $2B     !
1768- 8D 0E 02  STA  $020E   ! G  C
176B- A9 20     LDA  #$20    ! O  O
176D- EA        NOP          ! O  D
176E- A6 2B     LDX  $2B     ! D  E
1770- 20 1F 02  JSR  $021F   !-------
1773- A0 20     LDY  #$20    !
                             !NIBBLE
                             !COUNT/
                             !CHECK
                             !-------
17D4- A9 16     LDA  #$16    ! G  C
17D6- 8D 11 02  STA  $0211   ! O  O
17D9- A9 D0     LDA  #$D0    ! O  D
17DB- 8D 16 02  STA  $0216   ! D  E
17DE- 4C 9A 01  JMP  $019A   !
                             !-------
```

If you take time to look at this code you will find that the offending "nibble count" or "checksum" starts at $1773 and goes to $17D3! This is all you need to know to defeat it.

What you have to do, next, is jump around this. Do it by typing:

1773:4C D4 07

The other loads and stores of the accumulator are parameters for their loader. For further understanding, here are the parameters:

$211-high track # to start from.
$212-high sector # to start from.
$21E-# of pages (sectors) to load.
$21B-starting page to load at.
JSR $1E9-start the load.

The last thing to do is to find the "nibble count" code on the disk code and change it. Most good sector editors (like The Inspector) have LOCATE routines enabling you to find a pattern of bytes on a disk.

One final note: Now that you know where the check routine lives, we can defeat it. But you also must change the epilog bytes to normal DOS 3.3.

Datasoft uses $CC for their epilog bytes where normal DOS uses $DE. If you are familiar with loaders and RWTS, you can find it in their loader. The table is on page $4 (page $14).

Stepwise

With this information in hand, here is a step-by-step procedure for the deprotection of Zaxxon (Mockingboard version only):

1) Get out your DOS 3.3 System Master disk and run it

RUN COPYA

2) When COPYA asks for the slot number of the original disk, stop the program

CTRL-C

3) Enter the monitor

CALL 151

4) Patch DOS and COPY.OBJ0 so that they ignore read errors and only tracks $0-$16 are copied

B942:18
3A1:18
302:17
35F:17

5) Re-enter Applesoft and delete line 70 of COPYA so that COPY.OBJ0 is not reloaded

3D0G
70

6) Run COPYA to copy Zaxxon

RUN

7) If you have two disk drives you can leave the room during the copy

process; otherwise, you may have to put up with your drive making some horrible grinding noises. *Don't worry. This will not harm the disks or your drives.*

8) Run your sector editor and make the following changes depending on which version of Zaxxon you have. (Note: you may have to try all of these changes, depending on the release date of your Zaxxon. One of them should work, though):

For Mockingboard Versions of Zaxxon

Trk	Sect	Byte	From	To
$00	$04	$4F	$CC	$DE
$00	$04	$50	$D0	$EA
$00	$04	$51	$AE	$EA
$00	$07	$0D	$A0	$4C
$00	$07	$0E	$20	$D4
$00	$07	$0F	$84	$07

For Mockingboard Versions of Zaxxon, for which the above method does not work

Trk	Sect	Byte	From	To
$00	$07	$00	$A9	$4C
$00	$07	$01	$01	$C0
$00	$07	$02	$48	$08
$00	$04	$4F	$CC	$DE

For the older non-Mockingboard Versions of Zaxxon

Trk	Sect	Byte	From	To
$00	$07	$00	$A9	$4C
$00	$07	$01	$01	$C0
$00	$07	$02	$48	$08
$00	$04	$4F	$CC	$DE

9) Don't forget to write the sectors you have altered back to the deprotected Zaxxon disk.

APT for Zaxxon

As I said earlier, the Zaxxon in the arcades gives you five planes and our Zaxxon only three! If you want more planes, change byte $17 on track $09, sector $08 with your sector editor to the number of planes you want (between $00 and $FF). I chose to change this byte to $03. This gave me four planes, which is a nice compromise. This modification applies to all versions of Zaxxon.

Foreign Subscribers: Change your current subscription rates to surface rather than air and save a substantial amount in shipping costs. Please allow 4 to 6 weeks for change of subscription to begin.

Yes! I want to change my current airmail to a surface mail subscription.
Please type or print legibly.

Name _____

Address _____

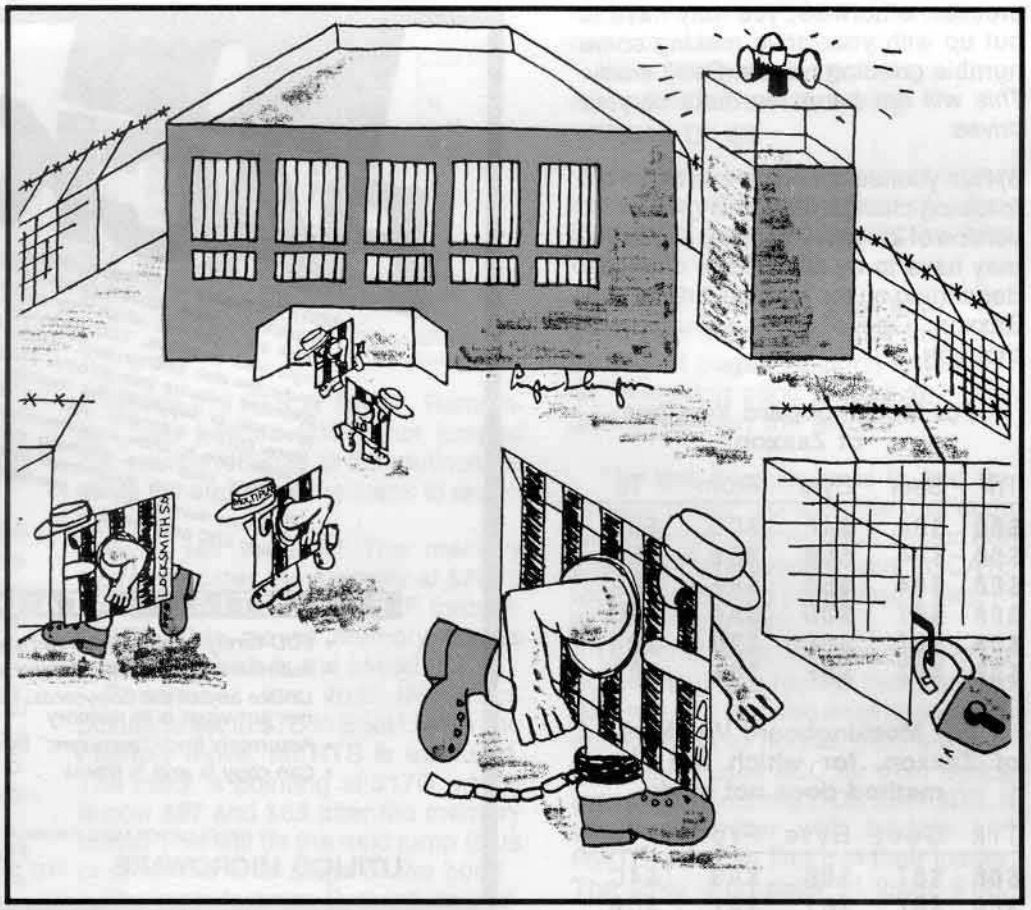
City _____ State _____

Country _____

To change your airmail subscription rates to surface mail, fill out this form and return to:

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

X



Making Liberated Backups That Retain Their Copy-Protection

By Thomas Dragon

The following is a reasonably efficient procedure for backing-up certain protected disks that are not easily copied by either Locksmith 4.1 or Locksmith 5.0. The "rub" is that you must retain a certain portion of the copy protection. That is, your copies can't be duplicated by Locksmith or other bit copy programs. However, it does provide a procedure for backing-up the disks as well as "liberating" them for study.

Freeing the Disks

STEP(1) First, you must have an APPLE II with 48K. It is especially convenient if you have an old INTEGER card living in your APPLE, but if you don't, this procedure will still work on a number of protected commercial software programs. Let us assume that you have an old INTEGER card. I'll get back to APPLESOFT later. The switch on the INTEGER card should be down so that APPLESOFT is the boot-up language. What's this ? You say that you don't use your INTEGER card anymore because you have an expansion card in slot 0? Well, clean that old INTEGER card off because, for our purposes, it will work in any slot which allows the switch to stick out of the back of the APPLE (slot 0, 2 or 4).

STEP(2) Now, pull out the old DOS 3.3 System Master and boot up your APPLE

PR#6.

Place a blank disk into the drive and INITialize it.

INIT HELLO

DELETE the "HELLO" program after you have INITialized the disk.

DELETE HELLO

This slave disk will serve a very special purpose. Later, we will want to boot the system with it. Unlike a system master, when booting with a slave disk, much of the APPLE memory ($4000-$9600) is left undisturbed.

STEP(3-a) Boot the protected disk and watch the screen very carefully during the boot process. If you see the APPLESOFT cursor for even a fraction of a second the chances are very good that this procedure will liberate the disk. Flip the INTEGER switch up after the boot process is complete. Press RESET and you are instantly into the monitor. Note that you will always jump into the monitor no matter where the INTEGER card is located. This is another well kept APPLE secret.

STEP(3-b) If you don't have an INTEGER card, make like Woody Woodpecker on the RESET key until the disk drive stops. Then enter CALL -151, and you should be in the monitor. If the drive doesn't stop after 10 or 20 rapid RESETs, then this procedure won't work on your APPLE. You might want to see if there is some old-timer who might be willing to sell his useless old INTEGER card for $15 or $25. There are a number of old INTEGER cards around that appear not to work. Surprisingly, many of these old cards will still work in terms of forcing a jump to the monitor. If the old timer has one that doesn't work, just ask him if you can have it to study.

STEP(4) Now that you are in the monitor, type **A56EG**. This will activate the CATALOG function if the protected disk has a reasonably intact DOS. You should then see a CATALOG of the disk contents. If you only get a "honking", then this procedure won't work. Don't despair. You will very seldom get a honking if you see the APPLESOFT prompt when you boot.

STEP(5) Next, we will move part of the protected RWTS into an area of the APPLE memory that will remain intact during the boot process. In order to accomplish this, we will use the monitor memory move. Enter

4800<B800.BFFFM

This will move the special DOS routines to $4800.

STEP(6) Remove the protected disk from the drive and replace it with the slave disk that you previously initialized.

Flip the INTEGER card switch down if you are using an INTEGER card and boot the slave disk by typing **6CTRLP** (assuming slot 6 is the boot slot). Don't worry about the "FILE NOT FOUND" error. At this point, I am always extra cautious because I don't want to accidentally lose the protected DOS information. Take out any disk with a bit of space on it and enter the following command from the keyboard

BSAVE PROTECTED DOS, A$4800,L$800

STEP(7) Place the System Master into the drive and enter **BLOAD FID.** Everyone reading this little note must know about FID. If you don't, let me just say that it is a DOS-less copy program. It does not carry its own DOS like most other variations of APPLE copy routines, but "sucks" up the DOS that is living in memory when it is activated. However, we have not activated FID. We have only loaded it into memory.

STEP(8) It is now time to move the protected DOS back into the regular DOS 3.3 memory area. Enter CALL -151 to get into the monitor. Then enter

B800<4800.4FFFM

Now you have modified DOS 3.3. The modified DOS is neither regular DOS 3.3 nor the true protected DOS. It is a hybrid of regular DOS 3.3 that employs the protection scheme that is on the protected disk.

STEP(9-a) Return to Applesoft by typing CTRLC. Now INITialize a slave disk by typing **INIT HELLO.** If all has gone well, the slave disk will now become a slave disk that retains the copy-protection of the original protected disk. Be sure to **DELETE HELLO** after the drive stops.

STEP(9-b) If you have made an error in moving DOS, just re-boot the slave disk and then go back to STEP (7). If you have somehow managed to destroy the memory image, re-boot the system and **BLOAD PROTECTED DOS** as saved in STEP (6) and then go to STEP (7).

STEP(10) The final step is to activate FID. You have the protected disk and you have a blank disk with the same DOS. Either enter **CALL 2051** or **CALL -151 803G**. Either CALL will essentially activate FID and allow it to use the protected DOS.

STEP(11) From this point, you should be able to transfer the files from the protected disk to your special slave with the use of FID.

The programs are now essentially liberated from the point of view that they can be listed, modified and saved. It may be that the protected "HELLO" program sets the RESET vector when it is run. If you wish to study the programs, simply load and look at them. To be sure, you should place a write-protect notch on your copied disk if the original copy was write-protected. The disks are still protected in the sense that any copy procedures that failed with the original disk will also fail with the liberated disk.

This procedure works well with the Microzine disks, the Videx pre-boot disks, Micro Power and Light disks, HIRES Secrets from Avant Garde, and an incredible collection of other disks that have either binary or APPLESOFT programs on them. Disks on which the procedure doesn't work include the Human Systems Dynamics (HSD) disks and E-Z Draw. It doesn't work for E-Z Draw because the E-Z Draw disk uses a 32K DOS and this discussion deals with a 48K DOS. It doesn't work for the HSD disks because they have an incredibly complex protection system.

REVIEW: S-C ASSEMBLER

Whether you are a beginning BASIC programmer or an experienced assembly language programmer, sooner or later you'll need an assembler in your Apple programming toolchest. S-C Software's Macro Assembler version 1.1 is one of the most versatile and easiest to use.

System Requirements

The S-C Macro Assembler requires an Apple][,][+, or][e; at least 32K RAM; and a single disk drive. It will work with the Apple's standard 40-column display, as well as the][e, STB and Videx 80-column boards. The assembler may be loaded into a language card, substantially increasing room for assembly language programs.

Supplied on an un-protected DOS 3.3 formatted diskette, the S-C Assembler can be backed up easily. The source code for the assembler is also provided, so experienced programmers can modify the assembler to their liking (or their highly kludged machines). Purchases may be made directly from S-C Software Corporation, 2331 Gus Thomasson Road, Suite 125, P.O. Box 280300, Dallas, TX, 75228 (214) 324-2050. The cost is $92.50. An excellent by-subscription newsletter for assembly language programmers, "Apple Assembly Line" is also offered by S-C Software. Subscriptions are $18 annually in the U.S.

What Is An Assembler?

High level languages such as BASIC and Pascal aren't a computer's "native"language. When executing a high level language, the computer must convert the program's IF THENs and FOR NEXTs into something it can understand. Conversion into the machine's native language takes time; the high level language statements occupy large amounts of the computer's memory and can only do a limited number of things. (How, for example, can you play music from BASIC?).

" Whether you are a beginning BASIC programmer or an experienced assembly language programmer, sooner or later you'll need an assembler in your Apple programming toolchest."

You know, however, that the Apple can make sounds; you've games that beep, zip, bang and chirp. Music, fancy screen scrolling, faster DOS, and many other things are all made possible through the use of the Apple's machine language.

Unfortunately, Apple machine language is a bunch of hexadecimal bytes (base 16 numbers between 0 and 255). To most people "20 9C FD 20 A3 FC 60" doesn't even vaguely resemble a program which will print a line of dashes across the screen, but that is what it will do. Assembly language takes commands only humans can understand, called mnemonics (neh- monics), and converts them into the hex bytes of machine language.

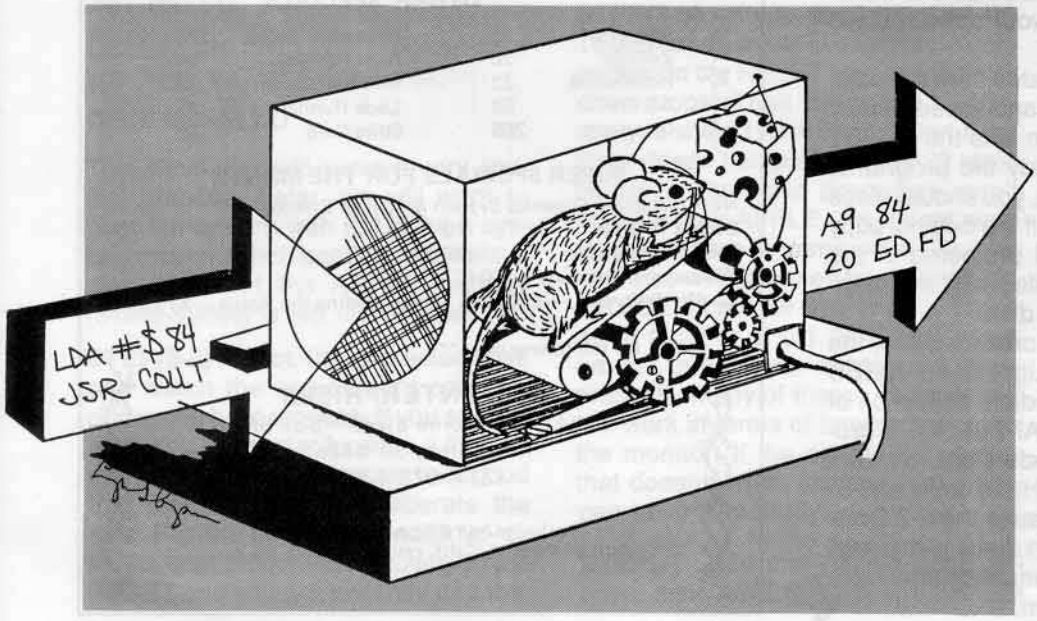
Creating machine executable programs in assembly language is a two step process. First, you type in the source program. Then, the assembler converts it into the hexadecimal bytes of machine language called the object code or object program.

Editing Features

The S-C Macro Assembler uses a line based text editor, as opposed to some word processors which are screen based. One feature inherent in the S-C Macro Assembler which I find particularly convenient is that all the standard ESC codes are preserved. These commands for moving the cursor and clearing the screen should already be familiar to you from BASIC or Monitor programming. Because the source code for the S-C Macro Assembler is provided, these ESC functions can be modified and customized (ala GPLE) by reasonably experienced programmers. In addition, all DOS commands may be entered normally. A lead-in character such as CTRL D is required with some other assemblers.

Twenty-nine commands, including global search and replace, are available during program editing (see fig. 1). All of the commands may be abbreviated as short as their first three letters. Sufficient parameters are available to make these commands very flexible.

Implementations of the commands are in many cases the same as in BASIC. For example, line ranges for commands may be written with a leading



or trailing comma. The command **DEL 2000,** deletes all lines of source code beginning with line 2000; **LIST ,1500** lists all lines through line number 1500.

Whenever the assembler is sending output (such as listing or assembling programs) you can temporarily halt it by pressing any key. Pressing RETURN aborts the listing.

If you load the assembler into a language card, it will look like Integer BASIC to DOS. You can temporarily exit from the assembler and run a program in Applesoft or machine language. By typing INT you can return to assembly language. However, before you leave the assembler, you should save your source program or it will be deleted when you try to return.

One problem is that source programs will be stored to disk as type "I" files. DOS cannot distinguish these from genuine Integer BASIC programs. Trying to load an integer BASIC program while in the S-C Macro Assembler will be confusing to both you and your computer. This is a minor problem which can be overcome by any reasonably crafty person. Two solutions have occured to me. One, is the addition of filename extensions (as in CP/M) to my source files. The extension .ASM is short, descriptive, and consistent with CP/M's conventions. The other solution is to keep source code files and Integer BASIC programs on separate disks.

You may also opt to have your source code stored as a standard DOS text file. The S-C Macro Assembler's text file capability allows compatibility with other assemblers, disassemblers, and text editors. To read a standard text file into the S-C Assembler you simply turn on the assembler AUTO numbering feature and then EXEC the desired text file into the editor.

Assembler Directives

When first learning assembly language, I found the distinction between the standard 6502 mnemonics and the assembler directives, or psuedo-ops, terribly confusing. It has always puzzled me why the writers of assemblers try to make their assembler's directives look like the 6502 mnemonics. After all, assembler directives only tell the assembler what to do and aren't really part of your program.

All of the S-C directives consist of two or more letters preceded by a period. This makes them stand out in a listing and in the programmer's mind. There are 25 assembler directives available from the S-C Macro Assembler (see fig.2).

When taken together, the S-C directives allow many advanced options. Included are conditional assembly, assembly direct to and from disk, multi disk operation, and the titling of assembly listings.

Program Documentation

S-C Software provides a handy reference card and 100 pages of documentation in a three-ring binder for their assembler. I found the documentation a bit too technical for those totally unfamiliar with assembly language programming. It will not teach you assembly language, but will teach you how to use the assembler.

The manual, which is indexed and has a table of contents, starts with a brief introduction. Next, is a four page tutorial to get you started. The largest part of the manual is occupied by sections on use of the assembler, in reference form. Also included are examinations of memory usage and customizing of the assembler, 6502 programming, and Sweet-16.

Special Features

Besides all of the usual features found in assemblers, the S-C Macro Assembler provides many special enhancements.

If you've ever looked at a listing of an assembly language program you'll see that there are five columns of information. From left to right are the line number, label, mnemonic, operand, and comment fields. The S-C Assembler lets you automatically tab between fields by using control-I (the TAB key on the][e).

The line numbers are used exclusively by the assembler. If you wish to see a range of lines or delete a line, use the lines' numbers. You may not branch from within assembly language by using line numbers, as is the case in BASIC. Instead, branches are made to labels which are stored in the label field. To print a character, you would not GOSUB 1000, but JSR (the mnemonic for jump to subroutine) COUT.

If you have even programmed in assembly language using another assembler you may have encountered the problem of coming up with a unique name for each label. Within a single subroutine you may need to take several internal branches, each of which must be defined by its own label. For instance, not only is the label COUT used, but also COUT1, COUT2, etc. This is avoided by the S-C Macro Assembler through the use of local labels. Local labels are only defined relative to the preceding normal label. A local label is a number between 0 and 99 preceded by a period. Thus, a label named .1 could be defined several

times within a single program.

A comment in assembly language is like a remark in BASIC. The S-C Macro assembler doesn't require a semicolon or anything else before a comment if you put it in the comment field. Full line comments are available if you precede them with an asterisk or semi-colon.

Macros are single instructions which, when assembled, are replaced by a series of instructions you have defined. They can replace commonly used sequences of instructions, can be used as alternate mnemonics for 6502 instructions, etc. As you may have guessed, the S-C Macro Assembler also supports macros.

The S-C Macro Assembler's use is not confined to the assembly of standard 6502 Mnemonics. Besides its capability to modify mnemonics using macros, it can assemble Steve Wozniak's Sweet-16. S-C Software also offers a line of cross assemblers which use the S-C Macro Assembler standard source editing features. When used with the Macro assembler these allow you and your Apple to write code for other microprocessors, from the 65C02 to the 68000. The cross assemblers range in price from $20 to $50.

Additionally, there are several other products available through S-C Software which include a full-screen editor and a disassembler which are compatible with the S-C Macro Assembler.

Conclusions

The S-C Macro Assembler is a state of the art assembler which is easy to use, versatile, and appropriate for persons with all levels of programming knowledge. Of the assemblers I have used, I found the S-C Macro Assembler to be the best assembler for the Apple. S-C Software provides adequate support, and their policy of not copy-protecting their products is admirable.

FIGURE 1

NEW- Deletes current source program and restarts S-C Macro Assembler.

LOAD- Loads a source program from cassette or disk.

SAVE- Saves current source pro gram to cassette or disk.

TEXT- Saves current source program to disk as a normal sequential text file.

HIDE- Hides source program in memory so that is can be MERGEd with another source file.

MERGE- Joins a source program with one which has been hidden by HIDE command.

RESTORE- Recovers a source program if assembly is aborted while inside of an .IN module (see Figure 2).

Editing Commands

LIST- Displays source program lines within a specified range or those containing a search string.

FIND- Same as LIST.

EDIT- Allows editing of source program lines. Edit displays the line to be edited and allows 15 control commands for moving the cursor, deleting and inserting text.

REPLACE- Universal search and replace, verify or automatic replacement.
Wildcards are allowed.

DELETE- Deletes a range of program lines.

RENUMBER- Renumbers source program.

COPY- Copies one or more lines from one place to another.

Listing Commands

FAST- Sets listing speed to normal rate.

SLOW- Sets listing speed slow enough to read.

PRT- Jumps to a printer driver routine at $10009 which must be written by the user.

'- Sends a specified string to the current output device.

Object Commands

ASM- Assembles current source program into object code.

MGO- Jumps to a memory location specified by a label or number.

VAL- Evaluates an expression and prints its value in hexadecimal.

SYMBOLS- Displays the assembler's symbol table (all of the labels your program uses).

Miscellaneous Commands

AUTO- Turns on automatic line numBERing.

MANUAL- Turns off automatic line numbering.

INCREMENT- Sets the increment used by automatic line numbering.

MEMORY- Displays your program's current location in memory.

MNTR- Puts you in the Apple's system monitor.

RST- Lets you specify the address to which the Apple will jump when the RESET key is pressed.

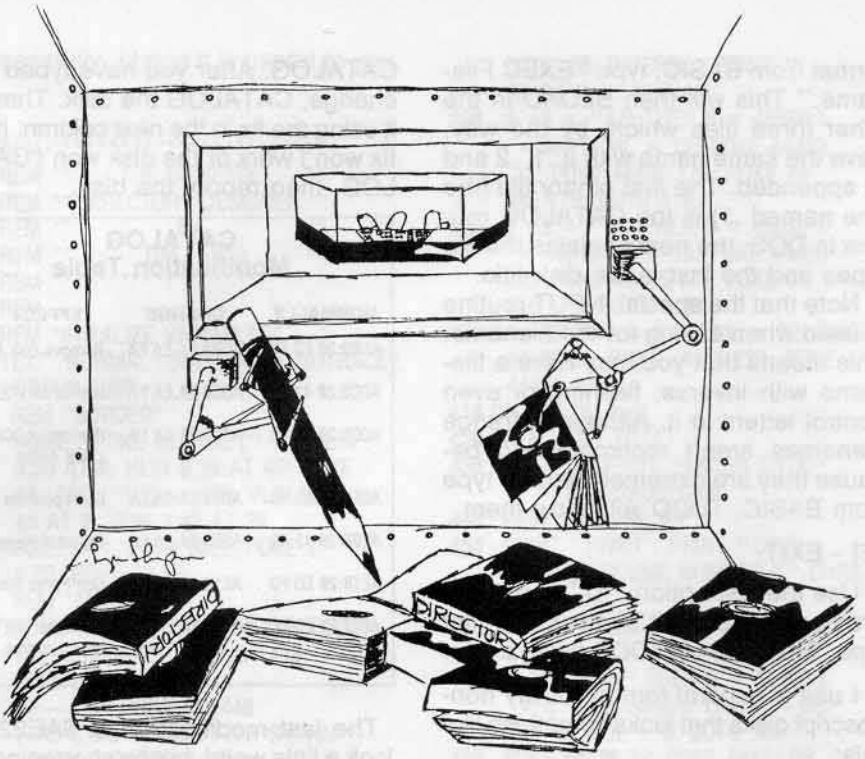
USR- Jumps to $1006, where you can place a machine language program or subroutine.

FIGURE 2

S-C Assembler Directives

.OR	Origin
.TA	Target address
.TF	Target file
.IN	Include file in assembly
.EN	End of program
.EQ	Equate
.SE	Re-definable symbols
.HS	Hex string
.DA	Data
.AS	ASCII string
.AT	ASCII terminated string
.BS	Block storage
.LIST	Controls assembly listing (ON or OFF)
.TI	Title
.US	User defined
.PG	Page eject
.DO	Conditional assembly
.ELSE	Conditional assembly
.FIN	Conditional assembly
.DUMMY	Begin a dummy section
.ED	End dummy section
.PH	Start phase
.EP	End phase
.MA	Macro definition
.EM	End macro

Reviewed by Jeff Thomas



The Disk Directory Designer

By Tim Lewis

Requirements:
48K Apple][plus or][e
DOS 3.3

The Disk Directory Designer (TDDD) is a utility designed to aid you in the creation of a new format for your CATALOG. Using this utility, you can change the message "DISK VOLUME" to any title you want up to 32 characters long, with normal, inverse and flashing letters in it. You can also redefine the standard file type codes (T, I, A, and B) to any character you want, and do the same for the lock and unlock characters. Then you can have your CATALOG displayed in 1, 2, 4, or 5 columns. Your work may then be stored on disk for later use. This program can be used to format each disk in the way that is most useful for you.

Entering TDDD

1) Type in the program as listed on page 20 making sure that the checksums that you have, using the program Checksoft, match the ones listed on page 22.

2) Save the program to disk.

SAVE DIRECTORY DESIGNER

3) Start the program RUNning.

RUN

How to use the Directory Designer

When the menu appears with the cursor waiting patiently at the bottom of the screen, press one of the letters in brackets (A-E) to select that option. Should you select the incorrect letter, press "ESC" or "RETURN" to exit the function.

Here is a description of each option.

[A] - CHANGE DIRECTORY TITLE

Press the letter "A" to cause a second menu to appear. This menu allows you to select the length of the title you wish to use. There are three lengths available: (A) 12 characters, (B) 16 characters, and (C) 32 characters. By pressing A, B or C, you select the maximum number of characters allowed in your title and how far the volume number (if displayed) will be from this title.

Specialized Input

This brings us to the specialized INPUT routine I developed for this program. Here is how to use it:

First notice the compression for the word normal (NRML) on the left of the greater than sign (>). As long as NRML is in effect, the characters you type are regular ones. Five control codes change the status of the characters you type. They are:

- CTRL**N** -Return to the default mode of NoRMaL.
- CTRL**I** -Characters that follow are INVerSe.
- CTRL**F** -Subsequent characters are converted to FLaSHing.
- CTRL**C** -Convert letters that follow to ConTRoL codes (displayed during the INPUT as inverse.)
- CTRL**L**- Following this, characters are converted to lowercase. If you do not have a computer equipped with lowercase, you probably shouldn't use this because it will display meaningless symbols.

You may use the left arrow key (←) and "RETURN" just like in a normal INPUT. In addition to this, characters that are usually not accessible through the Apple keyboard may be keyed in by pressing the following control codes:

- CTRL**^** – Converted to an underline ("_")
- CTRL**@**- Translated to a backslash ("/")
- CTRL SHIFT**M**- Produces a beginning bracket ("[")

This INPUT routine allows virtually every character generated by your Apple to be part of the input string.

After typing the title, the program will ask if you want the VOLUME number to be printed next to the directory title.

Once you type "Y" or "N" depending upon whether you want the volume number displayed, the program will show you the title in the form that it will appear in the directory. You can now: Return to the menu without changing the heading in memory, redo the title, or put the title you just created in memory and go to the main menu.

[B] - CHANGE DIRECTORY FORMAT

From the main menu, if you hit "B", you will be ready to change the actual directory format. As soon as you do, a secondary menu will appear and the maximum number of characters allowed in a filename will be printed at the bottom of the screen. To return to the main menu, simply press "ESC." Here is a description of the features of this secondary menu:

Option **A** will allow you to change the

lock and unlock codes. If you answer "N" to the question "DISPLAY LOCK/UNLOCK CODES ?" then the codes that are currently in memory won't even be shown in the directory. As in option A of the main menu, the previously mentioned special INPUT routine is used. Therefore a wide array of characters are available for this and the other format alterations.

By pressing "**B**", you enter the file type alteration subroutine which allows you to change the codes for the four standard (B, A, I, and T) file types in much the same way as the above LOCK/UNLOCK subroutine does. Note that a modification to the "R" type of file is not provided for.

The subroutine connected with the pressing of "**C**" merely asks you if you want the file lengths displayed in the directory. The file lengths (sometimes called sector numbers) are the three digit numbers between the file type and the filename.

Next come what I call the "funny characters." Press "**D**" to change the characters (usually they are spaces) to the left and right of the file lengths.

Last, by typing "**E**", another menu appears asking into how many columns you wish to divide the directory. Press the letter corresponding to the number of columns you wish. For those of you with long filenames such as mine, you will notice that the last characters are cut off when more than one column is selected. After every format alteration, the maximum number of characters is updated and displayed at the bottom of the format alteration menu. Note that in order to access files with names longer than the limit, you must still use the full filename.

Those of you with 80-column cards (this includes most Apple //e users) will notice that when in 80-column mode, the number of columns will be twice as many as selected (unless you choose one column). This is a result of using multiple columns. Also, for those of you with title names which use the backspace character to cover up the file length, file type, etc., this method may severely distort the appearance of your directory. This is also relative to the method used.

[C] - LOAD/SAVE FORMAT AND TITLE

If you type "C" from the main menu, the current directory format may be stored for later use or retrieved for modification. TDDD stores a format on a disk as four files- three binary and one text. To install a previously saved format from BASIC, type "EXEC Filename." This will then BLOAD in the other three files which, by the way, have the same name with a .1, .2 and .3 appended. The first binary file (the one named .1) is the CATALOG routine in DOS, the next contains the file types and the last is the disk title.

Note that the special INPUT routine is used when asking for the filename. This means that you may have a filename with inverse, flashing or even control letters in it. Although strange filenames aren't recommended because they are extremely hard to type from BASIC, TDDD will allow them.

[E] - EXIT

Use the last option, "D" to exit the program. If you exit by accident, just type "RUN" to get TDDD going again.

I use a general format for my nondescript disks that looks something like this:

APPLE SYSTEM][+

] TDDD	* SEARCH
* SPEED.SORT	] BLACK JACK
] SCREEN MAKER	* SCREEN.HELPER
* SCREEN.ROUTINES	* SCREEN.PIC
" AS->TEXT	> FLY

I replaced A for Applesoft with an ending bracket, I for Integer with a greater than sign, B for binary with an asterisk, and T for text file with a quote.

For experienced programmers, I have included a description of the program and how it works. For those of you who are not so interested, you can skip to the WARNINGS section for some precautions about this programs use.

Still with me? Good. Let's take a trip into the monitor and play around with DOS a bit.

1. Boot up on a disk with fresh, clean DOS on it such as the SYSTEM MASTER

PR#6

2. Enter the monitor

CALL-151

3. Examine the DOS CATALOG routine (it extends from AD98 through AE69)

AD98L

4. As you cruise around this portion of memory, you will notice many JSRs to $FDED and a few JSRs to $AE42 and $AE2F. Try replacing some of these with NOP's (code EA). Below are a few examples of what may happen to your CATALOG. After you have typed in a change, CATALOG the disk. Then fix it using the fix in the next column. If the fix won't work or the disk won't CATALOG, then reboot the disk.

CATALOG Modification Table

NORMALLY	CHANGE	EFFECT
ADB3:20 ED FD	ADB3:EA EA EA	Remove disk title
ADC0:20 42 AE	ADC0:EA EA EA	Get rid of VOL #
ADE5:20 ED FD	ADE5:EA EA EA	Remove LOCK/UNLOCK codes
ADF9:20 ED FD	ADF9:EA EA EA	Don't print file types
AE0B:20 42 AE	AE0B:EA EA EA	Get rid of file lengths
AE1B:20 ED FD	AE1B:EA EA EA	Don't print filename
AE22:20 2F AE	AE22:EA EA EA	Removes RETURN after filename.

The last modification at $AE22 will look a little weird, but by shortening the number of characters printed in Filename (AE16-AE17), you can get multiple columns.

Basically, my program manipulates the pointers and data tables used by DOS's CATALOG to change the directory's appearance.

Possible Modifications

When I first wrote the program, I was going to include options to switch or sort filenames. If anyone decides to make this modification, I would be interested in seeing it.

Another modification that would be nice is to have the program store the new directory format on the DOS tracks rather than as files.

Warnings

This program uses the area between $BCDF $BCFF to store the 32 character title. Some fast DOSs use this area and therefore, using my program could cause a crash. My customized DOS was engineered around this area.

TDDD uses the memory location used by the "R" file type for the 16 character title. For those of you using these relocatable files, a different format for these disks is advisable.

This program may interfere with modifications made using DOS BOSS and other DOS customizers.

Final comments

The Disk Directory Designer was written over the summer while I borrowed my high school teacher's computer to do some typing for her. It is a very useful utility and I have used it

successfully. I hope it is useful to you, too.

```
10  REM ************************
20  REM **      THE DISKETTE      **
30  REM ** DIRECTORY DESIGNER **
40  REM **            BY          **
50  REM **       TIM LEWIS      **
60  REM ************************
70  REM
80  REM *INITALIZE VARIABLES*
90  TEXT : NORMAL : SPEED=255: NOTRACE :
    GOSUB 1380
100 REM *BORDER*
110 HOME : POKE 48, ASC ("-"): HLIN
    0,39 AT 0: HLIN 0,39 AT 46: POKE
    48, ASC ("!") + 128: VLIN 2,
    45 AT 0: VLIN 2,45 AT 39
120 POKE 34,1: POKE 32,1: POKE 3
    5,23: POKE 33,38
130 REM *MENU*
140 HOME :M$ = "THE DISK DIRECTO
    RY DESIGNER": GOSUB 1440
150 VTAB 3:M$ = "----------------
    ------------": GOSUB 1440
160 VTAB 5:M$ = "WRITTEN BY TIM
    LEWIS": GOSUB 1440
170 VTAB 8: HTAB 10: PRINT  CHR$
    (91)"A] CHANGE DIRECTORY TIT
    LE"
180 VTAB 11: HTAB 10: PRINT  CHR$
    (91)"B] CHANGE DIRECTORY FOR
    MAT"
190 VTAB 14: HTAB 10: PRINT  CHR$
    (91)"C] LOAD/SAVE FORMAT & T
    ITLE"
200 VTAB 17: HTAB 10: PRINT  CHR$
    (91)"D] SEE DIRECTORY"
210 VTAB 20: HTAB 10: PRINT  CHR$
    (91)"E] EXIT PROGRAM"
220 REM *YOUR CHOICE?*
230 VTAB 23: HTAB 10: PRINT "YOU
    R CHOICE " CHR$ (91)" ]" CHR$
    (8) CHR$ (8);
240 GET A$: IF A$ > "E" OR A$ <
    "A" THEN 240
250 REM *GOTO PROPER ROUTINE*
260 HOME : ON  ASC (A$) - 64 GOSUB
    320,620,1140,280,1360: GOTO
    140
270 REM *SEE DIRECTORY*
280 TEXT : HOME : PRINT : PRINT
     CHR$ (4)"CATALOG"
290 PRINT : INVERSE : PRINT "PRE
    SS ANY KEY"
300 NORMAL : WAIT  - 16384,128: GET
    A$: POP : GOTO 110
310 REM *DIRECTORY TITLE*
320 M$ = "* DIRECTORY TITLE *": GOSUB
    1440: VTAB 5: PRINT "<A> 12
    CHARACTER TITLE"
330 PRINT "<B> 16 CHARACTER TITL
    E": PRINT "<C> 32 CHARACTER
    TITLE"
340 VTAB 9: PRINT "YOUR CHOICE <
     >" CHR$ (8) CHR$ (8);
350 GET A$: IF (A$ < "A" OR A$ >
    "C") AND A$ <  > CHR$ (27) AND
    A$ <  > CHR$ (13) THEN 350
360 IF A$ =  CHR$ (27) OR A$ =  CHR$
    (13) THEN  RETURN
370 HTAB 1: CALL  - 868: VTAB  ASC
    (A$) - 60: INVERSE : PRINT "
    <"A$">": NORMAL
380 IF A$ = "A" THEN B = 45999:T
     = 46010:ML = 12: POKE 44463
    ,11: POKE 44465,175: POKE 44
    466,179: GOTO 410
390 IF A$ = "B" THEN B = 45995:T
     = 46010:ML = 16: POKE 44463
    ,15: POKE 44465,171: POKE 44
    466,179: GOTO 410
400 B = 48351:T = 48382:ML = 32: POKE
    44463,31: POKE 44465,223: POKE
    44466,188
410 DH = T:DL = B
420 REM *GET TITLE*
430 VTAB 10: PRINT "ENTER TITLE"
    :V = 12: GOSUB 1460
440 REM *VOLUME #?*
450 PRINT : PRINT : PRINT "DISPL
    AY THE VOLUME NUMBER ?" CHR$
    (8);
460 GET A$: IF A$ <  > "N" AND A
    $ <  > "Y" THEN 460
470 IF A$ = "N" THEN B = 44480:T
     = B + 2: GOSUB 1750:B = DL:
    T = DH:VOL = 0: GOTO 500
480 POKE 44480,32: POKE 44481,66
    : POKE 44482,174:VOL = 1
490 REM *PRINT TITLE*
500 POKE 34,2: HOME : POKE 34,1:
     VTAB 5: PRINT "THE WAY IT W
    ILL APPEAR:": VTAB 8: GOSUB
    1720: IF VOL THEN  PRINT  PEEK
    (46017)
510 REM *OK?*
520 VTAB 11: HTAB 1: PRINT "RETU
    RN=>OK" SPC( 3)"SPACE=>REDO"
     SPC( 3)"ESC=>MENU": PRINT :
     PRINT "WHICH ?" CHR$ (8);
530 GET A$: IF A$ <  > " " AND A
    $ <  > CHR$ (27) AND A$ <  >
     CHR$ (13) THEN 530
540 IF A$ = " " THEN B = 45999:T
     = 46010:ML = 12: POKE 44463
    ,11: POKE 44465,175: POKE 44
    466,179: HOME : GOTO 320
550 HOME
560 IF A$ =  CHR$ (27) THEN B =
    45999:T = 46010:ML = 12: POKE
    44463,11: POKE 44465,175: POKE
    44466,179: RETURN
570 IF A$ <  > CHR$ (13) THEN  RETURN
580 REM *PUT TITLE IN MEMORY*
590 MM = 0
600 FOR I = T TO B STEP  - 1:MM =
    MM + 1: POKE I,IA%(MM): NEXT
    : RETURN
610 REM *DIRECTORY FORMAT*
620 M$ = "FORMAT ALTERATION": GOSUB
    1440: VTAB 4: PRINT "<A> LOC
    K/UNLOCK CODES": PRINT
630 PRINT "<B> FILE TYPES": PRINT
    : PRINT "<C> FILE LENGTHS": PRINT
640 PRINT "<D> FILE LENGTH SURRO
    UNDINGS": PRINT : PRINT "<E>
     NUMBER OF COLUMNS IN DIRECT
    ORY"
650 PRINT : PRINT "<ESC> EXIT TO
     MAIN MENU": GOSUB 1790: IF
     PEEK (44567) = 29 THEN CHA =
    30: GOTO 670
660 CHA = (40 / COL) - DC: POKE 4
    4567,CHA - 1
670 VTAB 22: PRINT "FILENAME LIM
    IT: "CHA" CHARACTERS": VTAB
    17: PRINT "WHICH ?" CHR$ (8)
    ;
680 GET A$: IF (A$ < "A" OR A$ >
    "E") AND A$ <  > CHR$ (27) AND
    A$ <  > CHR$ (13) THEN 680
690 IF A$ =  CHR$ (27) OR A$ =  CHR$
    (13) THEN  RETURN
700 HOME : ON  ASC (A$) - 64 GOSUB
    720,770,840,880,950: HOME : GOTO
    620
710 REM *LOCK/UNLOCK*
720 T = 44519:B = 44517:M$ = "DIS
    PLAY LOCK/UNLOCK CODES": GOSUB
    1070
730 IF A$ = "N" THEN 1750
740 GOSUB 1770: VTAB 6:M$ = "ENT
    ER 'UNLOCK' CODE": GOSUB 112
    0: POKE 44508,X
750 VTAB 11:M$ = "ENTER 'LOCK' C
    ODE": GOSUB 1120: POKE 44515
    ,X: RETURN
760 REM *FILE TYPE*
770 T = 44539:B = T - 2:M$ = "DIS
    PLAY FILE TYPES": GOSUB 1070
780 IF A$ = "N" THEN 1750
790 GOSUB 1770: VTAB 5:M$ = "ENT
    ER BINARY FILE CODE": GOSUB
    1120: POKE 45994,X
800 VTAB 10:M$ = "ENTER APPLESOF
    T FILE CODE": GOSUB 1120: POKE
    45993,X
810 VTAB 15:M$ = "ENTER INTEGER
    FILE CODE": GOSUB 1120: POKE
    45992,X
820 VTAB 20:M$ = "ENTER TEXT FIL
    E CODE": GOSUB 1120: POKE 45
    991,X: RETURN
830 REM *SECTOR #*
840 T = 44557:B = 44555:M$ = "DIS
    PLAY THE FILE LENGTHS": GOSUB
    1070
850 IF A$ = "N" THEN 1750
860 POKE B,32: POKE B + 1,66: POKE
    T,174: RETURN
870 REM *FUNNY CHARS*
880 M$ = "DISPLAY THE CHARACTER T
    O THE LEFT  OF THE FILE LENG
    TH": GOSUB 1070
890 IF A$ = "N" THEN  POKE 44541
    ,128: GOTO 910
900 VTAB 6:M$ = "ENTER THE CHARA
    CTER": GOSUB 1120: POKE 4454
    1,X
910 M$ = "DISPLAY THE CHARACTER T
    O THE RIGHT OF " + CHR$ (10
    ) + "THE FILE LENGTH": GOSUB
    1070
920 IF A$ = "N" THEN  POKE 44559
    ,128: RETURN
930 VTAB 6:M$ = "ENTER THE CHARA
    CTER": GOSUB 1120: POKE 4455
    9,X: RETURN
940 REM *COLUMNS*
950 HOME :M$ = "DIRECTORY COLUMN
     SELECTION": GOSUB 1440: VTAB
    5: PRINT "<A> 1 COLUMN (NORM
    AL)": PRINT "<B> 2 COLUMNS"
960 PRINT "<C> 4 COLUMN DIRECTOR
    Y": PRINT "<D> 5 COLUMNS IN
```

DEAR AUTHOR:

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

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

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

We prefer to see your submission on a DOS 3.3 disk using an Apple (or compatible) editing program. Please enclose a double-spaced hardcopy (paper) manuscript using a dot-matrix or letter-quality printer (or typewriter). Submissions will be mailed back if adequate 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. Payment ranges between $10 for a short softkey, and $50 per typeset page for a full-length article. We pay more for softkeys if the original commercial disk is enclosed for verification. We guarantee the disk's return.

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

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

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

Hardcore COMPUTIST
WRITER'S GUIDE
P. O. BOX 44549
TACOMA WA 98444

```
      DIRECTORY"
970   PRINT : PRINT "WHICH ?" CHR$
      (8);
980   GET A$: IF (A$ < "A" OR A$ >
      "D") AND A$ < > CHR$ (13) AND
      A$ < > CHR$ (27) THEN 980
990   IF A$ = CHR$ (27) OR A$ = CHR$
      (13) THEN RETURN
1000 T = 44580:B = T - 2
1010  IF A$ = "A" THEN POKE B,32
      : POKE B + 1,47: POKE T,174:
      POKE 44567,29: RETURN
1020  POKE 44567,0: GOSUB 1750:CO
      L = 5: IF A$ = "B" THEN COL =
      2
1030  IF A$ = "C" THEN COL = 4
1040  RETURN
1050 CHA = (40 / COL) - DC: POKE
      44567,CHA - 1
1060  REM *DISPLAY SOMETHING?*
1070  HOME : VTAB 3: PRINT M$" ?"
      CHR$ (8);:ML = 1
1080  GET A$: IF A$ < > "Y" AND
      A$ < > "N" AND A$ < > CHR$
      (13) AND A$ < > CHR$ (27) THEN
      1080
1090  IF A$ = CHR$ (13) OR A$ =
      CHR$ (27) THEN POP : RETURN
1100  PRINT CHR$ (8)":"A$: RETURN
1110  REM *ASK FOR ALTERATION*
1120  HTAB 1:V = PEEK (37) + 3: PRINT
      M$: GOSUB 1460:X = IA%(1): RETURN
1130  REM *LOAD/SAVE FORMAT*
1140 M$ = "* LOAD/SAVE DIRECTORY
      TYPE *": GOSUB 1440: VTAB 5:
      PRINT "FILENAME TO USE:"
1150 V = 7:ML = 15: GOSUB 1460:ML
      = L: POKE 771,237
1160  VTAB 11: PRINT "<L>OAD DIRE
      CTORY" SPC( 5)"<S>AVE DIRECT
      ORY"
1170  PRINT " ESC=>MENU" SPC( 12)
      "SPACE=>REDO": PRINT : PRINT
      "WHICH ?" CHR$ (8);
1180  GET A$: IF A$ < > "L" AND
      A$ < > CHR$ (27) AND A$ <
      > " " AND A$ < > "S" THEN
      1180
1190  IF A$ = CHR$ (27) THEN POKE
      771,240: RETURN
1200  IF A$ = " " THEN HOME : GOTO
      1140
1210  HOME : VTAB 12: HTAB 11: FLASH
      : PRINT "ONE MOMENT PLEASE."
      : NORMAL
1220  IF A$ = "S" THEN 1250
1230  FOR X = 1 TO 3: PRINT CHR$
      (4)"BLOAD";: GOSUB 1720: PRINT
      "."X: NEXT : POKE 771,240: RUN
1240  REM *SAVE FORMAT*
1250  PRINT : PRINT CHR$ (4)"BSA
      VE";: GOSUB 1720: PRINT ".1,
      A$AD98,L$97"
1260  PRINT CHR$ (4);"BSAVE";: GOSUB
      1720: PRINT ".2,A$B3A7,L$04"
1270  PRINT CHR$ (4)"BSAVE";: GOSUB
      1720: PRINT ".3,A";DL;",L";D
      H - DL + 1
```

"It is my sad duty to inform you that all of you, except one, will be replaced by computers."

```
1280  REM *EXEC FILE*
1290  PRINT  CHR$ (4)"OPEN";: GOSUB
      1720: PRINT : PRINT  CHR$ (4
      )"DELETE";: GOSUB 1720: PRINT
1300  PRINT  CHR$ (4)"OPEN";: GOSUB
      1720: PRINT : PRINT  CHR$ (4
      )"WRITE";: GOSUB 1720: PRINT
1310  FOR X = 1 TO 3: PRINT "BLOA
      D";: GOSUB 1720: PRINT "."X:
       NEXT : PRINT  CHR$ (4)"CLOS
      E"
1320  HOME : VTAB 7: PRINT "TO IN
      STALL NEW FORMAT:": PRINT
1330  PRINT "USE 'EXEC ";: POKE 7
      71,240: GOSUB 1720: PRINT "'
      "
1340  PRINT : PRINT : POKE 771,24
      0: GOTO 290
1350  REM *EXIT*
1360  TEXT : HOME : VTAB 24: POP
      : END
1370  REM *INITIALIZE*
1380 T = 0:B = 0:DL = 45999:DH =
      46010:CHA =  PEEK (44567) +
      1:COL = 1: IF CHA < 30 THEN
      COL = 2
1390 ML = 0:L = 0:H = 0:V = 0: GOSUB
      1790: IF CHA < 13 THEN COL =
      4: IF CHA < 5 THEN COL = 5
1400  POKE 768,169: POKE 770,76: POKE
      771,240: POKE 772,253
1410  DIM IA%(32)
1420  RETURN
1430  REM *CENTER*
1440  HTAB 20 -  INT ( LEN (M$) /
      2): PRINT M$: RETURN
1450  REM *INPUT ROUTINE*
1460 L = 0:X = 1: FOR I = 1 TO 32
      :IA%(I) = 160: NEXT : VTAB V
      : HTAB 1
1470  PRINT "NRML>" CHR$ (10);: FOR
      I = 1 TO ML: PRINT "-";: NEXT
      :H = 6
1480  VTAB V: HTAB H: GET A$
1490  IF A$ =  CHR$ (29) THEN A$ =
       CHR$ (91)
1500  IF A$ =  CHR$ (30) THEN A$ =
       CHR$ (95)
1510  IF A$ =  CHR$ (0) THEN A$ =
       CHR$ (28)
1520  IF A$ =  CHR$ (13) AND L <
      > 0 THEN  PRINT : RETURN
1530  IF A$ =  CHR$ (8) AND L = 0
       THEN 1480
1540  IF A$ =  CHR$ (8) THEN IA%(
      L) = 32:L = L - 1:H = H - 1:
       PRINT A$;: CALL  - 868: GOTO
      1480
1550  IF A$ =  CHR$ (14) THEN  HTAB
      1: PRINT "NRML";:X = 1: GOTO
      1480
1560  IF A$ =  CHR$ (9) THEN  HTAB
      1: PRINT "INVS";:X = 2: GOTO
      1480
1570  IF A$ =  CHR$ (6) THEN  HTAB
      1: PRINT "FLSH";:X = 3: GOTO
      1480
1580  IF A$ =  CHR$ (3) THEN  HTAB
      1: PRINT "CTRL";:X = 4: GOTO
      1480
1590  IF A$ =  CHR$ (12) THEN  HTAB
      1: PRINT "LOWR";:X = 5: GOTO
      1480
1600  IF L =  > ML OR A$ < " " OR
      A$ >  CHR$ (95) THEN 1480
1610 L = L + 1:H = H + 1: ON X GOTO
      1620,1630,1650,1670,1690
1620  PRINT A$;:IA%(L) =  ASC (A$
      ) + 128: GOTO 1480
1630  INVERSE : PRINT A$;: NORMAL
      :IA%(L) =  ASC (A$) - 64: IF
      A$ < "@" THEN IA%(L) =  ASC
      (A$)
1640  GOTO 1480
1650  FLASH : PRINT A$;: NORMAL :
      IA%(L) =  ASC (A$): IF A$ <
      "@" THEN IA%(L) =  ASC (A$) +
      64
1660  GOTO 1480
1670  IF A$ < "@" THEN 1620
1680  INVERSE : PRINT A$;: NORMAL
      :IA%(L) =  ASC (A$) + 64: GOTO
      1480
1690  IF A$ < "@" THEN 1620
1700  PRINT  CHR$ ( ASC (A$) + 32
      );:IA%(L) =  ASC (A$) + 160:
       GOTO 1480
1710  REM *PRINT WAY IT APPEARS*
1720  FOR I = 1 TO ML
1730  POKE 769,IA%(I): CALL 768: NEXT
      : RETURN
1740  REM *NEGATE CAT. ROUTINE*
1750  FOR I = B TO T: POKE I,234:
       NEXT : RETURN
1760  REM *ACTIVATE CAT. ROUTINE*
1770  POKE B,32: POKE B + 1,237: POKE
      T,253: RETURN
1780  REM *FIGURE EXTRA SPACE*
1790 DC = ( PEEK (44517) <  > 234
      ) + ( PEEK (44539) <  > 234)
1800 DC = DC + 3 * ( PEEK (44555)
      <  > 234) + ( PEEK (44541) <
      > 128) + ( PEEK (44559) <  >
      128): RETURN
```

Checksums

```
10    - $BADD   410   - $A22B   810   - $5CCB   1210  - $CC56   1610  - $EB3B
20    - $9B13   420   - $B7BE   820   - $762C   1220  - $92F5   1620  - $ABE2
30    - $4D3B   430   - $8BC1   830   - $614F   1230  - $052C   1630  - $0B7F
40    - $AD92   440   - $CDD1   840   - $CDC5   1240  - $6C48   1640  - $5FB9
50    - $C899   450   - $C189   850   - $B9FD   1250  - $95E9   1650  - $3788
60    - $FF65   460   - $6A87   860   - $C058   1260  - $16EC   1660  - $1861
70    - $A3BF   470   - $7D8F   870   - $1555   1270  - $0735   1670  - $9687
80    - $A900   480   - $AD1C   880   - $3FF9   1280  - $63AD   1680  - $035E
90    - $6635   490   - $1C58   890   - $2B69   1290  - $6E0E   1690  - $FD6E
100   - $F3B7   500   - $D176   900   - $9098   1300  - $26D7   1700  - $69A3
110   - $9C15   510   - $6E22   910   - $039A   1310  - $6925   1710  - $625F
120   - $2A19   520   - $5710   920   - $7F19   1320  - $1863   1720  - $E2E5
130   - $972C   530   - $B2D8   930   - $35A5   1330  - $5904   1730  - $9C8D
140   - $DB48   540   - $A360   940   - $9C7F   1340  - $A3B5   1740  - $D4D7
150   - $D3E1   550   - $F2F9   950   - $5D7B   1350  - $DB36   1750  - $00DF
160   - $8937   560   - $F90C   960   - $F48E   1360  - $445C   1760  - $9ED1
170   - $888C   570   - $FAE5   970   - $8836   1370  - $DEA9   1770  - $8A3F
180   - $DA93   580   - $06A9   980   - $5190   1380  - $9620   1780  - $FE91
190   - $AAB4   590   - $BAAB   990   - $B881   1390  - $09B4   1790  - $7566
200   - $B1A5   600   - $7DA0   1000  - $56B2   1400  - $2E2F   1800  - $DB86

210   - $9786   610   - $F7E7   1010  - $41FB   1410  - $841B
220   - $E4C8   620   - $1DEA   1020  - $8ADF   1420  - $DEA1
230   - $8B1A   630   - $CBC8   1030  - $4D2B   1430  - $375F
240   - $063C   640   - $1611   1040  - $145B   1440  - $EE53
250   - $A636   650   - $9DA9   1050  - $237F   1450  - $7D2D
260   - $3F80   660   - $0FD6   1060  - $B3D6   1460  - $A9EA
270   - $F000   670   - $3129   1070  - $2A97   1470  - $CAF9
280   - $425C   680   - $B7F8   1080  - $C88B   1480  - $9827
290   - $9F73   690   - $AD61   1090  - $86B5   1490  - $4DDB
300   - $D62B   700   - $0359   1100  - $6D68   1500  - $2020
310   - $5F77   710   - $8A89   1110  - $4053   1510  - $42DE
320   - $5780   720   - $56A7   1120  - $5D0E   1520  - $6F8B
330   - $2347   730   - $A7A1   1130  - $CE09   1530  - $C01D
340   - $C51E   740   - $332C   1140  - $AF13   1540  - $ABB5
350   - $403A   750   - $0BEF   1150  - $F603   1550  - $544D
360   - $E4E2   760   - $5CAE   1160  - $FD17   1560  - $6BEA
370   - $AA62   770   - $D33A   1170  - $E6BE   1570  - $378E
380   - $0E23   780   - $769C   1180  - $30E5   1580  - $9B68
390   - $FEC5   790   - $1EC2   1190  - $0CF3   1590  - $3166
400   - $07D0   800   - $A494   1200  - $BA32   1600  - $859C
```

COREFILERCOREFILERCOREFILERCOREFILERCOREFILERCOREFILERCOREFILERCOREFILERCOREFILERCOREFILE

COREFILER

COREFILERCOREFILERCOREFILERCOREFILERCOREFILERCOREFILERCOREFILERCOREFILERCOREFILERCOREFILE

Requirements:
Apple II Plus or compatible with 48K RAM
One disk drive

A lot of you out there are probably thinking that databases may be the greatest thing since the living girdle, but what good are they to me unless I open my own business manufacturing microprocessor-controlled panty hose, or some such item?

Because the survey of the readers of Hardcore COMPUTIST which we conducted has shown that the average reader cannot maintain a proper mental perspective without typing in at least 250 lines of BASIC or Assembly code each month, the editor decided to present a program that would both satisify our readers' desires and demonstrate an application of a database for the average person.

This program will allow you to keep a record of any kind of important information you have, whether it be your Christmas card mailing list or a catalog of your collection of sixteenth century Samurai swords.

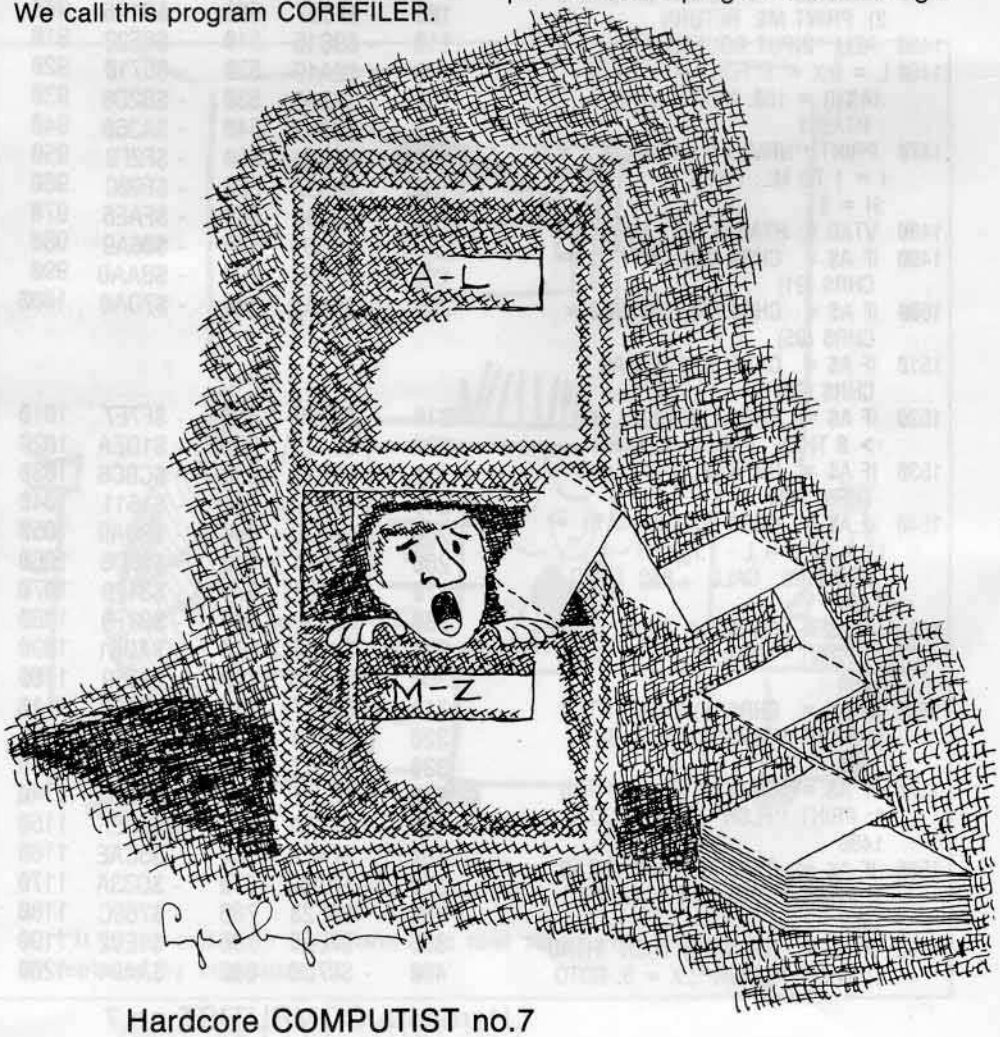
The user can set up the format of the files to meet his particular needs. Then he can enter, edit, search and finally print out the stored information, either one record at a time, or the whole file. We call this program COREFILER.

COREFILER evolved out of a program that Softkey Publishing uses in-house to keep track of its subscribers and advertising clients. Since most people don't have a great need to do this sort of thing, we decided to modify our original database program so that it could be configured to almost any application.

The COREFILER system is really four separate programs which work in conjunction with one another but, because of the length of these programs, this issue we will only contain the main database program. The following issue will contain the other three programs which handle screen formating, file configuration and menu presentation.

Basically a file management program as currently structured, COREFILER can handle a maximum of 450 records per file on a floppy or hard disk based Apple or Apple compatible. Only one file can be operated on at a time, but this should not be a serious restriction for limited home or small business applications.

Like most other database-type programs, COREFILER uses random access text files for quick storage and retrieval of data. Table 1 gives a complete list of the program's file manage-



ment functions which includes: Add, Delete, Edit, Search, Print and Read. A sorting capability has not been included.

COREFILER may not have all the whistles and bells which some of the commercial database packages have, but its price is surely hard to beat. Enterprising programmers might like to add a sort or more sophisticated search routines to the program.

Because of the length of the COREFILER program, I suggest that the CHECKSOFT program from Hardcore COMPUTIST 1 be run on COREFILER. This will make it much easier for you to spot any errors you may have made while typing in the program. The checksums following the COREFILER listing were generated by using CHECKSOFT in its normal configuration mode (ignore REM's and program pointers. Use line numbers).

Next issue we will print the remaining programs which make up the COREFILER system. By that time, you may have finished typing in the listing from this issue.

Table 1

COREFILER Basic Functions

A - Adds records to the current file.

E - Edits the information in the current record.

L - Retrieves the record previous to the current record.

N - Retrieves the record which follows the current record.

R - Reads in any record in the database.

S - Searches the entire database for the specified information in a particular field.

D - Deletes the current record from the database.

P - Prints out either the current record or the entire file. The printer interface is assumed to reside in slot#1.

/ - Allows the user to work on another existing file.

X - Exits the COREFILER program.

Editing Functions

CTRL D - Deletes the character under the cursor.

CTRL B - Jumps to the beginning of the current field.

CTRL E - Jumps to the end of data in the current field.

CTRL L - Deletes from the cursor to the end of the current field.

CTRL R - Recovers the data that was in the current field before editing was begun.

CTRL P - Exchanges the data in the current field with the data in the following field.

ESC - Saves the current record to disk and exits the edit mode.

Add Mode Functions

ESC - Saves the new record to disk. Hitting **ESC** a second time exits the "Add" mode.

CTRL X - Exits the "Add" mode without saving the current record to disk.

```
10  REM  ***************************
20  REM  *                         *
30  REM  *        COREFILER        *
40  REM  *                         *
50  REM  *    FILE MANAGEMENT      *
60  REM  *                         *
70  REM  *        PROGRAM          *
80  REM  *                         *
90  REM  ***************************
95  REM
100  POKE 1013,76: POKE 1014,9: POKE
     1015,3
110  FOR X = 777 TO 827: READ X%:
      POKE X,X%: NEXT X
120  DATA 201,132,240,3,76,201,22
     2,32,177,0,32,227,223,32,44,
     213,232,189,0,2,208,250,134,
     157,138,32,82,228,160,0,145,
     131,200,165,113,145,131,200,
     165,114,145,131,160,2,162,0,
     165,157,76,226,229
130 G$ =  CHR$ (7)
140 A$ = "":D$ =  CHR$ (4): TEXT
    : HOME :G$ =  CHR$ (7)
150 SL =  PEEK (770):DV =  PEEK (
    771)
160  IF  PEEK (768) = 255 AND  PEEK
     (769) = 255 THEN 180
170  POKE 768,255: POKE 769,255: POKE
     770,SL: POKE 771,DV
180  GOTO 2630
190  REM READ A FILE
200  PRINT : PRINT OF$XL",V"V: FOR
     X = 1 TO RC: PRINT RF$X",B"B
     Y(S): &  INPUT S$(X): NEXT :
      PRINT CL$: RETURN
210  REM READ A RECORD
220  IF RR = 0 THEN  RETURN
230 A$ = ">> READING RECORD #" +
     STR$ (RR) + " <<": GOSUB 11
     20
240  PRINT : PRINT OF$XL",V"V: FOR
     X = 1 TO NF: PRINT RF$RR",B"
     BY(X): &  INPUT A$(X): NEXT
     : PRINT CL$
250  RETURN
```

Attention Adventurers!

ADVENTURE DATA BASE

Hardcore COMPUTIST is looking for adventure hints to any of the popular adventure/fantasy games sold for the Apple II/II Plus or IIe. We hope to develop a data base of these hints and, if sufficient response is received, to publish a regular column in Hardcore COMPUTIST.

YOUR CLUES, PLEASE

We prefer that these hints not be dead giveaway solutions to dilemmas presented by the particular game but instead, contain just enough information to nudge the stumped adventurer towards the solution of his or her problem.

HOW AND WHERE

So, if you know how to open the jewel-encrusted egg, how to plug the hole in the rowboat, where to find the key to the treasure chest, or any other information that may be of help to your fellow travelers, please send this information on a 3×5 postcard to:

Hardcore COMPUTIST
Attention: Adventure Hints
P.O. Box 44549
Tacoma, WA 98444

P.S. Please don't forget to include the name of the adventure game to which your hint pertains.

```
260  REM GET ID NUMBER
270  PRINT : PRINT OF$XL",V"V: PRINT
     RF$: INPUT RO,RT,IO: PRINT C
     L$: RETURN
280  REM WRITE A RECORD
290  IF RR = 0 THEN WF = RR: RETURN
300 A$ = ">> SAVING RECORD #" +  STR$
     (RR) + " <<": GOSUB 1120
310  PRINT : PRINT OF$XL",V"V: FOR
     X = 1 TO NF: PRINT WF$RR",B"
     BY(X): PRINT A$(X): NEXT
320  PRINT WF$0: PRINT RC: PRINT
     RT: PRINT ID: PRINT CL$:WF =
     0: RETURN
330  REM READ VARIABLES FILE
340  PRINT : PRINT D$"OPEN"N$".CO
     NFIG": PRINT D$"READ"N$".CON
     FIG": INPUT NF: FOR X = 1 TO
     NF: INPUT VT(X),HT(X),NN$(X)
     ,H2(X),LN(X),BY(X): NEXT : INPUT
     XL: INPUT V: PRINT D$"CLOSE"
     : RETURN
350  REM FLIP FIELDS
360 CC = CT + 1
370  IF CC > NF THEN CC = 1
380 T$ = A$(CT):A$(CT) = A$(CC):A
     $(CC) = T$
390  RETURN
400  REM >> INPUT ROUTINE CODE <<
410  REM PRINT NEW STRING
420  VTAB VT(CT): HTAB H2(CT): IF
      NOT ED THEN  PRINT A$(CT): RETURN
430  IF  LEN (A$(CT)) < 1 AND HT(
     CT) = H2(CT) THEN  PRINT NN$
     (CT);:LL =  LEN (NN$(CT)): GOTO
     450
440  PRINT A$(CT);:L =  LEN (A$(C
     T)):LL = L
450  IF LL = LN(CT) THEN  RETURN
460  FOR X = 1 TO LN(CT) - LL: PRINT
      CHR$ (95);: NEXT : RETURN
470  REM DELETE SUBROUTINE
480  IF P > L THEN  RETURN
490  IF L = 1 THEN A$(CT) = "": RETURN
500  IF P = L THEN A$(CT) =  LEFT$
     (A$(CT),P - 1): RETURN
510  IF P = 1 THEN A$(CT) =  RIGHT$
     (A$(CT),L - 1): RETURN
520 A$(CT) =  LEFT$ (A$(CT),P - 1
     ) +  RIGHT$ (A$(CT),L - P): RETURN
530  REM INSERT SUBROUTINE
540  IF P = 1 THEN A$(CT) = CH$ +
     A$(CT): RETURN
550  IF P = L THEN A$(CT) =  LEFT$
     (A$(CT),P - 1) + CH$ +  RIGHT$
     (A$(CT),1): RETURN
560 A$(CT) =  LEFT$ (A$(CT),P - 1
     ) + CH$ +  RIGHT$ (A$(CT),L -
     P + 1): RETURN
570  REM  BACKWARD MOVE
580 P = P - 1: IF P > 0 THEN 610
590 P = 1: IF CT = 0 THEN  RETURN
600  NORMAL : GOSUB 420: INVERSE
     :CT = CT - 1: IF CT < 1 THEN
     CT = NF
610 B$ = A$(CT):L =  LEN (A$(CT))
620  RETURN
630  REM  FORWARD MOVE
640 P = P + 1: IF P > L + 1 THEN
     P = L + 1
650  IF P > LN(CT) THEN P = 1:CT =
     CT + 1:L =  LEN (A$(CT)): IF
     CT > NF THEN CT = 1
660  RETURN
670  REM  MOVE CURSOR BACKWARD
680  IF CH$ =  CHR$ (8) THEN  GOSUB
     580: GOTO 990
690  REM  MOVE CURSOR FORWARD
700  IF CH$ =  CHR$ (21) THEN  GOSUB
     640: GOTO 990
710  REM SAME AS BEFORE
720  IF CH$ =  CHR$ (18) THEN A$(
     CT) = B$:P = 1: GOTO 990
730  IF CH$ =  CHR$ (16) THEN  GOTO
     360
740  REM DELETE A CHARACTER
750  IF CH$ =  CHR$ (4) THEN  GOSUB
     480: GOSUB 420: GOTO 1080
760  REM EXIT TO MENU
770  IF CH$ <  >  CHR$ (27) THEN
     840
780  IF  NOT AD THEN  NORMAL : GOSUB
     420: POP : GOTO 1450
790  FOR D = 1 TO NF: IF A$(D) <
      > "" THEN ZW = 1
800  NEXT
810  IF ZW THEN ZW = 0: NORMAL : GOSUB
     420: POP : GOTO 1450
820 AD = 0:WF = 0:RR = RR - 1:RC =
     RC - 1: NORMAL : GOSUB 220: GOTO
     1450
830  REM RETURN TO CALLER
840  IF CH$ =  CHR$ (13) THEN  RETURN
850  REM RETURN WITHOUT SAVE
860  IF CH$ =  CHR$ (24) THEN  RETURN
870  REM JUMP TO END OF LINE
880  IF CH$ =  CHR$ (5) THEN P =
     L + 1: GOSUB 420: GOTO 1080
890  REM JUMP TO START OF LINE
900  IF CH$ =  CHR$ (2) THEN P =
     1: GOSUB 420: GOTO 1080
910  REM CLEAR TO END OF LINE
920  IF CH$ =  CHR$ (12) AND P =
     1 THEN A$(CT) = "": GOSUB 42
     0: GOTO 1080
930  IF CH$ =  CHR$ (12) THEN A$(
     CT) =  LEFT$ (A$(CT),P - 1):
      GOSUB 420: GOTO 1080
940  REM INITIAL ENTRY POINT
950 B$ = A$(CT):L =  LEN (A$(CT))
960  REM ENTER WITH STRING AND TA
     BS
970  GOSUB 420:P = 1
980  REM TAB AND PRINT STRING
990  GOSUB 420: VTAB VT(CT): HTAB
     H2(CT) - 1 + P: GET CH$
1000  REM CTRL CHARACTER CHECK
1010  IF CH$ < " " THEN 680
1020  REM INSERT A CHARACTER
1030  IF P <  = L THEN  GOSUB 540
     : GOTO 1060
1040 A$(CT) = A$(CT) + CH$
1050  REM INCREMENT CURSOR
1060 P = P + 1
1070  REM IS FIELD FULL?
1080 L =  LEN (A$(CT)): IF L > LN
     (CT) THEN L = LN(CT):A$(CT) =
      LEFT$ (A$(CT),LN(CT))
1090  IF P > LN(CT) THEN P = LN(C
     T): PRINT  CHR$ (7): RETURN
1100  GOTO 990
1110  REM CENTER AND PRINT
1120  GOSUB 1390: VTAB 10
1130 HT = 20 - ( LEN (A$) / 2): IF
     HT < 1 THEN HT = 1
1140  HTAB HT: PRINT A$: RETURN
1150 A2$ = ""
1160  REM CENTER AND PRINT WITH P
     ROMPT
1170  GOSUB 1390: VTAB 8
1180  IF  LEN (A$) > 1 THEN  GOSUB
     1130: PRINT
1190  IF  LEN (A2$) > 1 THEN A$ =
     A2$: GOSUB 1130: PRINT
1200  VTAB 16
1210  IF AX THEN  PRINT " PRESS R
     ETURN WHEN READY TO CONTINUE
     :"G$;
1220  IF  NOT AX THEN  PRINT "PRE
     SS 'Y' FOR YES OR 'N' FOR NO
      (Y/";: INVERSE : PRINT "N";
     : NORMAL : PRINT "):"G$;
1230  GET AN$: PRINT AN$
1240  IF AX AND AN$ <  >  CHR$ (1
     3) AND AN$ <  >  CHR$ (27) THEN
     1200
1250  IF  NOT AX AND AN$ =  CHR$
     (13) THEN AN$ = "N"
1260  IF  NOT AX AND AN$ <  > "Y"
      AND AN$ <  > "N" THEN 1200
1270  RETURN
1280  REM ZERO DATA STRINGS
1290  FOR CT = 1 TO NF:A$(CT) = "
     ": NEXT
1300  REM PRINT RECORD STATUS
1310  VTAB 1: HTAB 1: PRINT "RECO
     RD #"RR"  ";: HTAB 17: PRINT
     "USED "RC"  ";: HTAB 30: PRINT
     "UNUSED "RT - RC;: CALL  - 8
     68
1320  REM PRINT DATA TO SCREEN
1330  GOSUB 1390: FOR CT = 1 TO N
     F: IF HT(CT) = H2(CT) THEN 1
     350
1340  VTAB VT(CT): HTAB HT(CT): PRINT
     NN$(CT);
1350  GOSUB 420: NEXT : VTAB 19: RETURN
1360  REM PRINT 40 "="'S
1370  HTAB 1: FOR X = 1 TO 40: PRINT
     "=";: NEXT : RETURN
1380  REM CLEAR DATA WINDOW
1390  POKE 34,2: POKE 35,17: HOME
     : POKE 34,18: POKE 35,24: RETURN
1400  REM ERROR-NO RECORDS
1410 A$ = "> NO RECORD AVAILABLE
     <":AX = 1: GOSUB 1150
1420  REM COMMAND MENU
1430  FOR CT = 1 TO NF:A$(CT) = "
     ": NEXT
1440  REM NORMAL ENTRY POINT
1450  CALL  - 10621:ED = 0: GOSUB
     1310: IF WF THEN  GOSUB 290:
      GOSUB 1310
1460  IF AD THEN 1660
1470  HOME
1480  PRINT "(A) ADD";: HTAB 11: PRINT
     "(S) SEARCH";: HTAB 28: PRINT
     "(D) DELETE"
1490  PRINT "(E) EDIT";: HTAB 11:
      PRINT "(R) READ RECORD";: HTAB
     28: PRINT "(/) NEW FILE"
1500  PRINT "(L) LAST";: HTAB 11:
      PRINT "(P) PRINT";: HTAB 28
     : PRINT "(?) STATUS"
1510  PRINT "(N) NEXT";: HTAB 28:
      PRINT "(X) EXIT"
```

```
1520  HTAB 1: VTAB 24: PRINT "ENT
      ER COMMAND: ";: GET AN$: VTAB
      23
1530  IF AN$ = "A" THEN 1660
1540  IF AN$ = "E" THEN 1720
1550  IF AN$ = "L" THEN 1780
1560  IF AN$ = "N" THEN 1820
1570  IF AN$ = "S" THEN 1870
1580  IF AN$ = "R" THEN 2200
1590  IF AN$ = "P" THEN 2270
1600  IF AN$ = "D" THEN  GOSUB 24
      90: GOTO 1450
1610  IF AN$ = "X" THEN 2570
1620  IF AN$ = "?" THEN 2620
1630  IF AN$ = "/" THEN  RUN
1640  PRINT  CHR$ (7);: GOTO 1520
1650  REM ADD A RECORD
1660  IF RC = RT THEN A$ = "> DIS
      K FULL <": GOSUB 1150: GOTO
      1450
1670 AD = 1: HOME : PRINT : HTAB
      14: PRINT "ADD A RECORD": PRINT
      : PRINT " PRESS <ESC> TO RET
      URN TO COMMAND MENU."
1680 RC = RC + 1:SE = 0:RR = RC:E
      D = 1: GOSUB 1290:CT = 1:WF =
      1
1690 B$ = A$(CT): INVERSE : GOSUB
      970: NORMAL : GOSUB 420:CT =
      CT + 1: IF CT > NF THEN CT =
      1
1700  IF CH$ =  CHR$ (24) THEN WF
       = 0:RC = RC - 1:RR = RC: GOSUB
      350: GOSUB 220: GOTO 1450
1710  GOTO 1690
1720  IF RR = 0 THEN 1410
1730  HOME : PRINT : HTAB 14: PRINT
      "EDIT A RECORD": PRINT : PRINT
      " PRESS <ESC> TO RETURN TO C
      OMMAND MENU."
1740 ED = 1: GOSUB 1310:CT = 1:WF
       = 1
1750  INVERSE : GOSUB 950: NORMAL
      : GOSUB 420:CT = CT + 1: IF
      CT > NF THEN CT = 1
1760  GOTO 1750
1770  REM READ LAST RECORD
1780  REM
1790 RR = RR - 1: IF RR < 1 THEN
      RR = 1: GOTO 1450
1800  GOSUB 220: GOTO 1450
1810  REM READ NEXT RECORD
1820  REM
1830 RR = RR + 1: IF RR > RC THEN
      RR = RC: GOTO 1450
1840  GOSUB 220: GOTO 1450
1850  REM >> ENTRY SEARCH <<
1860  REM CLEAR SCREEN
1870  VTAB 1: HTAB 9: PRINT "
      ": GOSUB 1390: PRINT :X = 1
1880  REM PRINT FIELDS
1890  PRINT X". "NN$(X);: IF X <
       > NF THEN X = X + 1: HTAB 2
      0: PRINT X". "NN$(X): IF X <
       > NF THEN X = X + 1: PRINT
      : GOTO 1890
1900  HOME : PRINT
1910  REM GET FIELD NUMBER
1920  HOME : PRINT : PRINT "FOUND
       IN WHAT FIELD? (1-"NF") :":
      CT = 0:VT(0) = 20:LN(0) = 2:
      H2(0) = 30:A$(0) = "":ED = 1
      : GOSUB 970
1930  IF  LEN (A$(CT)) < 1 THEN 1
      450
1940 S =  VAL (A$(CT)): IF S < 1 OR
      S > NF THEN 1920
1950  REM GET SEARCH KEY
1960  HOME : PRINT : PRINT "ENTER
       ITEM TO SEARCH FOR:":LN(0) =
      LN(S):VT(0) = 22:H2(0) = 1:A
      $(0) = "":ED = 1:CT = 0: GOSUB
      970:SF =  LEN (A$(0))
1970  REM CLEAR SCREEN
1980 ED = 0: HOME
1990  IF SF < 1 THEN 1450
2000  REM IS DATA IN MEMORY
2010  IF S = SE THEN 2060
2020  REM GET THE DATA
2030 A$ = "> RECALLING DATA <": GOSUB
      1120: GOSUB 200
2040  GOSUB 200
2050  REM COMPARE TO KEY
2060 SE = S:RX = RR:A$ = "> SEARC
      HING <": GOSUB 1120
2070  FOR RR = 1 TO RC
2080  IF  LEFT$ (S$(RR),SF) = A$(
      0) THEN  GOSUB 220: GOSUB 21
      50
2090  NEXT
2100  REM KEY NOT FOUND
2110 A$ = "SORRY.  I COULD NOT FI
      ND:":A2$ = "> " + A$(0) + "
      <": IF  LEN (A2$) > 40 THEN
      A2$ = A$(0)
2120 AX = 1: GOSUB 1170
2130 RR = RX: GOSUB 220: GOTO 145
      0
2140  REM CORRECT RECORD?
2150  GOSUB 1310: HOME : PRINT : PRINT
      "IS THIS THE RECORD YOU WANT
      ?  (Y/";: INVERSE : PRINT "N
      ";: NORMAL : PRINT ") ";: GET
      AN$: IF AN$ = "Y" THEN  POP
      : GOTO 1470
2160  IF AN$ =  CHR$ (27) THEN  POP
      :RR = RX: GOSUB 220: GOTO 14
      50
2170  HOME :A$ = "> SEARCHING <":
       GOSUB 1390: VTAB 10: GOSUB
      1130
2180  RETURN
2190  REM >>READ A RECORD<<
2200  HOME : PRINT : HTAB 5: PRINT
      G$"ENTER RECORD NUMBER TO RE
      AD: ";:CT = 0:VT(CT) = 20:LN
      (CT) = 3:H2(CT) = 34:HT(CT) =
      H2(CT):NN$(CT) =  STR$ (RR):
      A$(CT) = "":ED = 1: GOSUB 97
      0
2210  IF  LEN (A$(CT)) < 1 THEN 1
      450
2220  HOME
2230 RX =  VAL (A$(CT)): IF RX <
      1 OR RX > RC THEN  PRINT  CHR$
      (7): GOTO 2200
2240 RR = RX
2250  GOSUB 220: GOTO 1450
2260  REM PRINT SCREEN DATA
2270  IF RR = 0 THEN 1410
2280  HOME : VTAB 19: PRINT "TURN
       ON YOUR PRINTER! THEN PRESS
      [RTN]";: NORMAL : GET A$
2290  HOME : PRINT "PRINT TEST PA
      TTERN FOR ALIGNMENT?": PRINT
      : HTAB 10: PRINT "[Y]ES OR [
      N]O";
2300  GET A$: IF A$ <  > "Y" AND
      A$ <  > "N" THEN 2300
2310  IF A$ = "N" THEN  PRINT : GOTO
      2390
2320 D1 = 1: PRINT : PRINT D$;"PR
      #";PR: FOR D = 1 TO NF: PRINT
       SPC( HT(D) - D1): IF HT(D) <
       > H2(D) THEN  PRINT NN$(D);
       SPC( H2(D) - (HT(D) +  LEN
      (NN$(D))))
2330  PRINT  LEFT$ (TE$,LN(D));: IF
      VT(D) = VT(D + 1) THEN D1 =
      H2(D) + LN(D): NEXT : GOTO 2
      360
2340  PRINT : IF VT(D) + 1 < VT(D
       + 1) THEN  FOR S3 = VT(D) +
      2 TO VT(D + 1): PRINT : NEXT
2350 D1 = 1: NEXT
2360  PRINT : PRINT D$"PR#0": HOME
      : PRINT "DO YOU WANT ANOTHER
       TEST?"; CHR$ (8)
2370  GET A$: IF A$ <  > "Y" AND
      A$ <  > "N" THEN 2370
2380  IF A$ = "Y" THEN 2320
2390  PRINT "DO YOU WANT TO PRINT
       THE WHOLE FILE"
2400  GET A$: IF A$ <  > "Y" AND
      A$ <  > "N" THEN 2400
2410  IF A$ = "Y" THEN BZ = 1: FOR
      ZB = 1 TO RC:RR = ZB: PRINT
      : PRINT D$"PR#0": GOSUB 230:
       VTAB 24
2420 D1 = 1: PRINT : PRINT D$"PR#
      "PR: PRINT : PRINT : FOR D =
      1 TO NF: PRINT  SPC( HT(D) -
      D1): IF HT(D) <  > H2(D) THEN
       PRINT NN$(D); SPC( H2(D) -
      (HT(D) +  LEN (NN$(D))))
2430  PRINT A$(D);: IF VT(D) = VT
      (D + 1) THEN D1 = H2(D) +  LEN
      (A$(D)): NEXT : GOTO 2470
2440  PRINT : IF VT(D) + 1 < VT(D
       + 1) THEN  FOR S3 = VT(D) +
      2 TO VT(D + 1): PRINT : NEXT
2450 D1 = 1: NEXT
2460  IF BZ THEN  NEXT ZB:BZ = 0
2470  PRINT : PRINT D$"PR#0": GOTO
      1450
2480  REM DELETE A RECORD
2490  IF RR = 0 OR RC = 0 THEN 14
      10
2500 A$ = "YOU ARE ABOUT TO ERASE
       A RECORD":A2$ = "DO YOU WIS
      H TO PROCEED?":AX = 0: GOSUB
      1170: IF AN$ <  > "Y" THEN 1
      450
2510  IF RR = RC THEN RC = RC - 1
      :RR = RC: GOSUB 220: GOSUB 3
      10: RETURN
2520  HOME :A1 = RR:RR = RC: GOSUB
      220: HOME :RR = A1
2530 RC = RC - 1: IF RC < 0 THEN
      RC = 0
2540  GOSUB 310
2550  RETURN
2560  REM EXIT PROGRAM
2570  IF  NOT PP THEN 2590
2580  PR# PR: FOR H = 1 TO 36: PRINT
      : NEXT H: PR# 0: CALL 1002
```

```
2590  REM
2600  TEXT : HOME : PRINT : PRINT
      CHR$ (4)"RUN HELLO.DB"
2610  REM  STATUS DISPLAY
2620  PRINT  CHR$ (7): POKE 772,2
      55: GOTO 1450
2630 RT = 450:NF = 17: DIM VT(18)
      ,HT(18),NN$(18),M$(18),A$(18
      ),H2(18),LN(18),BY(18)
2640  HOME : VTAB 3: PRINT "WHAT
      IS THE NAME OF THE FILE": INPUT
      N$: IF N$ = "" OR  LEFT$ (N$
      ,1) < "A" THEN 2640
2650  ONERR  GOTO 2750
2660  PRINT D$"VERIFY"N$: POKE 21
      6,0
2670  GOSUB 340
2680 PR = 1
2690 OF$ = D$ + "OPEN" + N$ + ",L
      ":RF$ = D$ + "READ" + N$ + "
      ,R":WF$ = D$ + "WRITE" + N$ +
      ",R":CL$ = D$ + "CLOSE": FOR
      D = 1 TO 40:TE$ = TE$ + "X":
      NEXT
2700  GOSUB 270:RC = RO:ID = IO
2710  DIM S$(RT)
2720  VTAB 2: GOSUB 1370: VTAB 18
      : GOSUB 1370: POKE 34,18
2730 RR = RC: GOSUB 220: GOTO 145
      0
2740  REM    FILE NOT ON DISK
2750  POKE 216,0: TEXT : HOME
2760 EM$ = N$ + " IS NOT ON THIS
      DISK"
2770  PRINT G$G$: VTAB 5: HTAB 20
      -  LEN (EM$) / 2: PRINT EM$
2780  VTAB 10: HTAB 4: PRINT "PLE
      ASE INSERT CORRECT DATA DISK
      "
2790  VTAB 12: HTAB 3: PRINT "RET
      URN TO CONTINUE OR ESC TO EX
      IT ";: GET A$
2800  IF A$ =  CHR$ (27) THEN  HOME
      : END
2810  IF A$ =  CHR$ (13) THEN  RUN

2820  PRINT G$: GOTO 2790
```

CHECKSUMS

```
10   - $BADD      400  - $7980      800  - $CE0A      1200 - $ADBF
20   - $9B13      410  - $903C      810  - $E0C3      1210 - $732C
30   - $4D3B      420  - $C453      820  - $40A8      1220 - $A320
40   - $AD92      430  - $7767      830  - $45A9      1230 - $1F72
50   - $C899      440  - $11B4      840  - $ECE4      1240 - $F346
60   - $FF65      450  - $1C1B      850  - $E9C3      1250 - $57D2
70   - $A3BF      460  - $EC21      860  - $9E6A      1260 - $0B7B
80   - $A900      470  - $2D01      870  - $1F17      1270 - $AC1A
90   - $924D      480  - $309D      880  - $08F6      1280 - $A8F4
95   - $2608      490  - $0E23      890  - $33D7      1290 - $B4F2
100  - $2D63      500  - $5C04      900  - $D3D3      1300 - $707E
110  - $C342      510  - $F8B8      910  - $8AB8      1310 - $0C68
120  - $5349      520  - $1BE0      920  - $D52F      1320 - $B274
130  - $B595      530  - $4741      930  - $BCA4      1330 - $C3B2
140  - $8931      540  - $BF4C      940  - $D1D0      1340 - $4673
150  - $BA4C      550  - $951F      950  - $F02F      1350 - $7828
160  - $AE4A      560  - $F9B1      960  - $A654      1360 - $B29C
170  - $F5E4      570  - $E2CD      970  - $055B      1370 - $E962
180  - $411A      580  - $6E1A      980  - $FABF      1380 - $88AF
190  - $308B      590  - $4490      990  - $3A8D      1390 - $4ADB

200  - $836B      600  - $620D      1000 - $7CB6      1400 - $BDC4
210  - $254D      610  - $1F29      1010 - $5113      1410 - $660D
220  - $22E4      620  - $7185      1020 - $926F      1420 - $8366
230  - $64E9      630  - $76B3      1030 - $8F43      1430 - $A562
240  - $9844      640  - $F7C6      1040 - $28A0      1440 - $F839
250  - $CDA1      650  - $3430      1050 - $B02C      1450 - $11E6
260  - $ED69      660  - $B4BC      1060 - $D38E      1460 - $C053
270  - $90E2      670  - $23B4      1070 - $2E15      1470 - $780B
280  - $3AF0      680  - $D72B      1080 - $B42A      1480 - $5433
290  - $7BE5      690  - $403B      1090 - $EE87      1490 - $E19C
300  - $DC22      700  - $CA1C      1100 - $00B3      1500 - $3720
310  - $0396      710  - $93EE      1110 - $9F3C      1510 - $8B9D
320  - $A083      720  - $A267      1120 - $92BC      1520 - $F3C7
330  - $174F      730  - $B482      1130 - $D4B8      1530 - $EE83
340  - $B896      740  - $0D01      1140 - $676D      1540 - $A6CA
350  - $C63F      750  - $2951      1150 - $050C      1550 - $AF79
360  - $D130      760  - $0A78      1160 - $8C80      1560 - $B05F
370  - $E01B      770  - $3CFD      1170 - $3706      1570 - $CF58
380  - $45B5      780  - $245C      1180 - $1F0A      1580 - $A117
390  - $E22D      790  - $EE27      1190 - $905F      1590 - $63BC

1600 - $A40E      2200 - $8DB7
1610 - $67FC      2210 - $9C8F
1620 - $4137      2220 - $60BF
1630 - $AB94      2230 - $50BC
1640 - $69D1      2240 - $C223
1650 - $A86D      2250 - $92CB
1660 - $1178      2260 - $815F
1670 - $DCD6      2270 - $3F3F
1680 - $8D05      2280 - $8085
1690 - $AD7F      2290 - $8DD4
1700 - $58A8      2300 - $B6B5
1710 - $32C3      2310 - $FB0D
1720 - $6778      2320 - $3F08
1730 - $8313      2330 - $EF94
1740 - $40DD      2340 - $C7AB
1750 - $00D2      2350 - $4008
1760 - $F336      2360 - $B8E6
1770 - $632D      2370 - $13A5
1780 - $ECF8      2380 - $73F7
1790 - $0CA6      2390 - $C7DA

                  2400 - $7F61
                  2410 - $DB03
1800 - $843B      2420 - $BCEF
1810 - $0011      2430 - $A1A8
1820 - $9ACA      2440 - $5F67
1830 - $F38C      2450 - $6604
1840 - $BB01      2460 - $314F
1850 - $8AE2      2470 - $6548
1860 - $9512      2480 - $2489
1870 - $267E      2490 - $73C1
1880 - $6900      2500 - $4FA1
1890 - $BF2C      2510 - $47C4
1900 - $D7C2      2520 - $8D0B
1910 - $1506      2530 - $0BDE
1920 - $CC9E      2540 - $33D6
1930 - $A0D3      2550 - $6995
1940 - $DC87      2560 - $2A34
1950 - $BC0F      2570 - $F9BC
1960 - $E0C4      2580 - $1FC3
1970 - $A30B      2590 - $EC05
1980 - $6EF8
1990 - $4796      2600 - $B73E
                  2610 - $15B9
                  2620 - $7FD7
                  2630 - $B18A
2000 - $0961      2640 - $B0DB
2010 - $6992      2650 - $B206
2020 - $BD8D      2660 - $C0BE
2030 - $7B3B      2670 - $667A
2040 - $EA2B      2680 - $5D82
2050 - $79C0      2690 - $3E9F
2060 - $9A1A      2700 - $F0F3
2070 - $6BBA      2710 - $D808
2080 - $1A35      2720 - $B744
2090 - $0F10      2730 - $AF6D
2100 - $5EB0      2740 - $D228
2110 - $A5AD      2750 - $B160
2120 - $E360      2760 - $B48E
2130 - $5C65      2770 - $F8AE
2140 - $C7A1      2780 - $35FB
2150 - $2069      2790 - $7AA0
2160 - $B5A8
2170 - $7CFB      2800 - $F5FB
2180 - $4260      2810 - $1BD7
2190 - $6E00      2820 - $E529
```

Softkey For Mask Of The Sun

By John J. Liska

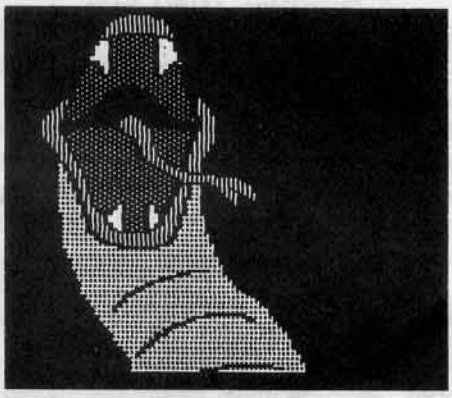
MASK OF THE SUN,
Ultrasoft, Inc.
24001 SE 103rd St.
Issaquah, WA 98027

REQUIREMENTS: Apple][Plus
One disk drive (two are preferred)
Mask of the Sun program disk
DOS system master
One blank disk

Now you can unlock one of the most maddening adventures you have ever attempted to solve-

Mask Of The Sun
By Ultrasoft

In this diabolical foray, your quest is to retrieve the legendary Mask of the Sun and live to tell the tale. You will encounter many things along the way which will defy common sense and your own intuitive logic, yet must be met.



The basic procedure is the same as the modified COPYA/DOS combination used to unlock ZORK (Computist #1). The only change to DOS seems to be the use of non-standard end marks on tracks $03 through $22 and incorrect checksums on the DOS tracks.

The first problem is to get the information off the original disk and onto the copy disk, here is how:

1) Boot the 3.3 Master Disk.

PR#6

2) Clear program in memory and initialize the front side of the disk you wish to copy to

FP
INIT SIDE A

3) Drop into the monitor

CALL-151

4) Modify DOS so that it ignores end marks and checksums.

B92D:18 60
B989:18 60

5) Put back in the system master and use COPYA to copy SIDE B to the back side (the side you **didn't** initialize in step 2).

RUN COPYA

6) When finished with the back side, use FID to copy every file (using the wildcard (=) and prompting features of FID) from side A except "LL(V27)" to the front side of the freshly INITialized disk.

BRUN FID

We now have all the files copied except the one that checked for copy protection. In order to make this disk work, we have to make a start up file.

7) Type in this start-up program and save it to the front side of the copy.

10 PRINT CHR$(4) "EXEC DISK"
SAVE SIDE A

That's it! Whenever you boot, you will get a FILE NOT FOUND message, but that is O.K. The reason for this is because the text file named DISK is trying to BRUN LL(V27), but since it isn't on this disk, DISK just goes on to the next command.

You now have a copy of Mask of the Sun on a normal disk for you to examine until you bust (or solve the adventure, whichever comes first).

Now, would someone please tell me how I can get past the ***BLASTED*** snake!

Upper & Lower Case Output For Zork

By Brian Burns

Requirements: Apple //e or][plus with lower case
Sector editor such as DiskEdit or Disk Zap
Any Infocom Adventure (not just Zork)

If you own an Apple //e, or have lower case on your II or II Plus, you are entitled to the following modification for Infocom adventures. This modification allows the game to display all of its messages in upper and lower case instead of all capital letters. If you didn't know already (I doubt you did), all of the text in any of Infocom's adventures is stored on the disk as upper and lower case (not in ASCII form, but as 3 characters compacted into 2 bytes). However, since not ALL Apples have lower case display as does the Apple //e, the lower case text must be converted to upper case before being printed to the screen. This is accomplished by a short routine which I will show you how to remove from the disk.

1) Make a copy of your Infocom game with the Softkey for Zork in Hardcore COMPUTIST #1, or the Softkey for Witness in Hardcore COMPUTIST #4. (These are exactly alike and either Softkey will work on any Infocom game).

2) After making the copy, run your sector editor program and read track 2, sector 9 of the disk you just made.

3) Look at bytes 4C through 56 (with older versions of ZORK I and II this area has been moved to bytes 9D through A8). They should contain the following....

C9 60 90 07 C9 80 B0 03 38 E9 20

4) Replace this code eleven EA's.

5) Write this sector back to the disk.

6) Read track 2 sector 7.

7) Change byte 9D from 3F to FF and write it back out.

8) Read track 2 sector B and change byte FC from 01 to 02.

9) Write the sector back out and you're done!

Explanation of Procedure

The first step is necessary for two reasons. First, most sector editor programs only work on normal (or normalized) disks. Second, it is NOT a good idea to make changes to your original disk (that is why it is write-protected).

Steps two through five inactivate the routine that converts the letters to uppercase.

Steps six and seven stop the room and score information at the top of the screen from being printed in inverse. This is necessary because when lowercase letters are inversed (ANDed with $3F), they become meaningless garbage.

Steps eight and nine alter the text window so that there is a blank line between the information at the top of the screen and the text of the game. This just helps to distinguish that information from the rest of the game.

I think you'll find upper and lowercase much easier on the eyes. Have fun.

For our U.S readers:

When ordering back issues, please note that we ship most orders United Parcel Post.

Continued from page 4

are not really hackers but who have one the programs in question, can actually follow along cook-book fashion and come out the other end with the desired results. Thanks!

I also very much appreciate the responses to most of the letters published in the Input section. One of the most frustrating things in most columns of this type is to have a reader ask a question in print, and then find one must wait three or four months (or more) to get even a preliminary answer. Now, if you would just increase the size of the Input section...Oh well, can't have everything at once.

A personal preference would be fewer softkeys for games and more for applications programs, but of course, I realize you have readers of all interests. Like others, I too have a wish list of programs for which I have been hoping to see softkeys. My list includes Sensible Speller, Visiblend, Basic Accounting and The Accountant among others. A suggestion: If you were to print the Top 30 or Top 50, it might give the hackers out there an idea of which programs are most in need of softkeys.

Finally, a need of my own: Does anyone know of a relatively inexpensive, programmable add-on keyboard. I am thinking of something like the VIP Key-Whiz, which is really what I want. But the price of VIP is so outrageous, it is out of the question. I thought a board called Keyport 717 might be the answer (it sells for $175 as opposed to $450 for VIP), but all of its functions are loaded into the Apple's memory and it requires a real programmer's knowledge to adopt to a particular application program. The VIP, on the other hand, puts the functions the user wants into its own RAM and requires no knowledge of programming. If only its price were not so high! Any ideas?

Once, again, thanks for the great improvement in the magazines. Keep up the good work!

Thomas E. Militello, M.D.
Rancho Palos Verdes, CA

Dr. Militello: We think your suggestion that we print a "wish list" of softkeys is an excellent one. In fact, because of your suggestion and the tremendous response we received from our readers after requesting softkeys in Hardcore COMPUTIST no. 5., we can see no reason to delay printing the "Most Wanted List."

You can find this list on page 4 of this issue.

Our staff would like to thank you for your suggestions, encouragement and the compliments on Hardcore COMPUTIST no. 5.

Worried About a Crash

I enjoy your magazine very much and look forward to every issue.

I have Flight Simulator II and cannot make a backup copy. I have tried Copy II +, Nibbles Away II and Pirates Friend. None of these will copy it. Do you know something about this program?

Tom Peragine
Pompano Beach FL

Mr. Peragine: Sorry, we do not as yet have a means to backup the excellent Flight Simulator II. We have included FS-II in our new "Wanted List" which you can find on page 4 of this issue.

Another Wish List

The latest Hardcore was the best ever. I especially appreciated the boot-code tracing.

I would appreciate your giving attention to the following: 1) Sensible Speller seems to have everyone baffled. Why? 2) Broderbund is especially hard to copy. One gets 0 track errors and even nibble counts, but "Arcade Machine" won't boot. It will run if the Master is booted first, but... 3) Is there any way you could interview "Mr. Xerox" or Krac-Man? Their talents for breaking stuff all the way down to FID-able files is amazing. 4) You might mention that "Cracking techniques" featuring material by the above is available through Pirates Harbor. However, I have not found it very helpful because it assumes I know DOS better than I do. 5) I am sure you have seen the Fast-Copy program (called Pirate's Friend, Fastass Copy, etc.) that copies 5 tracks at a time and works on many copy-protected programs. I believe it is in the public domain. If you have a copy, could you disassemble it and give us an overview of the logic of such a program.

I would also like to know the logic of the "parameters" one changes in Copy II + or Locksmith. I know there are address or data field headers, but how does one discover which to change and know why?

Continued on page 31

Advanced Playing Techniques

Star Maze

Contributed by Ferrell Wheeler

Star Maze
Sir-Tech Software

It seems the pinwheel and the wall snail are extremely difficult to shoot so, instead of wasting a bomb on them, here's a method for destroying them and anything else in the game at no risk:

1) Drive repeatedly straight into a wall at a fairly slow speed until the ship is completely inside the wall.

2) Once inside, turn around and position the ship so that only enough of its nose is sticking out to still allow for firing.

When in this position, nothing can hit the ship, not even when the snail is walking on the same wall the ship is in!

You can even use this technique to go through walls, but this seems to use up a lot of fuel.

The Wizard and the Princess

Contributed by Donald Oliveau

The Wizard and the Princess
Sierra On-Line Systems

If the player "dies" in this game, there's a way to revive him or her. When the computer asks you," Do you wish to play again?" after the "death" has occurred, answer "N" for no and then press "RETURN" **twice**. This restores you to life at the spot you were "killed" with all your inventory intact.

Wizardry: Proving Grounds of the Mad Overlord

Sir-Tech Software

It seems as though a lot of Wizardry players already know how to create a very powerful bishop, but here is the technique anyway:

1) Take the bishop into the maze and then camp.

2) Inspect the bishop's character and choose the identity ("I") option.

3) Keep trying to identify item #9 until the word "success" appears at the bottom of the screen.

4) The bishop should now have **100,000,000** experience points.

Adventure Tips

Cranston Manor . . . . . .

Sierra On-Line Systems

Knights in armour do not like rodents.
Drown the computer to get the platinum sphere.

Time-Zone . . . . . .

Sierra On-Line Systems

There is more than one piece of jade
in the Chinese Garden.

The Wizard and the Princess . . . . . .

Sierra On-Line Systems

Try kissing an amphibious creature.
The giant seems to be fond of music.
Don't waste time on peasant women.

Mask of the Sun . . . . . .

Ultrasoft

While under the Temple of the Sun, be sure to
wear the mask before every move.
The cat may need its head.
It's tough to shoot reptiles in the dark.

Deadline . . . . . .

Infocom Inc.

Dig for evidence in the rose garden.

Continued from page 7

This will tell you the loading location of the last BLOADED file. It will appear "backasswards" with the low byte first and the high byte second (i.e., 00 08 is equivalent to $0800). Do a 800l to list the first screen-full of the program.

Upon examination, you see that the accumulator is loaded and then stored in a location within DOS. After this happens a few times, you will see a JSR (jump to subroutine) at $B7B5. Now you may ask, "What is all this stuff?"

On-Line is using the second stage of DOS to load in the game of Mouskattack. You are looking at the first stage (of three stages) of the load and are actually tracing the boot as described in earlier issues of Hardcore COMPUTIST, but using normal DOS.

The listing will appear as follows:

```
0800- LDA #$00
0802- STA $B7EB     ;Volume 0 matches anything
0805- LDA #$01
0807- STA $B7F4     ;Command code, 1=Read
080A- LDA #$18
080C- STA $B7EC     ;Track # to start loading from
080F- LDA #$03
0811- STA $B7ED     ;Sector # to start loading from
0814- LDA #$95
0816- STA $B7F1     ;High byte of page to load to
0819- LDA #$00
081B- STA $B7F0     ;Low byte of page to load to
081E- LDA #$03
0822- STA $14       ;Sector counter in zero page
0824- LDA #$B7
0826- JSR $B7B5     ;RWTS sector read routine
0829- BCS $0822     ;Branch to $822 if Error
082C- INC $B7F1     ;Increment page to load
082F- INC $B7ED     ;Increment sector to load
0831- DEC $14       ;Decrement sector counter
0833- BNE $0822     ;If sector # <> 0 branch to $822
0835- JMP $9500     ;Jump to $9500
```

Starting with track 18, sector 3, the data is being loaded into location $9500.

The sector number and memory page in which the data will be stored is incremented and the process continues until sector F of track 18 is reached. Jump to location $9500 to start the next stage of the load. If you jump to the monitor at location $835 instead of $9500, you can examine the next stage of the load. Do this by typing

835:4C 59 FF N 800G

At $9500, the same thing happens again but in a larger perspective. The codes become more obscure and difficult to follow, so I won't list them here.

But now you ask, "What was the purpose of this exercise?"

By examining the code at $9500, you can find the starting page at which Mouskattack loads. You can even find the starting location! Can you see where?

The starting location of the loading program is at $A00 and the starting location of the loaded program is at $5300. This is the reward for all your labor!

Since the load code lives at $9500 and DOS occupies $9D00 and up, Mouskattack must live from $A00 to $94FF, with a starting location at $5300. Not mentioned is the third load which overwrites the demo code. This load is the only part of the disk that is really protected. Our COPYA copy will work up to this load, but not past it, for the game.

Keep in mind that the code from $900 to $95FF is not destroyed by a slave boot. Since Mouskattack lives within this area, it then becomes easy to crack. Here is the procedure:

1) Boot your original copy of Mouskattack.

2) After the demo and, when the prompt HOW MANY PLAYERS? appears, insert a blank, initialized slave disk.

3) Hit "Reset." It does not matter if you have an old-style monitor or not. With an autostart monitor, your slave disk will boot.

4) Enter the monitor.

CALL-151

5) To enable you to save larger binary files to disk, type:

A964:FF

6) Type in

9FD:4C 00 53

7) Save the program

BSAVE MOUSKATTACK,A$9FD,L$8B03

8) Have fun with the game!

Remember how I mentioned that the load of Mouskattack was rather slow? This is due to the method in which sectors are read in from the disk.

Also, the sector number increments along with the page number loaded to. If On-Line had started with the high page number, instead of the low page number and read the sectors in decreasing order, instead of incrementing order, the load would have taken a fourth of the time, providing the manufacturers had used normal DOS 3.3 skewing.

This is the logic that the fast-loading DOS's have taken in order to increase loading and save time.

Bugs In Hardcore COMPUTIST #5

page 8 — **Screenwriter II Softkey**

Step 5 should read:
5) Load the second file
BLOAD EDITOR PART1.OBJ0

. . . . . .

page 15 — **Cracking The Egbert II Communications Disk**

Step 6 should read:

6) Make the moved bootcode JuMP into the monitor after loading track 0, sector 0
•86F8:4C 59 FF N 8600G

page 18

Step 35 should read:

35) Fix the same catalog patch in the RCV program
•BLOAD RCV
•4302:20
•BSAVE RCV,A$4000,L$58B

page 19 - first column

The two lines following the sentence;"Type in the following in immediate execution mode:"; should read:

```
]F2=PEEK(175)+PEEK(176)*256-8:FOR
A=1 TO 4:POKE F2+A,0:NEXT
]SAVE RTTY
```

Likewise, the two lines following the sentence;"Again in immediate execution mode type:"; should read:

```
]F2=PEEK(175)+PEEK(176)*256-8:FOR
A = 1 TO 4:POKE F2+A,0:NEXT
]SAVE ECW
```

. . . . . .

We would like to thank Dennis Goodwin and Dr. Keith Goldstein (author of "Cracking The Egbert II Communications Disk") for calling these errors to our attention.

Surface Rates For Our Foreign Subscribers!

To better serve you, we now offer both surface and airmail subscriptions for foreign subscribers. Current rates:

	Air	Surface
1 yr.	$60.00	$40.00
2 yrs.	$116.00	$76.00
3 yrs.	$170.00	$110.00

To change your current airmail subscription to a SURFACE subscription, complete and mail the form on **page 11** to:

Hardcore COMPUTIST
P.O. Box 44549X
Tacoma, WA 98499

Please allow four to six weeks for change of subscription to begin.

Continued from page 28

Keep up the great work until the insanity of protection is ended.

John Langlois
Milton WI

Mr. Langlois: The Pirate's Friend copy program you mention seems to bear an amazing resemblance to the early versions of Copy II Plus. As such, it is doubtful whether this program is in the public domain, although it may seem to be because of the number of copies that are floating about.

You have come up with some good suggestions for future Hardcore COMPUTIST articles. Although we can't make any promises, we will try to work some of your ideas into articles for the magazine.

An Elite Adjustment

The documentation accompanying the RANA Elite I disk drives claims the speed of these drives is not adjustable, as are most other Apple-compatible drives. However, I have found that on my RANA I the speed can indeed be adjusted. Here's how:

1) Remove the disk drive cover. 2) The speed adjustment potentiometer is labeled R7 and is located in the middle rear of the disk drive analog board. A clockwise adjustment of this potentiometer will increase the drive's speed and a counter-clockwise adjustment will slow the drive speed.

Donald L. Russell
Faribault MN

Mr. Russell: Thanks for your tip. We might only add that if you make this adjustment you will void your warranty, but that is something most people do within 10 minutes of acquiring a new computer product anyway.

Whiz Kid

By Ray Darrah

Yes, here I am this month with another protection trick. Once again, this procedure "Locks Up" a program, but as long as it is on a normal disk, it can be copied.

RESET VECTOR

By now you've probably heard many times about a program making the reset vector do one thing or another. So here is an explanation of how it works.

It all starts when you press the key marked "RESET." Instead of going through the keyboard latch at memory location 49152 (or -16384 or $C000), the reset key is connected directly to pin 40 of the Central Processing Unit (CPU). This is the long chip placed at a right angle to the other chips on your circuit board.

Whenever pin 40 is grounded (i.e. the "RESET" key is pressed or the computer is turned on), the microprocessor (CPU) stops whatever it was doing and immediately runs the program that memory locations 65532 and 65533 ($FFFC and $FFFD) point to. These locations (and the program they point to) are in ROM and therefore, cannot be changed by a program.

RESET PROGRAM

When the CPU is first turned on, it acts just like pin 40 has just been grounded (i.e. "RESET" key was pressed). Because of this, when the "RESET" key is pressed, the program that the CPU runs has to determine if the computer was just turned on or not.

To solve this problem, Apple came up with a little system that works nicely. They decided to dedicate three RAM memory locations to the task of processing the "RESET" key. The first two locations tell the program where you want it to go when a "RESET" occurs and the third ensures that the location, pointed to by the first two bytes is valid.

If for some reason the third byte isn't the correct complement of the second byte (very likely to occur when just turned on because the RAM holds random numbers), the program assumes that the computer was just turned on and runs through the power up sequence which stores good values in the three locations, clears the screen, prints "APPLE][" and tries to boot a disk. On the other hand, if the third byte is correct, then it merely executes the program pointed to by the first two bytes.

This powerful system of Reset handling enables you to "tell" (via POKE) this Reset program to execute a program whenever the key is pressed. Or you could fool the program by making the powerup (third) byte incorrect. The program would then assume the computer to be just turned on and run through the above mentioned powerup sequence.

Now that you have an understanding of the "RESET" vector, all that is left to explain is how to make the third byte the correct complement of the second, so the computer won't boot-up when "RESET" is pressed. Luckily, the monitor ROM has a subroutine that does just this. Otherwise, it would be quite difficult to do from BASIC.

The procedure: Just POKE in the numbers that point to the subroutine you wish to execute and then do a CALL-1169. It's that easy! Try this:

1) Enter the monitor.

```
CALL-151
```

2) Type in this short program.

Continued from page 30

300:A0 60 A9 18 20 A8 FC AD 30 C0 88 D0 F5 20 EA 03 4C 03 E0

3) Get back into BASIC.

E003G

4) Point the reset vector to our new program.

POKE1010,0:POKE1011,3

5) Make the third byte (location 1012) valid.

CALL-1169

6) Press "RESET."

As the above example points out, the three locations are 1010, 1011 and 1012. It also shows that the routine which validates the third byte is at -1169.

If you don't want to go through the hubub of writing a machine language program, you could just fool the computer by changing the third (or powerup) byte. Try this:

1) Alter the powerup byte by one.

POKE 1012,ABS(PEEK(1012)-1)

2) Press "RESET."

Wow! It acts like you just turned it on.

INTEGER CARDS

All those softkeys that say "Reset into the monitor" work because the ROM on those cards have a totally different reset program. This different program simply executes the monitor program which, needless to say, just puts the computer in the monitor.

See you next issue!

Advertiser's Index

☐ 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

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

Enclose $4.00 for each EAMON scenario volume that you order or see our special EAMON collectors offer.

Special Collector's Offer

Every EAMON volume (including Utilities) ALL for only $140

☐YES! Send me the entire EAMON collection.

Make checks payable to:
Computer Learning Center
P.O. Box 45202
Tacoma, WA 98445

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. **HC7**
Adventure scenarios are packed on double-sided disks. Minimum Order: 2 Volumes. ($8.00)

Name _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone _____

☐ VISA ☐ MC #_____ Exp Date _____

Signature _____

U.S. funds only. Make checks payable to: Computer Learning Center

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 is in your hands.

Where The Adventures Never End!

Avoid the I.R.S.

(Inadvertently Ruined Software)

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.**

NEW!

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

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

Copy II PC (IBM)

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

We are the backup professionals. Instead of diluting our efforts in creating a wide variety of programs, we specialize in offering the very best in backup products. So, protect your software investment, **before** the I.R.S. gets you.

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).