PARAMETERS!

$3.50

hardcore computing

Vol. 1 Issue 3

HYPER DOS
500% faster than DOS 3.3

Interview with
MIKE MARKKULA
on copy protection

Softkey 4
BOOTCODE TRACING

REPRINT #1

Organization a problem?

Look what Sensible Software is doing for you...

DISK ORGANIZER

Here is a fast and friendly utility to organize the files on your diskettes using DISK ORGANIZER. You can order the files on a diskette not only alphabetically, but in any order you want. TITLES can be entered, describing a file or group of files on the diskette. These TITLES may be entered in normal, inverse, flashing, and lower case letters. All work is done on a RAM version of the Directory; so if you decide the changes you made are not correct, you can simply abort the process and keep the old catalog. Since DISK ORGANIZER works on a RAM version of the Directory, disk access is kept to a minimum; hence, DISK ORGANIZER is extremely fast.

The following functions are also included:

- DYNAMIC DISPLAY of all file names in the Directory.
- RENAMING the "HELLO" program.
- A SIMULATED CATALOG to show the modified Directory before it is written to the Diskette.
- ALPHABETIZING the file names.
- UNDELETING deleted files.
- PURGING deleted files.
- RENAMING files (with the same character input options as TITLING.
- LOCKING and UNLOCKING (some or all) files.
- DELETING files.
- DELETING DOS for increased data storage.
- A powerful SMART KEY to automatically locate the next valid file name for any specified operation.

DISK ORGANIZER ... Apple II or Apple II Plus with 48K and one or more Disk Drives ... **$30.00**

Is inter-disk travel and intra-disk file manipulation causing difficulties?

Then SUPER DISK COPY is for YOU!

Super Disk Copy is much more than just another copy program. It's practically a 'mini-DOS' in itself. Super Disk Copy is completely menu driven and works with DOS 3.1, 3.2, and 3.3. Files may even be transferred from one DOS to another (example: DOS 3.3 to DOS 3.2). Super Disk Copy makes the conversion to DOS 3.3 less painful. After seeing Super Disk Copy, you will agree it is the best copy program on the market! Among the many features are:

- Wildcards may be entered anytime a file is used. They allow both automatic or user verification of the file handling. (Example: to copy file names that start with 'Apple', you would enter 'Apple = ').
- COPY single files (Applesoft, Integer, Text, Binary, Relocatable).
- COPY DOS.
- COPY ENTIRE DISK. Super Disk Copy uses a unique technique for fast copying.
- A BRUTE FORCE COPY made for Pascal, FORTRAN, and CP/M diskettes.
- An optional rearrangement of files so that they occupy contiguous sectors for improved access time.
- UNDELETE deleted files.
- A PLOT of disk usage.
- FIX file sizes. Make sure your valuable files are protected and unused sectors are available.
- DELETE DOS. Frees up extra space on your diskettes.
- ALPHABETIZE file names in directory.
- REPLACE illegal characters in file names.
- Optional 'INIT' of copy diskette.
- LOCK or UNLOCK files and much, MUCH more...

SUPER DISK COPY was one of only two programs to receive a rating of 100 in PRICE/USEFULNESS RATIO in THE BOOK OF APPLE COMPUTER SOFTWARE 1982. They said "This is a definite MUST HAVE utility package."

48K and DISK II required ... **$30.00**

For a complete catalog send $1.00, refundable with your first purchase.

If you have a large software collection and have problems locating specific programs or you need to be reminded of what they do...then you need MULTI-DISK CATALOG

Multi-Disk Catalog is designed specifically for keeping track of the contents of the APPLE diskette library. The resulting master catalog can be sorted, searched, and printed. Multi-Disk Catalog is entirely menu driven, easy to use, and very fast. Some of the unique features included are:

- Works with 13 and 16-sector formated diskettes (DOS 3.1, 3.2 and 3.3). Diskettes from either type DOS can be added to the same data-base used.
- You assign the volume identification number (range 1 to 999); there is no need to INIT each of your disks with a unique volume number.
- A 2 letter classification code can be added to each file name. The classifications are retained when the disk volume is updated.
- Multi-key sorting. Up to 3 keys may be selected for sorting the file names. The sort keys include disk number, file name, classification file type, and file size.
- Multi-Disk Catalog is written entirely in machine language so it is very fast.
- A powerful search mask can be used to find a file name or groups of file names.
- Multi-Disk Catalog supports special printer intertaces.
- Both sides of a diskette may be entered under the same disk number.
- Titles may be entered for each diskette and much more!

"There is no doubt about it, this is the best and most versatile master catalog program available"— THE BOOK OF APPLE COMPUTER SOFTWARE 1982.

48K and one Disk Drive required ... **$25.00**

Sensible Software

6619 Perham Drive Dept. H • West Bloomfield, Michigan 48033 • (313) 399-8877
Visa and MasterCard welcome, please add $1.25 postage and handling per diskette.

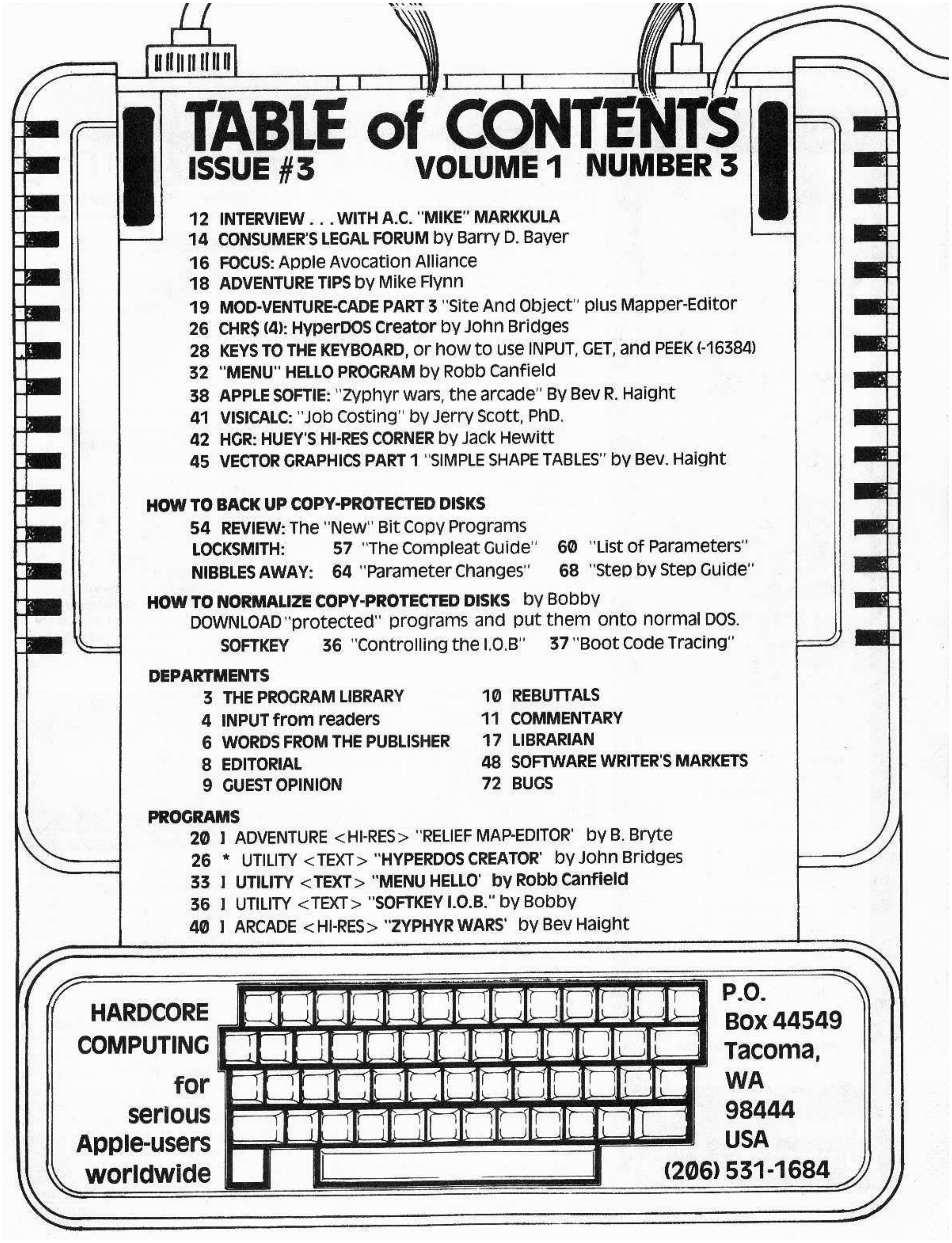


TABLE of CONTENTS

ISSUE #3 VOLUME 1 NUMBER 3

HARDCORE COMPUTING for serious Apple-users worldwide

P.O. Box 44549 Tacoma, WA 98444 USA (206) 531-1684

hardcore computing

Publisher, Editor:
Charles R. Haight

Associate Editor:
Bev R. Haight

Graphics Editor:
Carl Lawson

Editorial Assistant:
Julie Joringdal

Subscriptions, Distribution:
Beryl Flynn

Comptroller:
Karen Fitzpatrick

Columnists:
Barry D. Bayer
Bobby, of course
Jack Hewitt
Jerry Scott
Mike Flynn

Contributing Writers:
B. Bryte
Bev R. Haight
Charles R. Haight
John Bridges
Robb Canfield

Letters:
Allen Wyatt
Bill Parkhurst
Richard Cornelius
Tod Wicks
and many more . . .

COVER ART:
Greg Bellamy
Carl Lawson

Typeset via MODEM:
Graphic Services

Printed:
Peninsula Gateway

Published:
Softkey Publishing
P.O. Box 44549
Tacoma, WA 98444, USA.

volume 1
number 3

ADVERTISER'S INDEX

Community Computerist's Directory	5
Computer Hideout	56
D.R. Jarvis Computing, Inc.	47
Microware Distributing, Inc.	50
Omega Microware, Inc.	56, 58, 67
Sensible Software, Inc.	Inside front cover, 7
Soft CTRL Systems	15
Stellation Two	49
Sympathetic Software	71
Texas Ranch and Shoreline Systems	70

Address all editorial, advertising and subscription inquiries to the proper dept., HARDCORE Computing, P.O. Box 44549, Tacoma, WA 98444. Return postage must accompany all manuscripts, drawings, photos, disks, or tapes if they are to be returned. No responsibility can be assumed for unsolicited manuscripts. We suggest you send only copies.

Subscriptions

HARDCORE Computing is published quarterly by SoftKey Publishing. One year subscription rates (for 4 issues and 4 updates) are:

U.S.A.	$20
Canada	$29
Mexico	$32
S. America	$38
all others	$42

All foreign subscriptions must be in U.S. funds and prepaid. No credit cards, yet.

DOMESTIC DEALER RATES sent upon request, or call (206) 531-1684.

UPDATES will focus on the subject of Softkeys, "How to make backup copies of so-called uncopyable diskettes." Updates are available only to subscribers and are not sold in stores.

If HardCore is not published on schedule, your subscription will still be for 4 issues and 4 updates regardless of the time it takes to fulfill that order.

Other Words

ESQUIRE, Jan. 1982
"Secrets of the Software Pirates" by Lee Gomes

". . . Software houses saw Locksmith as the start of an open season on their products. They expressed their displeasure to the owners of the computer magazines, most of whom obliged by refusing to run Alpert's ads. When some Apple enthusiasts in Washington State heard about the boycott, they concluded it was nothing but censorship and another example of the magazines' ignoring the average Apple user to placate their advertisers. So they started their own publication, HARDCORE COMPUTING, which with its first issue came to play the role of iconoclastic underground magazine battling the industry's stodgy straight press."

TIME, Feb. 8, 1982
"Roaming Hi-Tech Pirates"

". . . HARDCORE COMPUTING, a small magazine in Tacoma, Wash., warns pirates about the latest technology that companies are using against them."

-- Personally, I was amazed at how TIME magazine confused and mixed up the subject of software piracy. In that small sidebar on page 61, they confused two quite different "pirates": the companies who pirate from each other, and users who make copies. Perhaps one day TIME will take time to print the user's side of this issue instead of the well-publicized complaints of a few softhouses who still admit that SOFTWARE IS A BOOMING BUSINESS!

HARDCORE Software Library

If you want the most recent version of a previously published program, or if you simply do not have the time or desire to type in the listing, you may be able to purchase the program from the Software Library.

To order any program, follow these steps:

1. See if it's available by checking the list shown below.
2. Write the charge for that program. For Applesoft BASIC programs, there may also be a compiled version available for an extra charge. Documentation, if available is extra. For Binary program, source code is probably available, so ask for it.
3. Total the charges, add in the disk/shipping/handling charge, and fill in the order blank. Include the check or money order in U.S. funds and mail to:

   HARDCORE SOFTWARE LIBRARY
   DEPT HC3
   P.O. Box 44549
   Tacoma, WA 98444

CHARGES LISTED IN THE ORDER FORM ARE FOR U.S. AND CANADA ONLY. FOR FOREIGN ORDERS, PLEASE ASK FOR THE PROPER SHIPPING CHARGES.

OUR DISKETTES ARE NOT COPY-PROTECTED AND WE ENCOURAGE YOU TO LET OTHER COMPUTERISTS MAKE COPIES OF THEM. BUT PLEASE ASK THEM TO HELP COMPENSATE THE AUTHORS BY TELLING THEM ABOUT THE HARDCORE HONOR ROYALTY SYSTEM:

FOR EACH PROGRAM COPIED, PLEASE SEND $1 TO THE PROGRAM'S AUTHOR IN CARE OF HARDCORE COMPUTING SOFTWARE LIBRARY, ESPECIALLY IF YOU APPRECIATED THE PROGRAM!

LANGUAGE	CODE	PROGRAM NAME	PROGRAM ALONE	DOCUMENTATION	COMPILED VERSION	CHARGE
		Updated versions				
A-soft*	U-ØØ1C	DiskEdit 2.7	$ 5.ØØ	$ 2.ØØ	$ \.\\	$___.______
A-soft*	U-ØØ2B	DiskView 2.Ø	$ 4.ØØ	$ 1.ØØ	$ \.\\	$___.______
		from Issue # 1				
Integer	S-ØØ1	Hi-Res Ink Blots	free	none	none	( )
A-soft	S-ØØ2	Hi-Res Ink Blots	free	none	none	( )
		from Issue # 2				
A-soft*	A-ØØ1	Artist's Easel	$ 3.ØØ	$ none	$ none	$___.______
A-soft	G-ØØ1	Amber's Ts	$ 2.ØØ	$ none	$ none	$___.______
A-soft	G-ØØ2	Text Invaders 2.Ø	$ 2.ØØ	$ none	$ 2.ØØ	$___.______
A-soft	G-ØØ3	Relief Mapper 1.Ø	$ 1.ØØ	$ none	$ 1.ØØ	$___.______
		from Issue # 3				
A-soft	G-ØØ4	Map Editor 2.Ø	$ 2.ØØ	$ none	$ 2.ØØ	$___.______
A-soft*	G-ØØ5	Zyphyr Wars 2.Ø	$ 1.ØØ	$ none	$ 1.ØØ	$___.______
A-soft*	U-ØØ3	Menu 2.Ø	$ 2.ØØ	$ none	$ none	$___.______
A-soft	U-ØØ4	IOB	$ 1.ØØ	$ none	$ none	$___.______
Assembly	U-ØØ5	HyperDOS 1.Ø	$ 3.ØØ	$ none	$ none	$___.______
					SUBTOTAL	$_____.______
		ADD Diskette, Shipping and Handling Charge				$ 6.ØØ
					TOTAL	$_____.______

Name ____

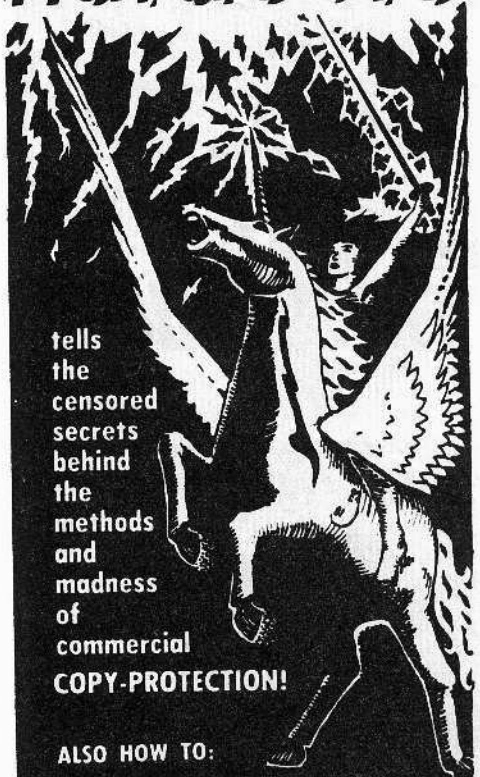
Address ______________________________________

City ______________________ State ______ Zip ____________

NOTICE: Prices can change with each new issue of HardCore Computing. You will be notified if the enclosed funds do not cover the new prices. Please allow 4 to 6 weeks for delivery. Be sure to make an immediate back up copy.

Assembly = source code   * = some machine code included in BASIC code.
Compiled A-soft version created by using TASC, by Microsoft.....

hardcore



tells the censored secrets behind the methods and madness of commercial COPY-PROTECTION!

ALSO HOW TO:

Market your software.
Copy-protect your disks.
Normalize altered D.O.S.
Use bit-copiers to make back-ups.

PLUS Game, Utility, Business, Educational program listings.

HARDCORE computing
Dept HC3
P.O. Box 44549
Tacoma, WA 98444

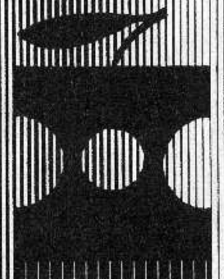
SUBSCRIPTION: $20.00 (U.S.A.)
$32.50 Mexico $28.50 Canada $42.00 others

NAME ________________________
ADDRESS ______________________
CITY ________________________
STATE ____________ ZIP __________

Sample copy: $5.00 — U.S.A. $7.00 — elsewhere

ISSUE 4 will be a...

SPECIAL on GRAPHICS



so... we are seeking: columns, articles, programs, news & reviews!

YOU CAN BE A HARDCORE WRITER!

for more info, see the inside back cover!

INPUT INPUT

Dear Sirs,
Please enter my subscription to the above address.
P.S
I would very much like back issues.
Stephen G. Wozniak
Los Gatos, CA

in bad taste

Dear Sir:
I agree with Nibble. The ad on the back of Update 2.1 is "too suggestive." Moreover it's in bad taste.
I do, however, appreciate the articles in your magazine.
Although it's nice to receive a timely "Update," they are going to be hard to keep track of. I'd just as soon you save them up and put them in a quarterly magazine.
Bob Britton
Spring, TX

blanketiblank

Gentlemen:
I can't stand still for George Blank's letter to you (p37, Hardcore 2.0). His putdown (how do you justify $20...) is unreasonable, but consider the source. It's a matter of content, not weight. I've learned much more real hard Apple lore from just two issues of Hardcore than from several years' worth of subscriptions to CC!
Mr. Blank comments on the lack of profitability in the software business. Let's look at profits vs. quality. The good stuff makes money. Ask Ken Williams (who apparently finds Hardcore pornographic) whether he makes money! The people who go bankrupt are those who don't (or can't) develop a solid, functional, well-engineered product which meets a real need, and is properly merchandized, distributed and supported. All the copy-protection in the world isn't going to save a poor product. And the availability of backup methods isn't going to kill a superior product (witness VisiCalc!).
Knowledge of the Apple II computer is essential to the development of good software. Hardcore meets a real need in disseminating vital information not addressed elsewhere. Keep it up!
Harry E. Brawley, Jr.
Weston, MA

tempting trap

Dear Mr. Haight;
I read with interest your magazine, Hardcore Computing and I have decided to respond in three ways: a subscription request, a letter to the editor, and a suggestion on a column to be used in Hardcore Computing.
I agree with your basic premises in your magazine, but in the future I would appreciate you getting down to the business of supplying more information about the internals of computing with an Apple Computer.
While it is obviously illegal to restrain trade by black-balling the advertisement of bit-copiers, I feel that your tendency to try to retaliate by accusing those people of dastardly deeds accomplishes very little, except getting them more excited about trying to stop you.
As the Apple computer showed the world by beating the big systems at getting information to the people sooner and more effectively, I think that the people who got us this far will continue (or others like them).
I applaud your continued efforts at getting the information about programs out to everyone. I agree, too, that the people who put out a great deal of effort to generate that information should be compensated for it.
I sincerely hope, however, that you do not fall into a tempting trap of assuming that anyone can decide what a fair compensation ought to be. No matter what the current social trends are, no one, absolutely no one, can determine what is right for someone else.
I believe the current copyright laws go a long way towards protecting the rights of both sides of the software problem. Reasonableness must prevail now in the implementation of that law. Perhaps there is room for modification so that consumers can be inside of the law, but I don't think the jury is in on that yet.
Sincerely,
Bruce Jorgensen
Richland, WA

likes 'n dislikes

Dear Editor,
I applaud:
**Hardcore Magazine,
**Bit Copy programs,
**Penguin Software for their new "no protection policy,"
**the President of Apple Computer for his statement on ending locked software in the industry,
**and all others in that vein.
I detest:
**censorship in the magazines,
**and those individuals and companies that try to force their attitudes about locked software on the end user's,
**Apple computer's new policy of "only over the counter sales."
In the end the competition of a completely non-censored competitive market-place will prevail. The sooner this happens, the more the computing public will benefit from better and more software/hardware.
I would like to see some in-depth articles about using hardware in making back-up

INPUT INPUT INPUT

copies: IE wiring disk drives back to back, loading programs in expanded RAM then dumping back to disks or other methods that you know.

Keep up the good work.
Dale Hurd
Yakima, WA

beyond belief

Dear Hardcore,

I recommend Sex-Rated for the Garbage Award. It is a do-nothing disk with crude graphics, about the 2nd grade, restroom-wall quality.

Next to my Apple, I like Sex best. Explicit pictures do not offend me either. But, this is beyond belief. Not only is it poor quality graphics, a minimum of animation, and in very poor taste, it unnecessarily exercises your disk drive to obtain the "Sounds of Sex." I give it a big fat -13 on the ratings.

John Bouchillon
St. Charles, MO

a buck a copy

Gentleman;

Here's $3 for letting me copy and use:

a. DiskEdit 2.3
b. DiskView
c. DataWrite

How can I get instructions on how to use these programs?

You gave these three programs to Dick Peschke, President of Apple-Dayton, several months ago when he visited you. During our Jan. meeting Dick told us about what you're trying to do, how, and why. Very impressive! Dr. John B. Matthews, our Vice-President, has also spoken out strongly in support of your activities and encouraged us to support you. I intend to do so.

I appreciate your publishing "unprotected" programs and trusting us to forward whatever you ask if we like the programs enough to copy and use them. That's a reasonable approach and the cost is certainly a bargain. I hope other software publishers will follow your lead.

I wish you continued success.
Sincerely,
G. Adams
WPAFB, OH

cp/m attitude

Sirs:

Enclosed please find a copy of an announcement that was printed in INFOWORLD Oct.19,1981. This is an indication of the attitude of the CP/M users toward copy-protection of the programs that we use.

I cannot believe that the CP/M users are more honest or intelligent than Apple users; however, they do insist that they can make as many back-up copies of a program, or modify it to suit their needs as necessary.

It is soon obvious to anyone using an Apple for business, as I do, that the 5 1/4 inch disks just do not do the job. Even a pair of 8 inch drives in single density is doing it the hard way.

Copy-protected programs do not allow the user to down-load to the 8 inch drives, so it is for this reason that all of my business software runs on my Apple under CP/M.

Possibly, if enough users of Apple DOS (Dumb Operations Systems) would let the producers know of their feelings, just as the CP/M users did, maybe it might do some good.

I am sending this letter and a copy of this announcement to each of the Apple-related magazines in the hope that they will publish them in the letters to the editors column for the enlightenment of the rest of the Apple users.

Sincerely,
W.R. Eade
San Marcos, CA

Editor's summary of article by Paul Freiberger that appeared in the Oct. 19, 1981 issue of InfoWorld:

SUPERCALC MAKER REMOVES COPY PROTECTION

"Consumers have successfully pressured Sorcim to remove the copy-protection features from its recently released SuperCalc program. "Sorcim had sought to protect its new VisiCalc-like product by using a multitude of copy-protection schemes."

In that article, "Sorcim's president, Richard Frank, admitted that the firm had received pressure to remove the copy-protection tricks."

He is quoted as saying "Customers complained, so we removed it."

Paul noted that "The pressure from buyers is similar to that experienced by MicroPro International last year when it attempted to protect its own CP/M- based program, WordStar, from copiers." He added that "MicroPro's decision to recall a month's supply of software cost the firm $50,000."

marketalk

Dear Editor,

This past year has seen much in the small computer literature about software piracy. A great deal has been written about the dishonesty of the software consumer who copies rather than buys. I wish to turn attention to the dishonesty of the seller who overprices software.

In a free market, the seller has a perfect right to charge "what the market will bear" for his product and protect that product from copyright or patent infringement. I believe that the market is basically honest. I submit that the extent of piracy by potential buyers of software directly reflects the degree to which sellers overprice their product. To amplify, consider what is known about the market and how this is perceived by buyers and sellers.

To the buyer, the price paid for software is seen as the cost of the right to use the software plus the cost to produce a copy of it.

Consider first only the production costs of some typical package composed of a single disk and perhaps 50 pages of documentation. Given current costs for raw materials in quantities of a few hundred, it is profitable to sell such software for as little as $5 per package.

The buyer considers his cost to copy such a package at about $10 depending on his cost for blank disks, copy programs, and photo-copying service etc. If the package retails for $20 then the buyer sees the cost as reasonable and buys; if it sells for $200 then he sees the cost for the right to use the program as an exorbitant 20 times the cost to copy it.

At 20-to-1 he copies.

If apples cost $100 a bag, you could bet I'd turn farmer or thief. But buyers and sellers of apples operate in a free market and apples cost closer to a dollar a bag even with the considerable expertise required to breed apples and operate an orchard.

Meanwhile the average list price of software in Softalk's Top Thirty is $140, a whopping 14 times the cost to copy any single one of them. This should promote some fairly stiff competition. Instead, new entries are introduced at still higher prices and the publishers scream bloody murder that people copy their software.

Assume that a publisher believes his effort doubles the value of his product and he sells it for ten dollars. If a publisher charges more than ten dollars for a duplicate of his disk then he is ripping you off. If he charges more than ten dollars for a new version he is either ripping you off or implying a claim by the author that the new version represents an extraordinary effort. If an updated version costs $50 to each of say 500 users then the

continued on page 25

A Few Words From The Publisher

After printing only three issues and two updates (all of them late . . .), and after examining them for content (good ol' hindsight!), we find that there are a great many more neglected areas of Apple computering which we need to cover.

One correspondent asked me, "Where can you go after you've completed your review of bit-copy programs? Will your magazine lose interest?".

Well, since you asked, I will confide to you my very own candied-Apple visions . . .

Computer software is a very unique product with its own special characteristics that easily differentiate it from other forms of software such as film, music, etc. **But treating computer software as if it is like music or film recordings also denies its most important characteristic . . . computer software is an active and interactive product, not passive like a recording of some image or sound.** It is sad that it is still lumped in the category of passive software. By thinking of it in terms of un-alterability, computer software becomes just another form of static literature.

Interactive as a machine, computer software has the ability to be altered; made better, easily, by changing a few codes here and there. This means that, unlike passive software, computer programs can become instantly obsolete, and instantly updated. It is this phenomenon of obsolescence vs updating that differentiates computer software from passive software. Updating is the most critical feature of programs, a feature that we would all love except for one thing . . . copy-protection.

This most wonderful of all aspects of interactive software has turned into one of the biggest nightmares of all. Normally, updating would simply consist of entering the new and altered code, but because software is often "locked" and unlistable, the only easy way to improve obsolete software is to send for the new program. This process is annoying to users, but quite profitable to most of the publishers.

There should be a place to turn when you want to update programs yourself, a place to exchange updating ideas even on copy-protected software. What software have you altered, improved, fixed? Tell us about how and what you did.

Another topic HARDCORE should cover is "how creatively is commercial software being used?" After all, there's a lot of good Apple software out there. Is it being used, and how? I don't mean a review of the software. Reviews can be found in all the other magazines. What I do mean is: "User's Only" columns showing how people are using certain types of software either for their original purpose or for more creative purposes. For example, VisiCalc has its own users' column in some magazines. Certainly there are other well-used programs that deserve a column of their own.

I can think of three right off: the three leading bit-copy programs. How about others? DOS ToolKit is covered in NIBBLE. What about other graphic packages? And other business and utility software? You tell us what columns you want. Better still, write an article explaining how you use some specific software; a step-by-step specific problem-solving technique. If you've used a graphic package to write software, share that program with our readers.

It's up to you, the users.

We're a small magazine aimed at the serious users of Apple software. And we're trying to keep alive the idea that it is the users and not the producers, who should determine the value of computer information and its usefulness. We've tried to provide open education and entertainment without predetermining if that information should be kept secret. Let's start making HARDCORE an effective user forum. Keep those letters and articles coming in.

Charles R. Haight

SENSIBLE SOFTWARE, INC.

The Utility People

Making Life With the Apple II A Bit Easier

DISK RECOVERY "THE SCANNER"

48K+, Disk II
Apple II/Apple II+

$30.00

Are you plagued with disk I/O errors?

Does the integrity and safety of your data concern you?

This program is a must for all Disk II owners!!

Just as "Apple Pascal" provides a "BAD BLOCK SCAN", DISK RECOVERY will do a complete scan of your diskettes' recording surfaces. Damaged areas will be "marked" as used in the disk directory so that no attempts will be made to "WRITE" to a bad sector. The VTOC is completely rebuilt to accurately reflect BAD, USED, and FREE SECTORS, further insuring the integrity of the diskette. A complete report is generated advising the user of all corrections BEFORE the object disk is altered. A resulting "DISK MAP" is presented for your review. Flexibility of the program allows evaluation of NEWLY INITIALIZED DISKETTES as well as DISKETTES THAT ALREADY CONTAIN PROGRAMS AND/OR DATA! The SPEED of analysis is unsurpassed - only 18 seconds for a 16 sector diskette! DISK RECOVERY is supplied in the revolutionary QUICKLOADER format which allows easy reconfiguring to make it fully compatible with 13 sector and 16 sector diskettes as well as your SUPER-TEXT data diskettes! TREAT YOURSELF TO PEACE OF MIND KNOWING THAT YOUR PROGRAMS AND DATA ARE SECURE WITH...DISK RECOVERY.

BACK IT UP II+

48K+, Disk II
Apple II/Apple II+

$60.00

WE LISTENED TO THE APPLE USERS!
WE LISTENED TO THE REVIEWERS!

NEW! IMPROVED!

NOW EVEN BETTER THAN BEFORE!

BACK IT UP II has been improved and enhanced to provide the end user with the most powerful duplication system for the Apple II computer. Numerous PARAMETER CHANGES are allowed to directly handle the multitude of possible security formats as they are encountered. Direct study of these formats is possible with our own NIBBLE READER. Both 'full tracks' and 'half tracks' may be analyzed individually. Preview the source disk with the QUICK SCAN option to readily locate pertinent data. When duplication is begun AUTOMATIC HALF-TRACKING is readily enabled. A powerful SYNCHRONIZATION MODE has been added to the BACK IT UP II system. The MEND TRACK function manipulates data from designated tracks that are otherwise irretrievable with other programs. The end user is kept WELL INFORMED of the progress and any problems encountered once duplication has commenced. As with all SENSIBLE SOFTWARE products BACK IT UP II is extremely easy to use - USER FRIENDLINESS is more evident than ever before. The documentation has been expanded to a TUTORIAL ON DISKETTE FORMATTING as well as an easy to read USER GUIDE.

The availability of this program in no way implies that SENSIBLE SOFTWARE, INC. supports software piracy and is intended for the sole use of the computerist who wishes to 'back-up' his/her diskette library. Although itself uncopyable, SENSIBLE SOFTWARE, INC. does have a very liberal $5.00 disk update/replacement policy.

SENSIBLE SOFTWARE, INC.
6619 Perham Drive Dept H
West Bloomfield, Michigan 48033
313-399-8877

Visa and Mastercard Welcome

Please add $1.25 postage and handling per disk
For a complete catalog, send $1.00, refundable with your first purchase.

editorial

Unauthorized adaption or imitation = GRAND PIRACY

While software houses have been ranting and raving about the users who are making back-up copies of their products, a more dangerous and destructive form of piracy has become an accepted 'modus operandi' of some the software houses: blatant piracy of computer game ideas and their modes of expression, right down to the details of screen images! It is what I call:

GRAND PIRACY (as in 'grand larceny')

The game that first comes to mind (and there are quite a few of them being copied by competing houses) is Atari's Pac-man and the veritable hosts of pirated versions produced for the Apple market. (Time magazine, unfortunately confuses these two forms of piracy. See Otherwords.)

While the copying of a disk directly deprives an author or company a few bucks (which may add up to many bucks), a single person stands to gain only a few dollars, but receives a great deal of security. If the disk crashes, the user still has a back-up (or, as I urged in my earlier editorials, a FEW back-ups).

But when another software company pirates a game idea and its visual expression, and then sells it for its own gain, it is collecting revenue that should have gone to the original author(s) and publisher. I believe that this is the WORST form of copyright infringement.

As a published author of software, I fear this form of Piracy more than any other because, while user-duplication/ distribution 'for profit' of my games deny me only money, the theft of my game concept and mode of play denies me my claim to authorship, which is more valuable than mere money.

And, to top it all off, it appeared that this form of Grand Piracy was fast becoming an accepted form of theft supported by the software houses!

So it is with great relief that I read Atari's announcement that they would go after and prosecute authors and software houses that steal its game ideas and expressions. I congratulate Atari for taking a firm legal stand against this most destructive form of copyright infringement . And I support them in their struggle to chastize On-Line Systems for its obvious acts of Grand Piracy.

For a while it seemed as if Apple-users and programmers were excellent craftspeople but quite unimaginative artisans. They seemed to be copying game ideas rather than coming out with new and better ones. I hope that, as more people realize how easy it is to write programs, more of them will enter the commercial field and add to the growing number of Apple-based programs available to the users.

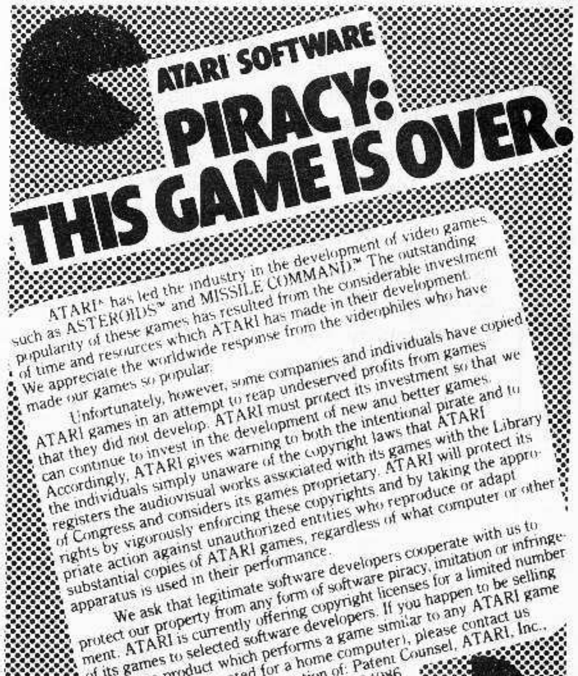
Software houses should adopt Atari's stand against Grand Piracy rather than continue their self-debasing accusations of petty piracy against the everyday users. Apple-ites are quite tired of copy-protected disks and will soon stand up, as CPM users do, and either stop buying protected programs or urge the soft-houses to stop protecting their disks. (*CPM programs are not protected. See Reader Input)

ATARI vs ON-LINE SYSTEMS

I have been notified that Atari won its suit against On-Line Systems. It is a victory for creative authors everywhere. My next question is whether Atari will now selfishly deny Apple-users of the Apple versions of Pac-man in order to promote sales of only its Atari game system. If so, then Atari is taking a very short-sighted view and should also be chastized. The authors of the copied games made an honest effort (and often a very exceptionally-crafted copy) of a popular game and they should be both compensated and credited (much like translators of books not originally written in English) and the game should be allowed to be licensed for distribution by On-Line and the other houses. If not, then I foresee that these games will be hot items on the 'black market.'

Since I've had my say on the topic of Grand Piracy, I openly solicit other views on this topic, especially from the software houses and authors involved.

Bev. R. Haight



ATARI SOFTWARE

PIRACY: THIS GAME IS OVER.

ATARI® has led the industry in the development of video games such as ASTEROIDS™ and MISSILE COMMAND.™ The outstanding popularity of these games has resulted from the considerable investment of time and resources which ATARI has made in their development. We appreciate the worldwide response from the videophiles who have made our games so popular.

Unfortunately, however, some companies and individuals have copied ATARI games in an attempt to reap undeserved profits from games that they did not develop. ATARI must protect its investment so that we can continue to invest in the development of new and better games. Accordingly, ATARI gives warning to both the intentional pirate and to the individuals simply unaware of the copyright laws that ATARI registers the audiovisual works associated with its games with the Library of Congress and considers its games proprietary. ATARI will protect its rights by vigorously enforcing these copyrights and by taking the appropriate action against unauthorized entities who reproduce or adapt substantial copies of ATARI games, regardless of what computer or other apparatus is used in their performance.

We ask that legitimate software developers cooperate with us to protect our property from any form of software piracy, imitation or infringement. ATARI is currently offering copyright licenses for a limited number of its games to selected software developers. If you happen to be selling a software product which performs a game similar to any ATARI game (such as a game created for a home computer), please contact us immediately. Write to the attention of: Patent Counsel, ATARI, Inc., 1265 Borregas Ave., Sunnyvale, Calif. 94086 guest opinion

in search of solutions to the problem of...

COPYING vs COPY COPY PROTECTION

There are many people who are genuinely concerned about "Piracy" and "copy-protection," people who are proposing solutions to this complex dilemma.

Bill Parkhurst, President of the Santa Rosa Computer Center, Inc., in California, took the initiative and wrote to us concerning one such solution. He also wrote to 3M about his proposal. In response to our inquiry, Bill Parkhurst explained some of the details of that solution. I encourage you to submit your own opinions and proposals for publication so that we can go on and make the software business both equitable and profitable for all concerned. Ultimately, the solution will be a unique one based on the fundamental differences between an "active" computer program and all the other forms of published and transmitted "passive" programs like literature, music, etc. But we must begin by seeking out the solutions used by those involved in the "passive" program industry.
—BRH.

Dear Editor:

Basically, my solution is parallel to what is being done in the recording arts.

You see, all royalties in the music industry are monitored and distributed by two publishing houses: BMI and ASCAP. If anyone writes a song which is going to be placed on magnetic media through a record company, then that author is automatically signed up as a member of either BMI or ASCAP. Part of their job (BMI and ASCAP) is to monitor air-play (the number of times a record is played on the radio by the various radio stations) and from that come up with the relative percentage per song in terms of the number of times it is played within a given period.

For instance, if Chubby Checker had a song that he wrote on the charts, say the Number One record, then Chubby Checker would receive the biggest chunk of the royalty pie.

Now to put that in perspective, if VisiCalc is the top selling program, then obviously VisiCalc deserves the top percentage of royalties for that given period that is being measured.

My solution would be to base the royalty on the total number of BLANK DISKETTES sold during the period.

I did receive a letter from 3M saying that they would be glad to supply the figures to the publishing house which is doing the monitoring for our industry. This publishing house would, of course, have to be established and incorporate in its membership a majority of the software producers in order to be effective.

Ultimately, it would BE effective because the revenues to the authors would be substantially greater than what they are getting now just from program sales. Also, as an additional benefit to the programmmers, they would no longer have to spend extra time with encryption techniques. This would ultimately benefit the consumer in the form of lower prices for software.

Copying a program is a sincere form of flattery. The average "pirate" is an educated, middle-class citizen with no criminal record. As a practical matter, as the music industry discovered, it doesn't pay to chase millions of consumers who are busy consuming. Record companies know people are recording songs off the air. So they base royalties on air-play. I have never heard a musician complain about people recording their songs from the radio.

So let them all copy to their heart's content and let the programmers get paid for it!

Sincerely,

Bill Parkhurst

PS. At Santa Rosa Computer Center, Inc., we do NOT encourage copying of programs. In fact, we spend a lot of time keeping people from doing it, which drives up the cost of doing business.

! rebuttal

Computer code must be kept hidden and secret!

Dear Editor Haight:

As a subscriber to Hardcore, I want to express my strong disagreement with some of the notions that you recorded in your editorial in the first issue of your magazine.

Yes, obviously, everyone has the right (and should be given the ability as well) to make back-up copies of disks. Your contention that everyone should be able to inspect the code, however, is counterproductive.

I think that the goal of any user is to see that high quality software is produced and available. How can we be sure that good materials are produced? By making sure that the creation of first-rate programs can be a money-making venture for people who have taken the time and effort to become good at writing software. If each program can be inspected, then those good ideas that people come up with can be used (pirated) by anyone else. Now spreading good ideas around may be good, but if someone's good creative ideas can be copied (by someone else who doesn't want to do the work of figuring them out) then the people with the good ideas will stop publishing. If they stop publishing creative materials, we all suffer. You have compared being able to see program code with being able to see the inside of an automobile engine. A better comparison would be with being able to see the recipes of a prize chef. The chef understandably wants to keep secret the recipes that took so much effort to create.

You also say that much software is over priced. The price of software, like the price of many other things, is determined in large part by a balance of supply and demand. If the price is high enough (too high?) then the best people will be attracted into the business of producing software. If the price is too low, the best people will make their money doing something else. Before you write your next editorial, I suggest that you give more thought to how we can ensure a continuing supply of creative software (new ideas) if peoples' ideas are taken as soon as their programs are published.

Very Truly Yours
Richard Cornelius
Assistant Professor
Wichita State U., KA

Where do first amendment rights of pirates stop?

Dear Editor;

I feel fairly confident that I know your stand on the issue of 'software protection' and 'software piracy.' After all, it is rather obvious from reading your first two issues. Let me tell you a little about myself.

My name is Allen L. Wyatt. I am a software author, editor, consultant, and one-time publisher. I feel that I have a right to make comments on this issue, for I have just as much at stake, if not more, than most of your readers.

Before you cry 'foul' or judge me as a member of the 'software bureaucracy holding poor customers captive,' let me state that I am not here trying to point fingers, I am not writing to try to 'call down' your magazine. In fact, I wish to applaud your first several issues. In between the flag-waving and wording designed to make people act on emotion alone, there were several very well-done and informative pieces that were well-worth reading.

I am writing, though, to raise a few questions. You see, many times in the real world, issues are not nessarily black and white. It is not a matter of the 'good guys' against the 'bad guys.' In fact, both sides of this issue have several good points. Conversely, both sides seem to be at fault in several areas.

Software publishers make quite an investment to bring a product to market. A good deal of money goes into development, royalties, marketing, packaging, support, raw materials and, yes, even 'protection.' The last area is there to try to 'protect' the other areas. Many times, the life of software is measured in weeks, or months at best, and a good deal of money is at stake for all concerned with a product. Besides this, publishers serve as a vehicle for spreading ideas. They serve to transform local markets into national or international markets.

On the other hand, users have a 'right' to software that works (bug-free), that is modifiable (if necessary), and a back-up to protect their investment in the software. It is frustrating at best to have a piece of software crash and not have a back-up.

Besides these valid points there are a few other points that must be considered. It would be extremely convenient and idealistic to say 'all software should be unprotected.' That would definitely solve the problem for users, but not necessarily for publishers and authors.

There are computer users who are genuine 'pirates,' even though I am sure the number is small. No matter how small the number, they can hurt and, in the case of small publishing houses, severely damage a business. As an example, look at Pirates Cove. They are, by their own admission, pirates and they claim protection by first amendment rights.

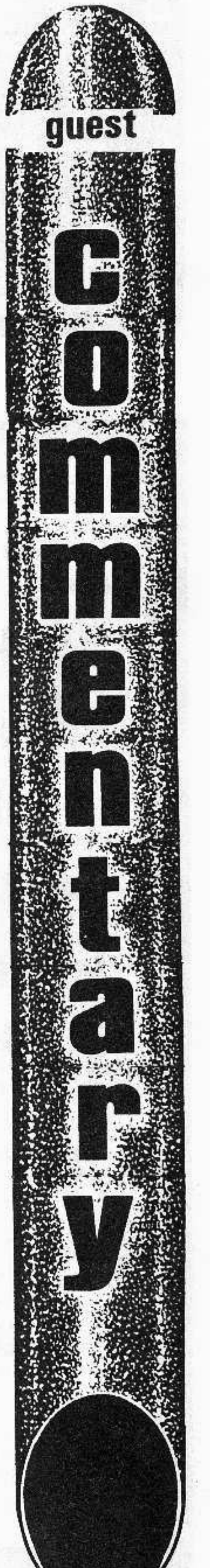
I, in fact, would like to see publishers not have to copy-protect software products. It would certainly be less expensive for everyone concerned.

Where do the first amendment rights stop, however? According to the courts, they stop when they interfere with the first amendment rights of another. When does that happen? Does it happen

continued on page 44

Software Insurance Policies

"A Whole New Ancillary Business." Tod Wicks



Dear Editor,

Much has been written in recent months in computer hobbyist magazines (both general and Apple-specific, as well as Apple club newsletters) about the pros and cons, mostly the latter, of bit copiers and, noncommitantly, software piracy. Some of these comments have taken the form of emotion-charged editorials or comments without byline presenting the prevailing view of the magazine management who are sometimes funded by a software house (ie. Kilobaud Instant Software). Other written pieces have been reader responses to the above. Now it's my turn.

I have owned an Apple II for over two years and use it for fun and games, hobby, and I use it with my home-business. I have invested some pretty good money in professionally-prepared programs, including Visicalc, DB Master, and a Pascal Text Formatter. These three, along with Synergistic Software's Mail Label Program (used to generate 2500 labels per month) are the most often used of my heavy artillery.

To not have backups would be foolish. I have an authorized backup for DB Master, sent when I returned the registration card. The text formatter came with two disks, and the Synergistic program was copyable.

I welcomed the first bit copier and immediately copied Visicalc (rather than send Personal Software $30 for a backup). I have heard of problems regarding dust and heat vis-a-vis the not-so-hardy disketts, so I self-insured my business by having THREE backups for my expensive software.

The original is kept in a protected chest, two copies are handy on my computer table, while the third copy is kept in a safe-deposit box. Overkill? Perhaps, but I'm assured that I won't have wasted my hard-earned cash if something happens to one of my program disks. And for most of us, the operating phrase is 'hard earned cash,' because money doesn't grow on trees and when a good program is produced, it is important to insure it.

This insurance of software could open a whole new ancillary business. I recently bought two new packages which totalled over $400. After leaving the computer store, I ran some other errands at another location. When I returned to my car, I noticed the software was missing. Someone had jimmied the lock of my hatchback and ripped me off before I could even make copies!!

I notified the police and drove to my insurance agent. The forms were processed and I went home to figuratively wait for my claim (less the $100 deductible) to arrive.

Several weeks went by and I was just told by my agent that this claim was not going to be honored by the home office due to an exclusion against magnetic information stored on disks, etc, except for the value of the blank medium. Ha, Ha.

I checked with a couple of other agents, who had to find out from their respective parent companies. To a man, they told me that software programs weren't insurable; I couldn't even get a rider to the policy to take care of the exclusion. I also found only sympathy from the distributor as they couldn't help me since I hadn't mailed them my registration card.

Now I'm out the $400.

Here's a neat solution which hopefully would satisfy dealers, customers, users, authors, distributors, in fact, everyone. This should take care of software problems ranging from loss due to theft or fire, to carelessness with the program disk (overwritten programs).

When the customer buys the software, he or she has an option to buy into an insurance program, either by the piece or a blanket coverage of software. If this is the customer's choice, then his name, the program name and serial number, date sold, and any other necessary information, are entered in a data bank.

If the user develops a problem, he or she returns to any computer store hooked into the data bank with his customer ID# and explains the problem. If the program is important to his/her needs, a 'loaner' could be let out, with a reasonable deposit kept at the store. The information relating to the damage or loss would then be entered in the data bank and a replacement copy mailed to the customer. Some safeguards would necessarily be built into this system to protect everyone involved from flagrant misuse. This could include a magnetically encoded ID# on a membership card, which could be read by a reader and verified by the data bank.

Another security device is simply the rapport built up between shopkeeper and customer. Most of us settle on one computer store as our primary vendor after a while, and we usually are welcome to browse and get a sort of 'favored customer' treatment, ranging from hot programming tips to a free cup of coffee to a willingness to take the time to answer questions. They know with whom they are dealing and could assist in the speedy recovery of lost or damaged programs. I know; I've already been helped immensely by the people at my computer store.

Thus the software authors and vendors, the editors who shout, 'Stop thief!' and all others involved with the production of software should start with positive foreward thinking and not stamp their collective feet and throw tantrums about what's already happened. These people are clever enough to bring out some damned good programs; they can put their creative thinking to work in helping the customers and themselves by learning from the past and applying those lessons to the future. I want action and insurance, not rhetoric and recriminations.

Sincerely,
Tod A. Wicks
Palo Alto, CA.

A.C. "Mike" Markkula, President of Apple Computers Inc.

"It is true. We are against software protection.

We don't think in the long term it is the best thing for the industry or the customer."

HARDCORE: What is your definition of the situation you call 'software protection'?

MARKKULA: 'Software protection' is the universe of schemes used to prevent unauthorized copies of a program from being made.

HARDCORE: And that would include hardware and software protection. . . .

MARKKULA: ...Software, hardware, fooling around with operating systems, licensing agreements, and so forth . . . Whatever is used to prevent unauthorized copying . . . The key word being: unauthorized.

HARDCORE: Do you agree with the definition of 'unauthorized' that includes back-up copies?

MARKKULA: No I don't. Some protection schemes are not perfect, and in their effort to prevent unauthorized copies they prevent authorized copies as well.

HARDCORE: At the conference* did you get the response you expected when you made your statements regarding copy-protection?

MARKKULA: Yes.

HARDCORE: Creative Computing** said that 'The software pirates in the audience applauded.' Do you feel that is an accurate assessment of the audience's response to your statements?

MARKKULA: No. There were many legitimate software authors who participated in that applause and came up and said so after the meeting. They heard the whole statement. A lot of it was left out when the quotes came out in the papers. And the idea of forming an industry association that would tackle this problem and do what was good for the customers, the authors and the publishers simultaneously is what the authors were after.

HARDCORE: Was that the first time you made such an announcement regarding software protection?

MARKKULA: Yes.

HARDCORE: Have you made any similar announcements after that?

MARKKULA: Yes. I say the same thing every chance I get.

HARDCORE: What has been the response to those statements since then?

MARKKULA: I've gotten a lot of letters, and they've been running about half-and-half in favor of or against. There are many suggestions that are constructive suggestions: ideas on how we might go about solving this problem as an industry, and we're collecting those and writing a 'White Paper,' if you will, to try to get the industry off of dead-center and begin to make some progress toward solving the problem.

HARDCORE: Did you receive any innuendos from software people saying that they'll cease software development for the Apple and go on to other computers if they aren't able to copy-protect their material?

MARKKULA: Yes, and those generally came from having read in the news media what was quoted which didn't include the rest of the discussion there.

HARDCORE: In other words, the quotes were generally out-of-context . . .

MARKKULA: Yes, because I said that we would SUPPORT software protection until such time that we could offer an alternative solution which would be considered appropriate and reasonable by the various elements involved in this business. One of the comments I made was that 'it was like Listerine . . . we don't like it but we use it.' And that's really our only choice at this point. There is no alternative which would allow the authors of the community to continue to have a profitable business for themselves if we don't support protection in some way.

HARDCORE: What do you think of the authors and the software companies who are now changing their protection policies and not protecting their material and, in fact, encouraging that back-up copies be made.

MARKKULA: I think that they're going to win.

*** At the Boston Computer Society's forum on "The Future of Personal Computers" held at the Northeast Computer show (Boston, October '81), Mike Markkula, president of Apple Computers, Inc. surprised many of the audience by taking a stand on the "Copy-protection" controversy in favor of its elimination.**

"In perhaps the most surprising and controversial statement of the evening," reported Creative Computing, "Markkula urged the elimination of software protection. The software pirates in the audience applauded." CC went on to add that "he promised that Apple 'will work as diligently as we can to eliminate the situation that we call software protection'. " Byte also reported a similar quote.

Bev. Haight interviewed Mike Markkula by phone in May to find out how the world of personal computers reacted to such a stand. The following article is an edited transcript of that conversation.

I think the whole thing is a matter of economics. The end result (of copy-protection measures) now is that developing legal mechanisms to really take those people to task who make unauthorized copies (has evolved) to the extent that it is deleterious to the authoring and distribution community. If we can increase the volume of programs that are sold sufficiently that the 'economies of scale' reduce the manufacturing costs and the distribution costs to the point where the legitimate customer would prefer to buy the original item which has a warranty, a real 'honest to goodness' manual with it...I think that he will consider it in his best interest to do just that. And, I think, he will feel that he is getting a fair value for what he spends.

HARDCORE: That depends a great deal on increasing the number of Apple sales, too, doesn't it?

MARKKULA: Well, it's not just Apples, it's ALL personal computers. Of course, we would love it to be all apples, but that's sort of the short-sighted view.

HARDCORE: You mentioned the 'White Paper' you are working on. What steps have you taken in the direction of uncovering a solution other than software protection.

MARKKULA: Well, the first thing we did was a very extensive technical analysis in conjunction with some people at Stanford and the conclusion of that study is that, unfortunately, we can find no technical solution that works, that can't be broken. There is theoretical proof of that. And I think that we should make that fact known. I think people should understand that.

HARDCORE: In other words, copy-protection simply won't work if the information about how it's protected gets out.

MARKKULA: Even if it doesn't, someone will find a way to make copies. The question is, 'how long does it take, and how much effort do they have to put in to do it.' It's not a question of: 'is it unbreakable,' because technically, it can be proved that there is no scheme which cannot be broken.

The next thing we did was to put some people to work making a study of other publishing types of industries such as records and tapes, books, and so forth, to find out whether or not they have had similar problems and what they have done about it. And it turns out that they DO have the same problems, and there ARE revenues lost due to unauthorized copying. And in some cases, the revenues are significant.

HARDCORE: As in the video industry?

MARKKULA: Video, records . . . So we tried to collect the data, the numbers, so that we can say, 'Now look. Here's how big the problem is.' Nobody has any data to tell you what percentage of the software gets copied or in the long run, what percentage of revenue are we going to lose because of this.

So we tried to get some data together and draw some conclusions based on other industries as to how big the problem might be in our industry. That, we hope, will lead to some reasonable and thought-out solutions which are based on reality rather than a lot of emotionalism.

And the thing we are doing in that Paper is to propose some mechanism that might be used on an industry-wide basis, not just an Apple effort, to pursue some of these solutions and decide on one and go ahead and implement it. And those possible solutions have a fairly wide range.

HARDCORE: Have you proposed this to either those who support, or oppose you?

MARKKULA: Not yet. We want to make sure that we get all of this stuff done. And then we're going to send a copy to each of the people who have written in. Also, we hope to get the paper published someplace so that it can gain wider distribution.

HARDCORE: Has that Paper been completed?

MARKKULA: No. We're still working on it. Somebody has to do some work, you know. You can't just sit back in your office and conjure up these things. You've got to put some effort into it. There are four people working on it now, not all full time. But four different individuals are contributing.

HARDCORE: Have there been any software companies or magazines that say that they support you.

MARKKULA: No. Most people are still in the questioning phase. And part of the reality of the problem is that no one knows how much of a problem it really is. No one can make a value judgement today, or a business judgment, that's based on anything other than rumor, hearsay and supposition. So, if you were running a software company and you were faced with the decision: 'should I protect my software or not?' You don't have any way to know what the ramifications of that decision would be. You don't know how much you are going to lose. You don't know how much the market would increase if you could offer the product at a lower cost (because you didn't have to spend money developing a protection scheme.)

There isn't any data for people to make that decision on. So those that make the decision would say that the safest thing to do is to protect it.

One of the things that Apple is going to do in order to help generate some of that data is offer a major program in an unprotected form.

HARDCORE: What program is that?

MARKKULA: I can't reveal that yet. But we will be announcing a new program which is a very significant one, and we're not going to protect it!

HARDCORE: Is this just an experiment, a short-lived venture that can be quickly changed?

MARKKULA: It can certainly be changed. But, we really are going to carefully measure the sales of that product and try to assess how many unauthorized copies are made in order to generate data for the industry to use in helping to make decisions on whether or not we should have some form of software protection.

HARDCORE: I thought that someone had earlier attempted something like that . . . where they took a program and released it unprotected in a specific area in order to see how fast pirated copies would reach the opposite coast . . .

continued on page 67

barry d. bayer

consumer's

FROM A BUYER'S POINT-OF-VIEW

GOOD GUYS, BAD GUYS, AND

The purpose of this column is to give some hints to you, the Apple II user, as to how you might be able to break even in a jungle of software and hardware purchases.

I'll try to give you my ideas as to the best way of dealing with manufacturers or sellers of unknown quality (unless you have to, DON'T), and some general hints as to what to look for and what to look out for. I'll start this thing out on the assumption that consumer dissatisfaction does exist, and that at least some of the time such satisfaction is justified.

Let me know if you see things going on (or happening to you) which you feel are wrong. I'm probably not going to solve your problem, but I might be able to help others avoid the same mistakes. (Of course the threat of publicity might also have a salutory effect on a potential settlement, but then again, I suspect there are manufacturers out there who could care less.)

If I don't receive correspondance relating current problems, I will assume that everyone is happy, that nobody feels wronged, and that this column is completely unnecessary.

INVISICALC

Let's start with some personal comments on good guys and bad guys I have dealt with. (And sometimes they are the same outfits).

High on my list of split personalities is Personal Software, Inc., the publisher of VisiCalc.

VisiCalc is a marvelous program.

(Hooray!)

Why did they charge me the outrageous price of $65 to update from 3.2 to 3.3, when the reason that I couldn't 'Muffin' the program, myself, is that it is 'locked'.

(Boo!)

But the 193 version has significant enhancements, such as Boolean operators, and DIF.

(Hooray!)

But the 193 was officially declared defective as of August 1981, only a couple of months after I got my upgrade.

(Boo!)

But PSI sent me the Version 202, free, to replace the defective 193.

(Hooray!)

But they charge someone who bought the 193 for $200, originally, another $15 to upgrade to 202.

(Boo!)

The company claims that the 'upgrade' charge only covers their costs, and is not an attempt to wring another dollar out of the consumer.

(Hooray!)

But PSI strongly objects to the thought that a registered 202 owner might help both the company and another registered 193 owner (who might have inadvertantly purchased the program well after the recall date---PSI did not draw a lot of attention to the recall) by simply making a LOCKSMITHed copy of 202 and giving it to the 193 owner, thus saving PSI a lot of time and trouble. (A violation of copyright, they say, and probably correctly so. But why would they object if they weren't making money on the $15?)

(Double Boo for the Double Talk!)

If I had the 193 version, and particularly if I had purchased it after August 25, I would scream bloody murder until my dealer, the manufacturer, or somebody, replaced the defective program free of charge. I wouldn't sue by myself, however. The $15 replacement charge clearly isn't worth it.

(As I reread the foregoing paragraphs, I guess I sound a bit ungrateful, as PSI did send me my update free. Yes, Personal Software, I am grateful you sent me mine. But why not do the right thing all the way around?)

MX-80 MIX-UP

With that tirade out of the way, let me go on . . . and present a boquet to JADE Computer Products, who started out with a normal misunderstanding that was at least a bit my fault, and ended up giving me full satisfaction, with only 2 '800 number' phone calls.

Last summer I decided it was time to buy a moderate priced dot-matrix printer with friction-feed and hi-res graphics, and that the Epson MX-80 F/T seemed the way to go.

(Bayer consumer protection rule #1: 'Never buy ANY computer product unless you've seen it advertised at least SIX months in a row.' This probably insures that anything I buy will be obsolete before I get it, but it cuts down on the possibility that I will pay a premium price to become a Beta test site.)

The JADE price, which included graphics, was just right, and I ordered one by telephone, charging a credit card. I was dumb enough not to get an order number, nor the salesman's name (but thanks to JADE's attitude, this wasn't fatal to my cause.)

Many weeks passed, with JADE responding to my calls with the message that they were simply not getting what had been promised to them. I decided to hang on, however, and one bright October day UPS delivered the package. A quick look at the documentation made me suspect that the hi-res graphics PROMs were not part of the package, and this suspicion was confirmed by telephone.

Although I had NO written documentation of my order which indicated that I had intended to purchase the graphics as part of the agreed price, Scott Anderson of JADE agreed that the advertising on which I had relied did indicate that the PROMs were part of the deal. (I get the impression that either the importer or JADE's distributor may have misadvised JADE when the original advertising was placed.) He

LEGAL FORUM

SPLIT / PERSONALITIES

agreed to either (a) send me the PROMs, (b) take the printer back on other merchandise, or (c) take the printer back and cancel my charge. Here I was quite ready to be righteously indignant, but Anderson and JADE wouldn't even let me rant and rave. I was impressed! I'll certainly consider them on my next purchase.

(I finally installed the PROMs a couple of days ago, and was ready to demand JADE replace them under warranty, because they were defective. I saved myself a considerable amount of embarassment by checking the entire installation procedure outlined in the manual, and changing the switch position I had previously neglected to do.

Bayer Consumer Protection Rule #2: 'Always read and follow the documentation before you start serious complaining.')

NONWARRANTIES

As long as we're talking about the Epson Graftrax 8Ø PROMs, let's finish up this month with something that you will read about again and again in this column: LAWYER OVERKILL IN WARRANTY DISCLAIMERS. Let me start out by saying that the PROMs work very well; I have had no problems with them. But the 'boilerplate' in the manual is unbelievable.

First of all, although I thought I PURCHASED the Graftrax 8Ø package as part of the printer, Epson America, Inc., claims they are only LICENSING me to use the program, 'in connection with a standard, unaltered MX-8Ø or MX-8Ø F/T computer printer manufacturer by Epson.'

(Does this mean that, if I repaint the case a different color, I can't use the graphics?)

This 'license agreement' gets better. **'Customer understands and agrees that he/she will not copy, reprint, duplicate or modify all or any part of Graftrax 8Ø . . .'** and that Customer will not try to 'reverse engineer' the program, that is, to disassemble to object code to see what makes it tick.

On the inside of the manual, we have the statement that Epson disclaims all implied warranties including, oddly enough, warranty for fitness for any particular purpose (presumably, even the only particular purpose for which the Customer can use it, if the 'license agreement' is to be believed), and that 'THE BUYER (AND NOT EPSON AMERICA INC, ITS DISTRIBUTOR, OR ITS RETAILER) ASSUMES THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. . .

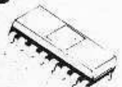
Clever. The Buyer (notice 'buyer' and not 'licensee' . . . hmmmm) has to pay for patching the program if it doesn't work, but he is prohibited by the license agreement from modifying or disassembling the program.

I suppose the answer is that Graftrax 8Ø seems to work without any problems that I see, so who cares what the documentation says, anyhow.

But why is such overkill necessary?

And what would EPSON have done if there really were problems in the ROMmed graphics program?

**ROMPLUS*
NEEDS
ROMS
FOR
APPLES**



RESIDENT FIRMWARE UTILITY PROGRAMS WILL EXPAND THE CAPABILITIES AND INCREASE THE USEFULNESS OF YOUR APPLE II OR APPLE II PLUS! *

***APPLESOFT EDITROM**
Global search, change, or remove any string, variable, literal, constant, or basic command word that appears in your Applesoft Programs. **EDITROM** uses no ram space that will interfere with your program. It does not reset any system pointers to protect itself and will operate with any size system - 16K, 32K, or 48K. After **EDITROM** has been initialized, the ampersand (&) command can then be used to call the **EDITROM** back for repeated use without readdressing the ROM + BOARD. Completely compatible with Konzen's Program Line Editor. If **PLE** is up, EDITROM will keep PLE up and allow joint operation.
Will operate with any version of **DOS** and requires FP in Rom. **35.95**

COMMANDROM
COMMANDROM Is like having a resident **'FID'** but with more operating features and conveniences. **COMMANDROM** will read a disks (13 or 16 Sector) File Directory Listing and display the following: A Command Menu, current drive number, number of sectors used and left, number of pages set up to hold all file names, the first page of file names, and an indentification letter next to each file name on display. Pressing any one of the command keys will load or run any file (A 'B' file load will display start and length addresses.), lock or unlock a file or all files, delete a file - with verify before deletion, change from one drive to the other, read a new disk, display a Track/Sector Map, change page numbers to view all file listings without recataloging, or exit to current language or monitor. No system pointers are reset and no RAM is reserved for **COMMANDROM.** Requires 48K, 3.2 or 3.3 **DOS** and **INT** or **FP** in **ROM** **35.95**

BASICSROM
Will boot a 13 sector disk on a system configured for 16 sector operation. The **BASICSROM** can be addressed on coldstart (without **Auto Start ROM** or warmstart (with **Auto Start ROM)** at any time **35.95**

DISK COPY/SPACE ROM
Duplicates a disk, from either **SINGLE** or **DUAL** drive, single or dual controller, 13 or 16 sector and with or without **VTOC.** Options include - Gross copy, active sectors only copy, **DOS** overwrite, auto boot of copy disk, free space on disk in sectors and kilobytes, and Init and volume number are selectable. Requires a minimum of 32K **35.95**

***APPLESOFT RENUMBER/ MERGEROM**
Made famous by Apple Computer Inc., this powerful utility will not disturb any part of a program in memory, when it is activated. Requires 48K, with or without Disk II. **35.95**

'YOUR' PLE ROM
Now you can put your Program Line Editor in **ROM.** Write for details. . **$45.95**
*REGISTERED TRADEMARKS

BOX 599
WEST MILFORD, N.J. 07480
SOFT CTRL SYSTEMS
201-728-8750

FOCUS

APPLE AVOCATION ALLIANCE

The free enterprise system has encouraged the birth and proliferation of some very greedy as well as some very consumer-conscious software outlets and producers. Most of us are aware of the greedy ones. They're usually the ones with the big bucks that permit clever and prolific advertising. It's the others that should receive the publicity and our business. One of those is known as the Apple Avocation Alliance, Inc. (which I call: the Triple A).

AAA is not only a Library of Public Domain Programs, but it is also a mail-order retail outlet for computerists, and a publisher of the 3A Press newsletter.

Most computerists have a very meager software library that probably cost them an arm and a leg (which perpetuates those high-profit but low consumer-considerate businesses). The AAA operates on a low profit margin and therefore depends upon a larger quantity of sales, otherwise their growth will be slow and that will be a loss to all Apple afficianados. Here's why.

The AAA has over 121 volumes (diskettes) of Public Domain Software. Each disk has ten to 40 programs, and each is categorized. The categories include: Art and Graphics, Astronomy, Aviation, Business and Finance, Chemistry and Biology, Demos, Education and school, Electronics and Radio, Food, Games and Adventures, Math and Statistics, Music and Sound, Sex, Religion, Utility, etc. They even have 16 Eamon Adventures, though each is on a separate disk (see Adventure Tips, Issue #2, by Mike Flynn).

The hard-copy catalog of the Library runs 15 pages, plus there's a very informative 3A Press newletter added on.

So how do you go about buying these programs? You don't! These are public domain programs. All you pay for is the fee to copy them! And that fee is either $1 or $2, depending on how it's done. If you order their pre-copied Media diskettes, you pay only $1 plus the cost of the diskette, which varies from $2.25 each (for less than ten) and $1.90 each (for ten to 19) to only $1.70 each for over 200. If you want the volumes copied upon another brand, the copy fee is $2 plus the cost of that brand of diskette. I strongly suggest you order their catalog ($2) even if you're only interested in the low-priced products (such as diskettes) that they offer. These products include diskette supplies such as sleeves, file case, labels, hub rings, binder pages, and printer paper, ribbons, mailing labels, cables, nibbling punch, etc.

To order, however, you must subscribe, but that is only $3. AAA also sells commercial software and hardware at their low prices. How can they sell at such low prices? Most "dealers" get supplies at a discount of 40 to 60 percent, and then re-sell at the list price. By selling products at almost the original discount price, most software prices begin to be more reasonable. Unfortunately, they've received some flak for selling at such reasonable prices, especially from Verbatim.

(Ron Maleika, the AAA Librarian, said, in the 3A Press, "I've returned bad disks to Verbatim for replacement, but they have NOT honored their guarantee to replace them! Can you believe it? They didn't even have the courtesy to return the bad disks. I've phoned them seeking to buy their disks directly and received a lecture that I'm selling their disks too cheaply and they demanded to know who was selling them to me at such low prices. Why would they want to know except in some way to punish their distributors? They promised twice to have their distributors contact me, but no one has called or written. Isn't it a shame that some companies get so big they can afford to IGNORE us little guys, and with impunity?")

How did AAA get so many programs?

According to Ron, it began in late 1980. "My friend, Jim Hassler, a HAM radio fan, had started The Apple Net. It was and is a public forum over radio to talk about Apple! As a result of the radio sessions, Jim began to acquire Apple programs from other HAMs. Soon he found himself acting as a librarian for several scores of programs. As it developed, he became swamped with many duties, and since I was at hand and had an Apple, I inherited the library.

"The HAMs continued to contribute programs and I expanded by mailing a small listing to Apple Clubs around the world. I also gleaned magazines, etc., and sent a list of programs to any address that seemed interested in the Apple."

"As programs were exchanged, the library grew and keeps growing at rather an astounding rate. Word of mouth is doing the rest. I've considered advertising in one or two of the good Apple-oriented magazines, but I'm afraid the exposure would really swamp me."

This is no longer quite true because Ron has some more help. Bill Williams is in charge of commercial software. And Helen Williams is the corporation secretary. Hardware and accessories are handled by Ron Deutsch. The Public Domain Library is still run by Ron Maleika. As the AAA grows, so does its services and products.

You can help increase Ron's workload by trading public domain software. To make it easier on Ron, though, first send a hard copy list by volume number of the program titles you have to exchange. Include their sector lengths. He can then indicate which programs or volumes he would like to exchange and negotiate exchange rates (how many volumes you will get in exchange for your programs). After all, he may already have some of the programs you wish to exchange.

What happens when someone exchanges a copy-righted program?

"I offer a token reward to anyone who identifies a copyrighted program on one of my disks as it is impossible for me to be familiar with every one of the thousands of programs available for Apple."

Ron also carefully lists and studies each program searching for a copyright statement. Sometimes he gets "fixed" programs or ones where the copyright notice was removed. This just makes Ron's job harder. He does have copyrighted programs in the "public domain library" but these were contributed or exchanged by their authors.

Who uses the public domain library?

"Not only are Apple clubs big users, but also school districts and universities. As adjuncts to the Apple Tutorial, public domain programs are a way to learn at a small cost. Programming techniques from hundreds of programmers are available to study and emulate. Not to be overlooked, even many

continued on page 49

SOME APPLE-ONLY MAGAZINES...

It is my firm belief that the greatest user support is given, not by the computer dealers but, by the many computer clubs. Second to the clubs are the numerous computer magazines. And if your local Apple dealer doesn't support clubs or doesn't carry computer magazines then that computer store does not support the user and is merely a sales outlet. Since Apple Computer, Inc. now depends solely upon local retail outlets as it's only direct interface with the users, any store that doesn't support clubs or magazines is doing the users (consumers) and Apple a disservice. You should complain to Apple whenever this happens.

If I seem to be trying to get you to read other magazines, it's because magazines are one of the best sources of user information. Everyone should subscribe to at least one of the all-Apple mags now available.

Idealistically, I'd like all Apple owners to subscribe to HARDCORE.

Realistically, however, HARDCORE still has a ways to go before it can deliver the volume, variety and quality of Apple-info presently delivered by an assortment of Apple-mags.

Therefore I suggest that, if you do not already subscribe to the magazines I am going to quickly review, you should at least get a copy of each and examine them. They are:

1. NIBBLE
The Reference For Apple Computering
P.O. Box 325
Lincoln, MA 01773

2. International Apple Core presents
APPLE ORCHARD
P.O. Box 1493
Beaverton, OR 97075

3. CALL A.P.P.L.E.
304 Main Ave.,
Suite 300 Renton, WA 98055

4. SOFTALK
11021 Magnolia Boulevard
North Hollywood, CA 91601

5. PEELINGS II
The Magazine of Apple Software Evaluation
P.O. Box 188
Las Cruces, NM 88004

NIBBLE

Personally, NIBBLE is my favorite mag because it really has variety. A glance at their table of contents will prove my point. It has many features that HARDCORE is striving to acquire:

1. Reader involvement in the improvement of listed programs.
2. Numerous short and informative articles on utilities, aids and tricks.
3. Several feature program listings, many for home/business.
4. In-depth explanations accompanying every listing.

Call A.P.P.L.E. & Apple Orchard

While NIBBLE is a fully professional magazine (by Micro-Sparc Inc.), some of the other dominant Apple mags are actually professionally crafted 'newsletters.' That word is not meant to be derogatory. It means that those magazines are actually functions or services provided by an often non-profit computer club. The two most recognized of these publications are Call A.P.P.L.E and APPLE ORCHARD.

Without going into the purposes of the clubs, etc, let me simply say that they often contain many useful articles on programming, and some really excellent material on assembly language utilities. Irregardless of their editorial policy (which, as a member, you can alter by voting), they are fine publications.

With the entrance of Peter Weiglin as the new editor (replacing Val Golding, present editor of Call A.P.P.L.E.), APPLE ORCHARD can now compete with Call A.P.P.L.E. As a result, I expect to first see some remarkable improvements due to its new, competitive status.

Two other magazines have taken the big leap into competing with the big pros. They are PEELINGS II and SOFTALK.

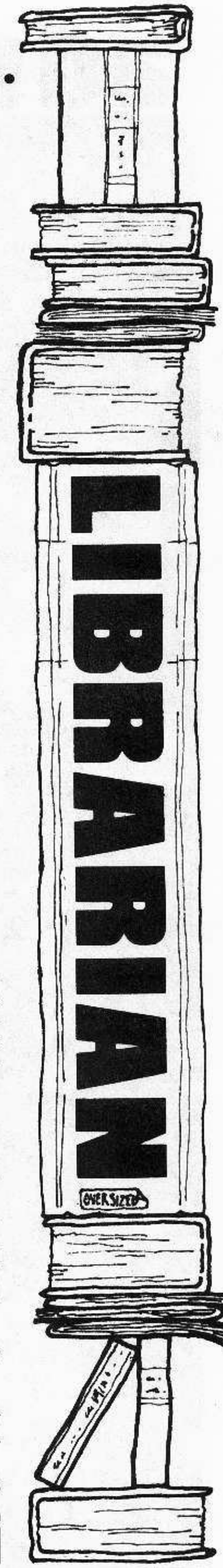
SOFTALK

Originally only an industry "fluff" magazine telling who is doing what and who got promoted to where, SOFTALK has expanded in size and coverage and is now running major informative articles on programming. It is also now charging for subscriptions.

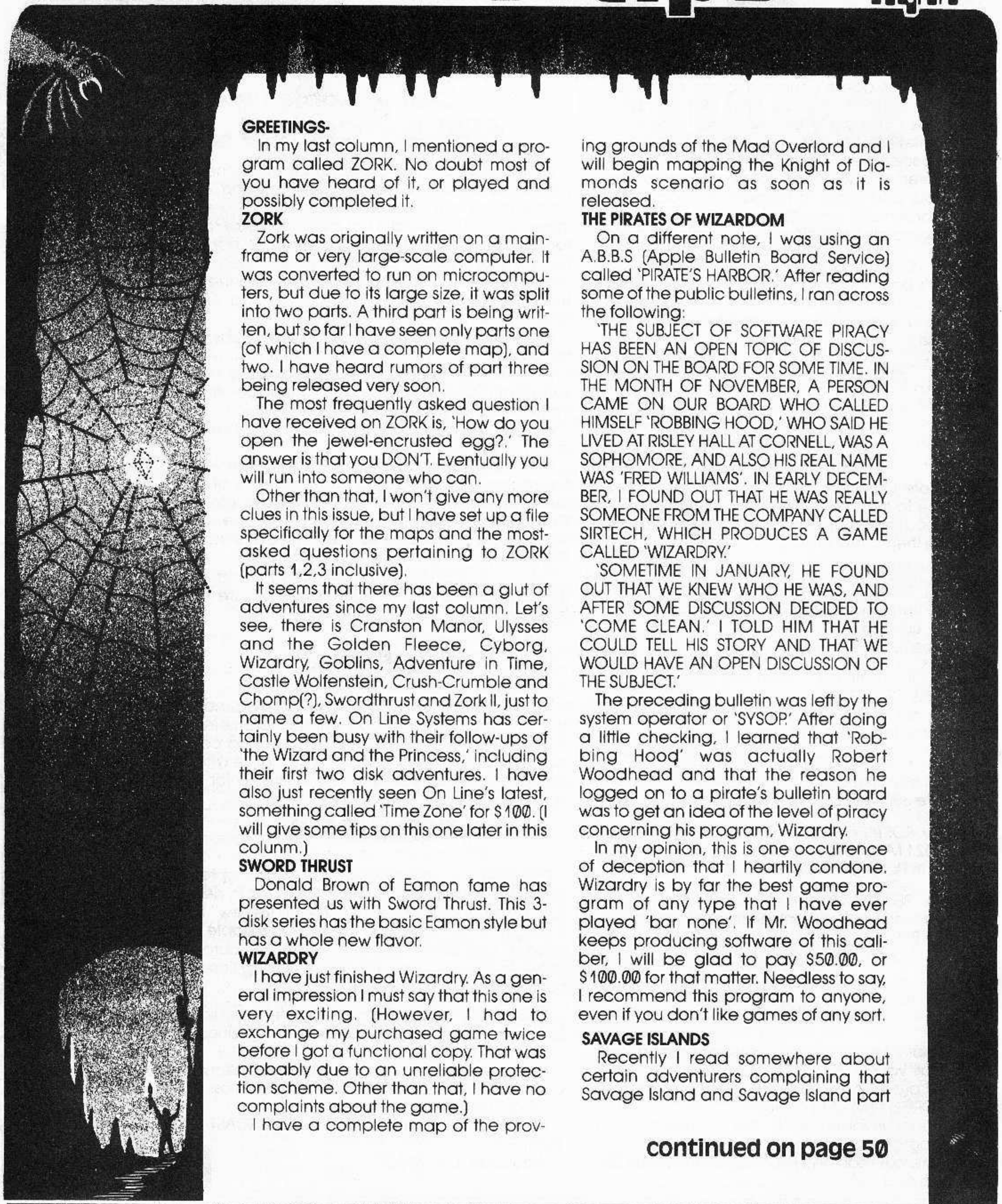
PEELINGS II

PEELINGS II is now typeset and is looking really good. The reviews within are incisive and in depth. As a professional "House of Software Review" they must get nearly all the commercially available software, so I consider PEELINGS II to be THE source of software reviews. They are also reviewing firm and hard ware.

Now to round off this Librarian's notes, I would like to mention another multi-computer magazine:
DATACAST
Software Systems and Telecommunications for Users, published Jim C. Warren Jr. and Wireless Digital, Inc.
For more information, write to DATACAST, 333 Swett Rd.,
Woodside, CA. 94062.



adventure tips by mike flynn



GREETINGS-

In my last column, I mentioned a program called ZORK. No doubt most of you have heard of it, or played and possibly completed it.

ZORK

Zork was originally written on a mainframe or very large-scale computer. It was converted to run on microcomputers, but due to its large size, it was split into two parts. A third part is being written, but so far I have seen only parts one (of which I have a complete map), and two. I have heard rumors of part three being released very soon.

The most frequently asked question I have received on ZORK is, 'How do you open the jewel-encrusted egg?.' The answer is that you DON'T. Eventually you will run into someone who can.

Other than that, I won't give any more clues in this issue, but I have set up a file specifically for the maps and the most-asked questions pertaining to ZORK (parts 1,2,3 inclusive).

It seems that there has been a glut of adventures since my last column. Let's see, there is Cranston Manor, Ulysses and the Golden Fleece, Cyborg, Wizardry, Goblins, Adventure in Time, Castle Wolfenstein, Crush-Crumble and Chomp(?), Swordthrust and Zork II, just to name a few. On Line Systems has certainly been busy with their follow-ups of 'the Wizard and the Princess,' including their first two disk adventures. I have also just recently seen On Line's latest, something called 'Time Zone' for $100. (I will give some tips on this one later in this colunm.)

SWORD THRUST

Donald Brown of Eamon fame has presented us with Sword Thrust. This 3-disk series has the basic Eamon style but has a whole new flavor.

WIZARDRY

I have just finished Wizardry. As a general impression I must say that this one is very exciting. (However, I had to exchange my purchased game twice before I got a functional copy. That was probably due to an unreliable protection scheme. Other than that, I have no complaints about the game.)

I have a complete map of the proving grounds of the Mad Overlord and I will begin mapping the Knight of Diamonds scenario as soon as it is released.

THE PIRATES OF WIZARDOM

On a different note, I was using an A.B.B.S (Apple Bulletin Board Service) called 'PIRATE'S HARBOR.' After reading some of the public bulletins, I ran across the following:

'THE SUBJECT OF SOFTWARE PIRACY HAS BEEN AN OPEN TOPIC OF DISCUSSION ON THE BOARD FOR SOME TIME. IN THE MONTH OF NOVEMBER, A PERSON CAME ON OUR BOARD WHO CALLED HIMSELF 'ROBBING HOOD,' WHO SAID HE LIVED AT RISLEY HALL AT CORNELL, WAS A SOPHOMORE, AND ALSO HIS REAL NAME WAS 'FRED WILLIAMS'. IN EARLY DECEMBER, I FOUND OUT THAT HE WAS REALLY SOMEONE FROM THE COMPANY CALLED SIRTECH, WHICH PRODUCES A GAME CALLED 'WIZARDRY.'

'SOMETIME IN JANUARY, HE FOUND OUT THAT WE KNEW WHO HE WAS, AND AFTER SOME DISCUSSION DECIDED TO 'COME CLEAN.' I TOLD HIM THAT HE COULD TELL HIS STORY AND THAT WE WOULD HAVE AN OPEN DISCUSSION OF THE SUBJECT.'

The preceding bulletin was left by the system operator or 'SYSOP.' After doing a little checking, I learned that 'Robbing Hood' was actually Robert Woodhead and that the reason he logged on to a pirate's bulletin board was to get an idea of the level of piracy concerning his program, Wizardry.

In my opinion, this is one occurrence of deception that I heartily condone. Wizardry is by far the best game program of any type that I have ever played 'bar none'. If Mr. Woodhead keeps producing software of this caliber, I will be glad to pay $50.00, or $100.00 for that matter. Needless to say, I recommend this program to anyone, even if you don't like games of any sort.

SAVAGE ISLANDS

Recently I read somewhere about certain adventurers complaining that Savage Island and Savage Island part

continued on page 50

Site and its elements...

In the last installment I discussed the characteristics of adventure mazes and defined a "room" as just one step or motion in such a maze. This time I'll describe the difference between a site and a site's elements, or objects.

Site means location and in mazes a site is best describe by those features of a "room" that generally cannot be transported out of the "room." The things that can be moved about are called site elements. These are the objects that the adventurer can GET. That means that the site is merely a locale while an element is an object. Most adventurers tend to emphasize objects rather than site because it is easier to interact with objects. After all, ever try to GET a mountain, or DROP a river? Site, therefore, is not usually critical in the unraveling of most adventure mazes that depend heavily upon object possession and manipulation. Site CAN be used more often and in much the same way that site descriptions enhance and project any fantasy's world view. Here are a few examples of how site should be used:

"Snow blankets the ground as the blizzard howls deafeningly" instead of "you freeze to death."

"The ground rumbles menacingly" instead of "you are in a volcano."

"Your feet splash into mud" in a desert, instead of "you sink into quicksand."

"Acrid smoke chokes you" instead of "you are burned alive."

"A towering shadow looms before you. A growl fills the cavern" instead of "a demon grabs you."

With more "story-like" site clues, the adventurer is given the option of resolving the problem before an untimely (and often frequent) demise. It requires a more complex program and results in a far more challenging game.

This must also apply to hi-res adventures, of course. The accompanying graphics should be more detailed, and include more motion than previous adventures. Nor is it necessary to show "top-views" of such mazes in order to incorporate action graphics. A variety of viewpoints would be appropriate and welcomed.

GET GOLD! GET OUT!

Another complaint of mine concerns game objectives. Gathering objects is second nature to adventurers who learn early on that stealing is an acceptable way of life for any adventurer. However, theft is by no means the only way to acquire objects, wealth and fame. An adventurer may buy them. Bartering for goods can be quite complex and a vital strategic element. Treasure hunting is one of the prime attractions of adventures and discovering such a trove makes many boring adventures at least a little less of a bore. But what about other means of gaining possessions?

If one doesn't beg, borrow or steal, then how does one GET them? Well, the adventurer can create them: forge spearheads and swords, string a bow, carve a boat. Some adventures require the player to combine several items to create a new object: a rope plus a spear equals a rope bridge to cross a narrow gorge.

Perhaps the adventurer can tame a wild animal and thereby gain a loyal and useful ally. Or an adventurer can accumulate knowledge, skill and power instead of a pile of gold, weapons, and magical implements.

Maybe the solution is not through gold at all, but rather in the friends and acquaintance you nurture: not WHAT you know, but WHO.

In fact, a good adventure should use all these modes. But keeping track of multi-solutions for any game can be very complex. To keep the program as small as possible, it will be necessary to code all site and object characteristics as well as participant characteristics . This flexibility will allow more than one solution to a particular adventure, resolving the problem of a single solution that often leaves the adventurer lost.

You should be able to move a sword from one room and leave it in another. You should be able to hide possessions that you do not wish to carry. Some objects must be destructable while others must be capable of being combined to form yet another object. And still other objects must be "breakable" into transportable sizes, or simply removable (like jewels from a building).

CODING INFORMATION

One way to code such information is to use strings in applesoft. For example, P$(Ø) would consist of the personal possessions and characteristics of a particular entity such as the adventurer...An example is:

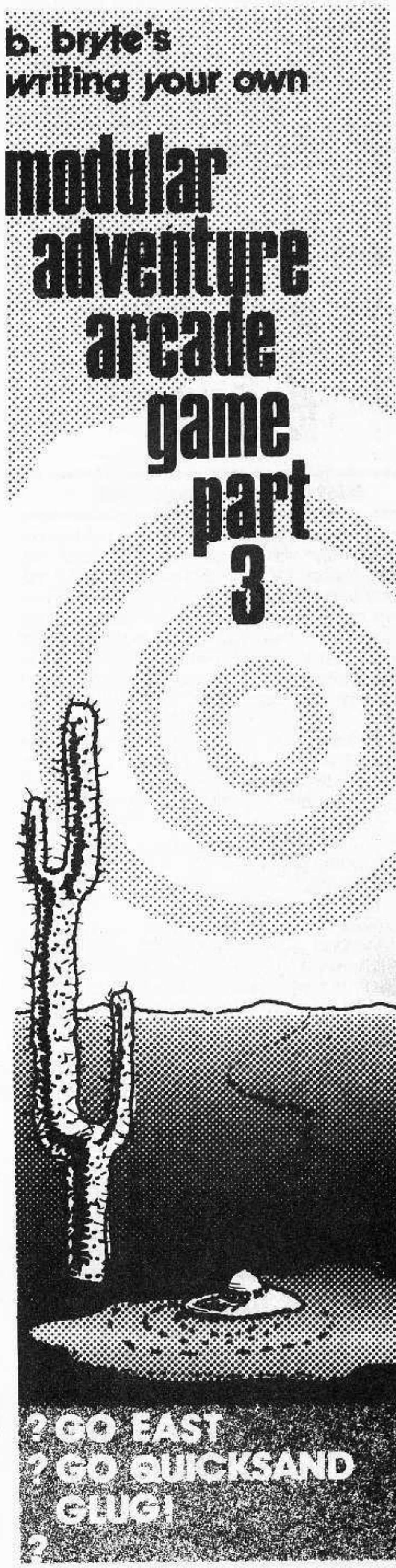
P$(Ø)= 'KDSOI:9LJKØ45ØM..." (23Ø CHARACTERS LONG)

To find out if the adventurer has a knife, for example, the program must go to the knife column (let's say that it's in the 3Ø column). We would extract:

MID$(P$(Ø),3Ø,1).

Now let's say that the character extracted is an "H." That would be a coded symbol for something: the knife is broken, and being held in the left hand. However, to decode it would take a great deal of space. I personally prefer to code all my variables outside of Applesoft, that is, directly into a hex or binary (Ø1ØØ1Ø1Ø) format that I can BLOAD them into a running program without using text files. The byte (Ø-255) code for any particular adventurer (if more than one) would be stored at a specific place in memory and in a particular coded manner. With the full 256 ASCII values available to me in the binary format (instead of a mere 85 inside a string), the possible combinations are far greater. If the

continued on next page



knife column is still column 30, it could be retrieved by a peek (KNIFE) where KNIFE = memory location where the knife data is coded. Let's say that the byte there is 133.

To decode it, let's assume that the following general possession coding was used:

coder	value range	explanation
0	0	not in possession
1	1 - 25	secretly hidden
2	26 - 50	special carrier
3	51 - 75	in the bag
4	76 - 100	pant's pocket
5	101 - 125	shirt pocket
6	126 - 150	head
7	151 - 175	right leg
8	176 - 200	left leg
9	201 - 225	right arm
10	226 - 250	left arm
11	250 - 255	torso

```
CODER  X  25  =  VALUE RANGE
```

If the value is 0 then the adventurer has no knife. If the value is between 226 and 250 then the knife is in or on the left arm. The first part of the code can be obtained by dividing the PEEKed value by 25. The resulting range can then be further decoded into more detail. In the case of the arms, additional code options would be different from the legs or head, etc:

1. shoulder
2. upper arm
3. elbow
4. crook of elbow
5. lower arm
6. wrist
7. back of hand
8. palm
9. thumb

10-13 the fingers
14-25 additional details.

This part of the coding is the "remainder." In this example, the code was 133. Division by 25 gives 5 with a remainder of 7. It decodes as: The adventurer has a knife in the palm of the right hand.

Knife on right arm = 5
In palm = 8

The initial coder (divisor of 25) could apply to all possessions, while each range would have various specific codings.

It may seem very complex, or a case of coded overkill but this allows much greater flexibility and realism. Gone will be the single solution adventures.

The command parser would do all the work of decoding possession codes and reallocating objects and entities. Not only must the command parser translate the commands that an adventurer enters either by keyboard or paddle, but it must also handle all possible errors and give appropriate reward or punishment for an adventurer's creativity in seeking odd-ball solutions when all else fails.

The command parser must be able to use single key commands in cases where response time is measured, and still have an extensive vocabulary and a language algorithm to turn sentences into correct action without misunderstanding (or simply not understanding). And that is the topic of the next installment.

Meanwhile, here is a more complete listing of the Island-Maze maker turned "editor." Unlike the original maze-maker which was a demonstration program used to encourage programmers to look for alternate maze depictions, this "map editor" is specific in its graphic presentation (which defines it severely). I encourage you to come up with your own maze-maker modules.

```
$HEX    < MEMORY DIAGRAM >    DECIMAL

 800 ======================= 2048
     :::(Loader Program>:::::
     ::::::::::::::::::::::::
     :::(Island Data Set>::::
2000 ======================= 5120
     ////////////////////////
     //Hi-Res Page 1 Buffer//
     ////////////////////////
4000 ======================= 10240
     :::(Editor Program>:::::
     ::::::::::::::::::::::::
     :(hi-res page 2 buffer):
     ::::::::::::::::::::::::
```

"LOADER" FOR EDITOR

```
10 D$ = CHR$ (4)
20 POKE 103, 1 : POKE 104, 64
30 POKE 16384, 0
50 PRINT D$ "RUN === "

(=== file name you SAVEed editor under.)
```

To use the Island Map Editor- Displayer, you must run its "loader" program first. The "loader" resets the "beginning of program" pointers to point above Hi-res Page 1. The binary Island Data Set is stored just below this buffer. (See MEMORY DIAGRAM. Notice that the Island Data Set resides just below hi-res page 1, while the program begins on hi-res page 2.)

EDITOR'S NOTE

If you already have the RELIEF MAPPER 1.0, then all you need to do to turn it into the MAP EDITOR is to add in the new lines, redo some other lines and make small changes in a few others.

1. All line numbers that are preceded by a slash need not be redone. They are also used by the MAP EDITOR.
2. All line numbers that are preceded by a dot must be altered a little to make it work with the MAP EDITOR. Examine the line and edit your lines accordingly.
3. YOU MUST REDO THESE LINE in the mapper program because I made some alterations in order to put in the editor:

300 through 390
10000 through 10012

The easiest way to do this it to simply DEL 300,390 and DEL 10000,10012, then type in the new lines.

4. DELete line 10090.

-----BRH

Island Map Maker-Ed

```
  10  GOTO 22000

 -------- GET KEY% OF STROBE -----------
  50  KEY% = PEEK ( - 16384) : IF KEY% >
      127 THEN POKE  - 16368,0
  52  RETURN

 ----- DECODE SITE% TO SITE$ (N) -------
  60  NN = SITE% + 1 : ON NN GOTO 3000,
      3100, 3200, 3300, 3400, 3500, 3600,
      3700, 3800, 3900

 ------ SITE TEXT MAP SYMBOLS ----------
  70  S$ = "--" : RETURN
  71  S$ = ")(" : RETURN
  72  S$ = "<>" : RETURN
  73  S$ = "!!" : RETURN
  74  S$ = "%%" : RETURN
  75  S$ = "**" : RETURN
  76  S$ = "##" : RETURN
  77  S$ = "^^" : RETURN
  78  S$ = "&&" : RETURN
  79  S$ = "==" : RETURN

 ---------- MEM CHECK ------------------
\ 100  IF TMEM% < BMEM% OR TMEM% > EMEM%
       THEN POP : POP : GOTO 1500
\ 110  RETURN
\ 130  IF TMEM% < BMEM% OR TMEM% > EMEM%
       THEN  POP : GOTO 2080
\ 140  RETURN

 ----- MAKE ELEVATION ------------------
\ 200  ELEV% =  INT ( PEEK (ZMEM%) / 10)
       - 10 - INT ( RND (1) * (SLOPE%
       (SIDE) + 1)) - 2
\ 210  IF ELEV% < 0 THEN ELEV% = 0
\ 240  RETURN

 ---------- N-E-S-W SIDES --------------
\ 250  TMEM% = PMEM% - 20
\ 255  RETURN
\ 260  TMEM% = PMEM% + 1
\ 265  RETURN
\ 270  TMEM% = PMEM% + 20
\ 275  RETURN
\ 280  TMEM% = PMEM% - 1
\ 285  RETURN

 --------- TMEM% CALCULATION------------
           OF X, Y, ELEV, & SITE
  300  Y0% = TMEM% - BMEM%
  310  YY% = Y0% / 20 : XX% = Y0% - YY% *
       20
```

How to use: ISLAND EDITOR~MAKER MAP

The first thing that happens when you RUN the program is that the "Master Menu" is displayed.

```
        ISLAND MAZE MENU
-----------------------------------
  < 0 >  HIRES/MENU SWITCH

  < 1 >  MAKE NEW ISLAND

  < 2 >  ADD MOUNTAINS

  < 3 >  EDIT ISLAND DATA

  < 4 >  LOAD ISLAND DATA

  < 5 >  SAVE ISLAND DATA

  < 6 >  EXIT PROGRAM

  < 7 >  DRAW RELIEF MAP


  CHOOSE A NUMBER
```

The program is entirely menu-driven. All you need to do is select the proper number for the job you want done. RUN the program now and follow along with this Tutorial.

First off, let's make an island. That's not as simple as it seems. There is more to making an island than you might suspect (or that I suspected when I started this project).

Using this program, there are at least 2 main steps (more if you wish to see it in hi-res or if you want to customize it). In order to make a complete island, we'll follow these steps:

A. {1} **MAKE NEW ISLAND**
B. {7} **DRAW RELIEF MAP**
C. {0} **HIRES/MENU SWITCH**
D. {2} **ADD MOUNTAINS**
E. {3} **EDIT ISLAND DATA**
F. {5} **SAVE ISLAND DATA**

All the menu choices are independent and, if you are in a hurry, you can use this short-cut method:
A. {1} MAKE NEW ISLAND
B. {2} ADD MOUNTAINS
C. {5} SAVE ISLAND DATA

Or, if you want, you can repeat some of the steps. For instance, you can {5} ADD MOUNTAINS until you get enough.

LET'S MAKE AN ISLAND RIGHT NOW.
Press: {1} MAKE NEW ISLAND.

This subroutine flips to the text page and performs the following steps:

A. It erases the island data-set in memory by turning all the bytes into 124 (line 10020) which is site-type 4 (meadows) and elevation 2.

B. It randomly places large and small patches of site-types all over the data-set. These site-types are:

number	name	map code
4	Meadow	%%
5	Forest	**
6	Farmland	##
7	Desert	~
8	Jungle	&&
9	Swamp	==
0	Water	--

SITE PATCHES APPEAR

```
==   ------########  ******    ****======
==   ----#######     ****    ******====
==-----     ########******      ****====
  ------      #########****   ****    ====
==  ----      ####======************    ====
    --    #####=======**          **&&
         &&&&##--====          ****&&&&&&
####  &&&&&&&&----====----------****&&
########&&&&-------==    ----****&&&&
#####&&&&&&&&------######--**&&&&&&&&&&
####&&&&&&&&  ----######  --****&&&&&&
####&&&&&&         ^^##          &&&&
##    &&&&&&&&&&^^^^^^^^^^      ======
      &&&&&&  &&&&^^^^^^^^^^^  ====  ####
==  &&&&&&&&  &&&&&&^^^^^^     ======
====  &&&&  &&&&&&^^^^^^     ======
==          &&&&&&^^^^     ======  ----
      -----------&&&&&&          ==--------
     ----------  &&&&^^           ------
        --       &&^^         &&&&   ----
                   ^^       &&&&        --
```

This operation is displayed as it occurs.

C. It then creates the island's perimeter and fills in the sea by going in a clock-wise direction. You will see this process, too. Those areas marked as "--" (water) were changed, in the data set, into "100" or "90."

SEASHORE FORMS

```
-------------------------------------------
------------------------**-------------------
--------   ##------****---------****------
--------      ----########----****     ----
--------      ##--=======****--******------
------     ######=======**  --  **---------
----     &&&&##--====          *****&&------
------&&&&&&&&----====-----------**--------
----####&&&&-------==    ----****&&------
------&&&&&&&&------######--**&&&&------
--##&&&&&&&&  ----######  --****&&&&----
----&&&&&&         ^^##          &&------
------&&&&&&&&&&^^^^^^^^^^      ==--------
---------&&&&  &&&&^^^^^^^^^^^  ====------
------&&&&&&  &&&&&&^^^^^^     ======----
----  &&&&----&&&&^^^^^^     ======------
----  ---------&&&&^^^^     --==----------
-----------------&&&&&&      ----==---------
--------------------&&^^   ----------------
----------------------     ----------------
-------------------------------------------
```

itor - Hi-Res Displayer

```
320  XMAP% = XX% * 13 + YY% * 4 - 30
330  IF XMAP% < 0 OR XMAP% > 259 THEN
     POKE TMEM%, 100
350  FACT% =  PEEK (TMEM%) / 10 : ELEV%
     = FACT% - 10
360  IF ELEV% > 8 THEN SNOW = 1
370  SITE% = PEEK (TMEM%) - FACT% * 10
380  YMAP% = YY% * 7 - ELEV% + 13
390  RETURN

----------- NORTH SIDE ----------------
\ 500  TMEM% = PMEM% + 1
\ 510  GOSUB 100
\ 520  ZMEM% = TMEM% + 20
\ 525  IF WIDTH = LIMIT * 2 THEN ZMEM% =
       ZMEM% - 1
\ 530  GOSUB 200
\ 550  PMEM% = TMEM%
\ 590  RETURN

------------ EAST SIDE ----------------
\ 600  TMEM% = PMEM% + 20
\ 610  GOSUB 100
\ 620  ZMEM% = TMEM% - 1
\ 625  IF WIDTH = LIMIT * 2 THEN ZMEM% =
       ZMEM% - 20
\ 630  GOSUB 200
\ 650  PMEM% = TMEM%
\ 690  RETURN

------------ SOUTH SIDE ---------------
\ 700  TMEM% = PMEM% - 1
\ 710  GOSUB 100
\ 720  ZMEM% = TMEM% - 20
\ 725  IF WIDTH = LIMIT * 2 THEN ZMEM% =
       ZMEM% + 1
\ 730  GOSUB 200
\ 750  PMEM% = TMEM%
\ 790  RETURN

------------- WEST SIDE ---------------
\ 800  TMEM% = PMEM% - 20
\ 810  GOSUB 100
\ 820  ZMEM% = TMEM% + 1
\ 825  IF WIDTH = LIMIT * 2 THEN ZMEM% =
       ZMEM% + 20
\ 830  GOSUB 200
\ 850  PMEM% = TMEM%
\ 890  RETURN

------------ LOOK AROUND --------------
900  PMEM% = TMEM% : GOSUB 300 : GOSUB
     950
```

continued on next page

4. It redraws the entire coded island from top to bottom.

```
----------------------------------------
-------------------%%-------------------
--------%%%%------%%$$--------%%%%------
--------%%%%----%%####%%----%%$$%%%%----
--------%%%%%%--======$$%%--$$$$%%------
------%%%%####%%======$$%%--%%%%--------
----%%%%&&&&##--====%%%%%%%%%$$$$%%------
------%%&&&&&&----====--------%%--------
----%%##&&&&-------==%%%%----$$$$%%------
------%%&&&&&&-------######--$$&&%%------
--%%&&&&&&&&%%----######%%--$$$$&&%%----
----%%&&&&%%%%%%%%^^##%%%%%%%%%%%%------
------%%&&&&&&&&^^^^^^^^^^%%%%%%--------
--------%%&&%%&&&&^^^^^^^^^^%%==%%------
------%%&&%%%%&&&&&&^^^^^^%%%%====%%----
----%%%%%%----%%&&^^^^^^%%%%====%%------
----%%--------%%&&^^^^%%%%--%%----------
----------------%%&&&&%%%%----%%--------
-------------------%%^^%%---------------
-------------------%%%%-----------------
----------------------------------------
```

5. And finally, it returns you to the menu.

PRESS {3} EDIT ISLAND DATA

Now you can edit the island you created. This island has no mountains, yet, so this step is for those who want to change the island before the mountain-building begins (which builds ONLY on the available ground).

Notice that the island data code is rePRINTed on the text page. The flickering, clicking cursor in the upper right corner (which means you are in the CURSOR MODE) can be moved over any part of the island by using the arrow keys to go left and right, the {RETURNT} key to go up, and the SPACE key to go down. As you move the cursor, the site-type and elevation is displayed just below the map.

Move to the spot you want to edit and press the {ESC} key. It will beep, and then display the cursor position only by flashing it (no more noise!). You are in the EDIT MODE.

Pressing Ø through 9 will change that site to the type you select and then return you to the buzzing CURSOR MODE.

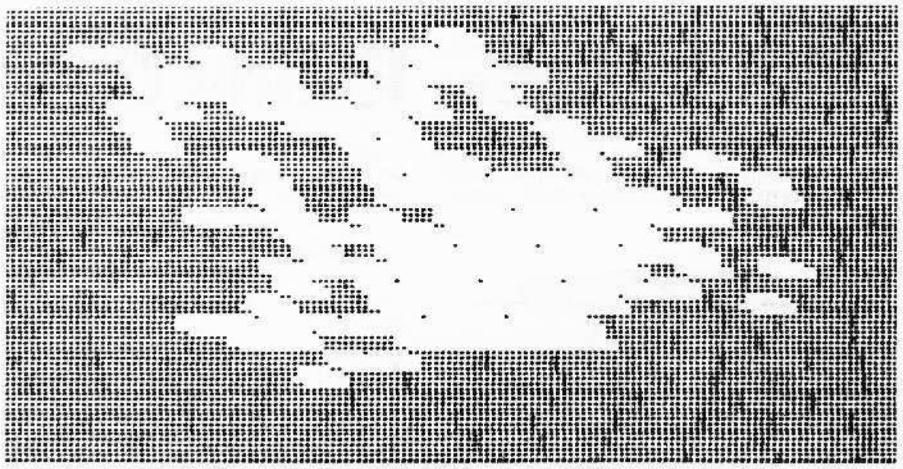
To edit the elevation, press the Asterisk key (use the shift).

The four directions give you the elevation on each side of the cursor position, and the cursor's elevation is shown to the right just above its site type. To select an elevation, press the {SPACE} key until the inverse marker moves up the scale (1 - 12) to the elevation you desire. Should you exceed 12, the marker starts again at 1. Press {RETURN} to select the elevation. You will be returned to the noisy cursor mode. The elevation, as you can see, has been changed.

If you selected an elevation above 8, the cursor will have a different clicking sound. Move the cursor aside and you will see why. All site symbols marking an elevation above 8 are in inverse.

Did you notice that you couldn't select an elevation of zero? Actually, you can, but not from the EDIT ELEVATION MODE. Go to the normal EDIT MODE by pressing ESC while in the CURSOR MODE. All you have to do is select "Ø} WATER."

To return to the master menu, press "X" while in the CURSOR MODE.

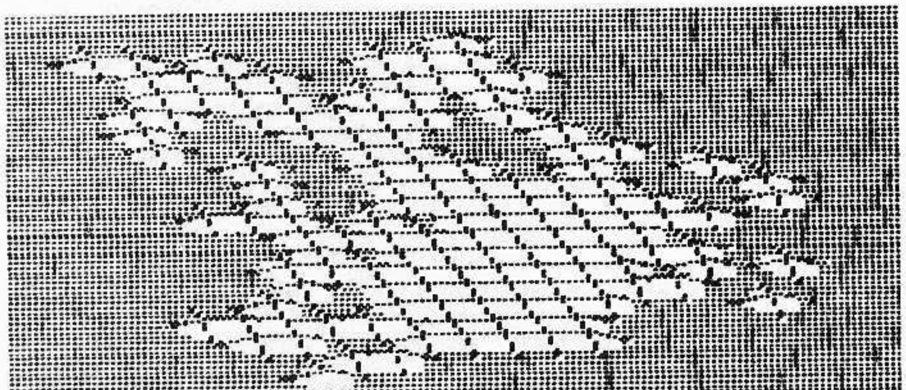


step 1: A Naked Island... inverse image

```
 910  FOR TEST = 1 TO 4 : ON TEST GOSUB
      250, 260, 270, 280 : GOSUB 300 :
      ELEV$ = STR$ (ELEV%) : IF TMEM% <
      BMEM% OR TMEM% > EMEM% THEN ELEV$
      = "**"
 920  ON TEST GOSUB 960, 970, 980, 990
 930  NEXT : TMEM% = PMEM% : RETURN
 950  INVERSE : VTAB 23 : HTAB 22 :
      PRINT "ELEVATION: " ELEV% ;: VTAB
      24 : HTAB 22 : PRINT "SITE: "
      SITE$ (NN) ;: NORMAL
 960  VTAB 23 : HTAB 1 : PRINT "NORTH: "
      ELEV$ ;: RETURN
 970  VTAB 24 : HTAB 12 : PRINT "EAST: "
      ELEV$ ;: RETURN
 980  VTAB 23 : HTAB 11 : PRINT
      "SOUTH: " ELEV$ ;: RETURN
 990  VTAB 24 : HTAB 2 : PRINT "WEST: "
      ELEV$ ;: RETURN

 --------- SLOPES FOR EACH SIDE --------
\1000  FOR SIDES = 1 TO 4 : SLOPE%
       (SIDES) = RND (1) * 4 + 1 : NEXT
       SIDES

 ------------- MAKE MOUNTAINS -----------
\1100  R% = RND (1) * 3 + 3 : FOR
       MOUNTAIN = 1 TO R%
\1110  ZTIP% = RND (1) * 400 + BMEM% - 1
\1120  ELEV% = RND (1) * 5 + 8
\1130  IF  PEEK (ZTIP%) < 110 OR  PEEK
       (ZTIP%) > 180 THEN 1110
\1140  ELEV% =  RND (1) * 5 + 8
•1150  POKE ZTIP%, (ELEV% + 10) * 10 + 3
•1200  FOR LIMIT = 1 TO 4
\1210  PMEM% = ZTIP% - 20 * LIMIT -
       LIMIT
\1300  FOR SIDES = 1 TO 4
\1400  FOR WIDTH = 1 TO LIMIT * 2
\1410  ON SIDES GOSUB 500, 600, 700, 800
\1430  ZELEV% = (ELEV% + 10) * 10
\1440  IF  PEEK (PMEM%) < 110 THEN 1500
•1450  IF  PEEK (PMEM%) < ZELEV% THEN
       POKE PMEM%, ZELEV% + INT ( RND
       (1) * 2 + 5)
\1490  GOSUB 2000
\1500  NEXT WIDTH
\1510  NEXT SIDE
\1520  NEXT LIMIT
\1530  NEXT MOUNTAIN
•1540  HGR : GOSUB 20000 : GOSUB 21000
•1590  GOTO 22000


            PLOT, GRAPH, & DRAW
 -------LINES TO NEW ELEVATIONS---------
\2000  HCOLOR= 3
\2005  TMEM% = PMEM% : GOSUB 300
\2010  HPLOT XMAP%, YMAP% : X% = XMAP% :
       Y% = YMAP%
\2020  FOR TEST = 1 TO 4 : ON TEST GOSUB
       250, 260, 270, 280
\2030  GOSUB 130: GOSUB 300
\2040  IF ELEV% > 8 THEN SNOW = 1
\2045  IF TMEM% < 100 THEN 2080
\2050  HCOLOR= 2 : IF SNOW > 0 THEN
       HCOLOR= 3 : SNOW = 0
•2055  IF  ABS (X% - XMAP%) > 14 THEN
       2080
\2060  HPLOT X%, Y% TO XMAP%, YMAP%
\2080  NEXT TEST
\2090  RETURN

 --------- SITE DESCRIPTIONS -----------
 3000  SITE$ (NN) = "OPEN WATER." :
       RETURN
 3100  SITE$ (NN) = "A CASTLE." : RETURN
 3200  SITE$ (NN) = "A TOWN." : RETURN
 3300  SITE$ (NN) = "UNKNOWN." : RETURN
 3400  SITE$ (NN) = "A MEADOW." : RETURN
 3500  SITE$ (NN) = "A FOREST." : RETURN
 3600  SITE$ (NN) = "FARMLAND." : RETURN
 3700  SITE$ (NN) = "A DESERT." : RETURN
 3800  SITE$ (NN) = "A JUNGLE." : RETURN
 3900  SITE$ (NN) = "A SWAMP." : RETURN

 ----------- PRINT SITE INFO ------------
 5000  HOME : POKE 216, 0 : D3 = 1 :
       GOSUB 12000 : D3 = 0 : POKE 34,
       20 : HOME : EDIT% = 1
 5010  TMEM% = BMEM% + 1 : FM% = 1
 5020  HOME : PMEM% = TMEM%
 5030  GOSUB 300 : VTAB 21 : HTAB 33 :
       PRINT "ELEV: " ELEV% ;
 5040  NN = 0 : GOSUB 60
 5050  VTAB 21 : HTAB 13 : PRINT
       "~~~~~~~~~~~~~~~~~~~~" ;: HTAB 13 :
       PRINT "SITE: " ;: INVERSE : PRINT
       SITE$ (NN) ;: NORMAL
 5090  ON SITE% + 1 GOSUB 70, 71, 72,
       73, 74, 75, 76, 77, 78, 79
 5200  GOSUB 50 : VT% = YY% + 1 : HT% =
       2 * XX% + 1
```

```
5210  VTAB VT% : HTAB HT% : INVERSE :
      PRINT S$ ;: Z = PEEK ( - 16336) -
      PEEK ( - 16336) : NORMAL : VTAB
      VT% : HTAB HT% : PRINT S$ ;: IF
      ELEV% > 8 THEN INVERSE : Z =
      PEEK ( - 16336) : VTAB VT% : HTAB
      HT% : PRINT S$ ;: NORMAL
5220  IF KEY% = 216 THEN  GOTO 22000
5230  IF KEY% = 141 THEN 5800
5240  IF KEY% = 155 THEN 5900
5250  IF KEY% = 149 THEN 5500
5270  IF KEY% = 160 THEN 5600
5290  IF KEY% = 136 THEN 5700
5490  GOTO 5200

----------- KEY = -=> -----------------
5500  TMEM% = TMEM% + 1
5550  IF TMEM% > EMEM% THEN TMEM% =
      TMEM% - 400 : PRINT G$ ;
5590  GOTO 5020

----------- KEY = <SPACE> -------------
5600  TMEM% = TMEM% + 20
5650  IF TMEM% > EMEM% THEN TMEM% =
      TMEM% - 400 : PRINT G$ ;
5690  GOTO 5020

----------- KEY = <=- -----------------
5700  TMEM% = TMEM% - 1
5750  IF TMEM% < BMEM% THEN TMEM% =
      TMEM% + 400 : PRINT G$ ;
5790  GOTO 5020

----------- KEY = <RETURN> ------------
5800  TMEM% = TMEM% - 20
5850  IF TMEM% < BMEM% THCN TMEM% =
      TMEM% + 400 : PRINT G$ ;
5890  GOTO 5020

----------- KEY = <ESC>> --------------
5900  VTAB 21 : HTAB 1 : PRINT
      "<EDIT MODE>" G$ G$ G$ ;: VTAB
      VT% : HTAB HT% : FLASH : PRINT S$
      ;: NORMAL
5910  VTAB 22 : HTAB 1 : PRINT
      "1>CASTLE~~~4>MEADOW~~~7>DESERT~~
      ~9>SWAMP2>TOWN~~~~~5>FOREST~~~8>J
      UNGLE~~~0>WATER3>SPECIAL~~6>FARM~
      ~~~~*>ELEVATION~~" ;
5920  GOSUB 50
5940  IF KEY% > 175 AND KEY% < 186 THEN
      ES% = KEY% - 176 : GOTO 6500
5950  IF KEY% = 155 THEN  GOTO 5020
5960  IF KEY% = 170 THEN  GOTO 6000
5970  GOTO 5920

------------- EDIT ELEVATION -----------
6000  HOME : GOSUB 900
6010  VTAB 21 : FOR A = 1 TO 12 : HTAB
      A * 3 : PRINT A ;: NEXT : A = 1 :
      NORMAL
6020  GOSUB 50 : IF KEY% = 141 THEN 6070
6030  IF KEY% <  > 160 THEN 6060
6040  B = A : A = A + 1 : IF A > 12
      THEN A = 1 : PRINT G$ ;
6050  HTAB B * 3 : PRINT B ;
6060  INVERSE : HTAB A * 3 : PRINT A ;:
      NORMAL : GOTO 6020
6070  IF SITE% = 0 THEN SITE% = RND (1)
      * 5 + 5
6080  IF A > 8 THEN SITE% = 5 : IF A >
      10 THEN SITE% = 4
6090  POKE TMEM%, (A + 10) * 10 + SITE%
      : PRINT G$ G$ G$ ;: GOTO 5020

-------------- EDIT SITE ---------------
6500  IF ELEV% < 1 AND ES% > 0 THEN
      FACT% = 12 : PRINT G$
6510  IF ES% = 0 THEN FACT% = 10
6520  TPK% = FACT% * 10 + ES%
6550  POKE TMEM%, TPK% : PRINT G$ ;
6560  GOTO 5020

---------------- E N D -----------------
9000  END

-------------- VARIABLES ---------------
10000  FM% = 0
10001  IF M1 > 0 THEN M1 = 0 : GOTO
       5000
10005  HOME : VTAB 10 : HTAB 10 : PRINT
       "CLEARING MAZE MEMORY" : POKE
       216, 0
10010  G$ = CHR$ (7) : D$ = CHR$ (4)
10011  MAX = 6
10012  EDIT% = 1
10020  FOR A = BMEM% TO EMEM% : POKE A,
       124 : NEXT A
10021 TEXT : HOME : D3 = 0 : GOSUB 12000
       : GOSUB 13000
10030  AA = BMEM% : BB = BMEM% + 19 :
       TEST = MAX : GAP = 20 : FOR IN =
       AA TO BB : GOSUB 11000 : NEXT IN
```

continued on next page


step 2: A Flat Island Topography...

Now let's draw our island on the hi-res page.

Press {7} DRAW RELIEF MAP. When you do this, the following steps are performed:

1. The hi-res page is displayed and erased.
2. The screen is turned blue and then dark blue.
3. Black patches appear here and there.
4. Beeping is sometimes heard.

The relief map drawing process consists of:

1. Creating a sea of blue.
2. Plotting the black land masses.
3. Drawing the horizontal lines.
4. Drawing the "vertical" lines.
5. Returning immediately to the main menu.

Notice that the vertical relief lines are at an angle? That's to give a sense of "perspective" to the map, as if you are viewing the island, not from directly on top, nor from directly in front, but from above, to the front, and a little to one side.

The process, I admit, is slow in Applesoft (so if you have access to one of the many applesoft compilers, try compiling the program. See the article on Applesoft compilers in the Apple Digest.)

The beeping you sometimes hear occurs only when there is a land mass that would have plotted off the screen. That particular piece of land was, at the sound of the beep, being edited out of the island data set and replaced by water.

If you have been following along pressing the proper keys, you will have already created an island, played with it a bit with the editor, and now have it drawn on the hi-res page.

When you are finally returned to the master menu, press {0} HI-RES/MENU SWITCH, and then press {2} ADD MOUNTAINS.

WARNING: If you are not viewing the hi-res page when you press {2}, anything drawn on the hi-res page will be erased before the mountain-building process begins.

As described in my earlier article, the mountain is built around its tip and all lines going up to and coming down from any elevation above 8 are plotted in white (snow-capped?). When the process is complete, it will clear the page and redraw the entire island with new relief lines reflecting the new elevations.

If you are impatient, you can end any step by using the RESET key. Then to return to the menu, simply RUN it again. The island data-set is not destroyed by RESET because it is stored in binary form just below the hi-res page. Whenever you want to stop any process in the middle, just RESET.

When the mountain building process is complete and you are returned to the menu (or you ended it with a RESET, followed by a RUN), you can build more mountains, or go into the editor, or you can SAVE the island data set.

Just as an experiment, press {2} while you are viewing the master menu. See what happens? You are viewing the actual mountain-building process in the "raw." Notice how the relief lines avoid the water?

Return to the menu (wait till it's done, or use RESET). Now let's edit it again.

Press {3}. If you have built any white capped mountains, you should now see inverse characters mixed among the normal ones. Those are your mountains. You should also notice a new site symbol: the exclamation point. It refers to a mountain peak. Try editing the mountain. Add in your own mountains. Surround the island in a mountain range. Put in a huge lake in its center. Create huge deserts, slender jungles, winding forests. Now is also the time to put in castles and towns.

There are some limitations to the editor (which you may remove if you like), and some quirks. For example, selection of any elevation to cover a site that was formerly water will result in the generation of a random site-type that is not water. And selection of water results in an elevation of zero. If you select an elevation above 8, then a random site-type is also generated. When replacing water with any site-type, the elevation becomes 2. Finally, you cannot build land over deep water and if you try, nothing will change.

Use the editor to move through the island, customizing it. To return to the master menu, press "X" while in the CURSOR MODE. To get out of the EDIT MODE without changing the site, use {ESC} (just as you did to enter the EDIT MODE.

SAVE the island data set by pressing {5} SAVE ISLAND DATA. If you want to reclaim an island previously saved, press {6}LOAD ISLAND DATA.

It's generally a good idea to save your island before you edit or add mountains just in case you don't like the results. In this way you can just retrieve the original island data and start again.

Now that you have built some unique islands, what are you going to do with them? That is the topic of the next installment:

THE COMMAND PARSER

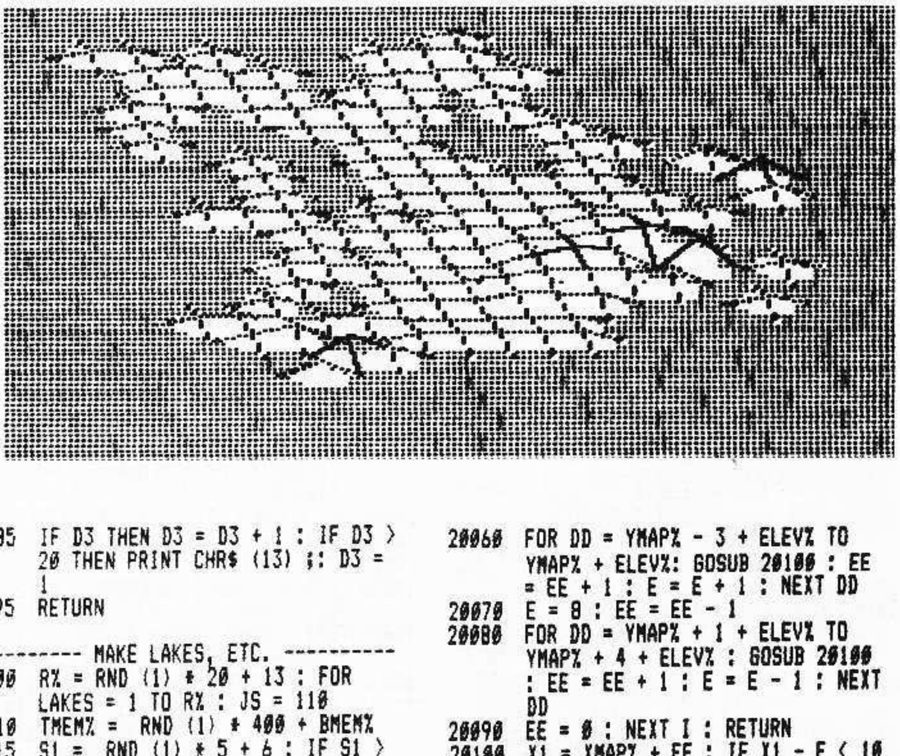
The command module will let you move through the island using both "single-key" commands and "sentence" commands. It is the command parser that permits the adventurer to engage the elements of site, object, and adventure!

step 3
A Few Mountains...

```
\10040  AA = BB : BB = BMEM% + 399 :
        = AA TO BB STEP 20 : GOSUB 11000
        : NEXT IN
\10050  AA = BB : BB = BMEM% + 380 :
        TEST = MAX : GAP =  - 20 : FOR
        IN = AA TO BB STEP - 1 : GOSUB
        11000 : NEXT IN
\10060  AA = BB : BB = BMEM% : GAP = 1 :
        TEST = MAX : FOR IN = AA TO BB
        STEP  - 20 : GOSUB 11000 : NEXT
        IN
•10080  D3 = 1 : GOSUB 12000 : GOTO
        22000
\11000  R% = RND (1) * MAX + 1 : IF  ABS
        (R% - TEST) > 2 THEN 11000
•11020  TEST = R% : EDGE% = IN + (GAP *
        TEST) : POKE EDGE%, 134 : JS =
        100 : FOR JJ = EDGE% - GAP TO IN
        STEP - GAP : POKE JJ, JS
 11030  JR = RND (1) * 5 : IF JR < 1
        THEN JS = 90
 11050  I = JJ : GOSUB 12020
 11080  NEXT JJ
\11090  RETURN
```

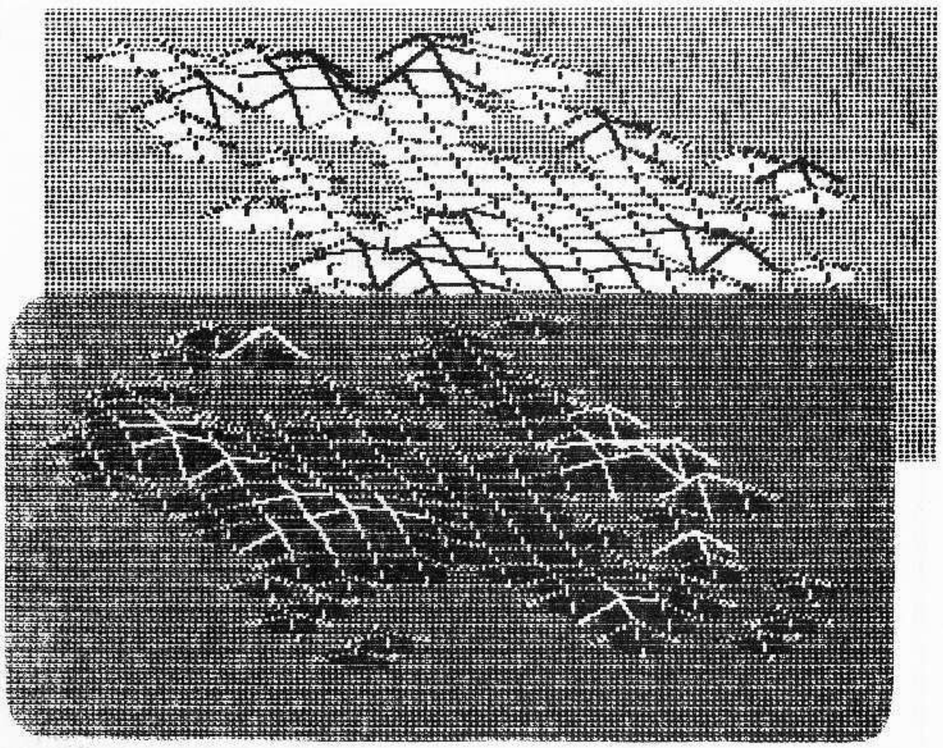
This symbol (~) is used to indicate a space. When entering this program, just type in a space in place of ~.

```
---------- PRINT MAP BACKGROUND ---------
12000  HOME : VTAB 21: PRINT
       "--WATER~~~==SWAMP~~~&&JUNGLE~~*
       *FOREST~" : PRINT
       "^^DESERT~~%%MEADOW~~##FARM~~~~<
       >TOWN~~~" : PRINT
       ")(CASTLE~~!!SPECIAL SITE"
12005  VTAB 1
12010  PRINT CHR$ (13) ;: FOR I = BMEM%
       TO EMEM%
12020  Y0 = I - BMEM% : YY% = Y0 / 20 :
       XX% = Y0 - YY% * 20
12030  VTAB YY% + 1 : HTAB 2 * XX% + 1
       : T% = PEEK (I)
12040  TT% = T% / 10 : SITE% = T% - TT%
       * 10 : ELEV% = TT% - 10 : IF
       ELEV% > 8 THEN  INVERSE
12050  ON SITE% + 1 GOSUB 70, 71, 72,
       73, 74, 75, 76, 77, 78, 79 :
       PRINT S$ ;: NORMAL
12060  IF D3 THEN  NEXT I
```



```
12085  IF D3 THEN D3 = D3 + 1 : IF D3 >
       20 THEN PRINT CHR$ (13) ;: D3 =
       1
12095  RETURN

------------ MAKE LAKES, ETC. -----------
13000  R% = RND (1) * 20 + 13 : FOR
       LAKES = 1 TO R% : JS = 110
13010  TMEM% =  RND (1) * 400 + BMEM%
13015  S1 =  RND (1) * 5 + 6 : IF S1 >
       9 THEN JS = 100 : S1 = 0
13020  FOR L = 0 TO  RND (1) * 3 + 3
13030  FOR LL =  -  RND (1) * 2 + 2 TO
       RND (1) * 2 + 2 : PMEM% = TMEM%
       + LL + L * 20
13040  IF PMEM% < BMEM% OR  PMEM% >
       EMEM% THEN 13070
13050  POKE PMEM%, JS + S1
13060  I = PMEM% : GOSUB 12020
13070  NEXT LL
13080  NEXT L, LAKES
13090  RETURN

----------- DRAW ENTIRE MAP ------------
•20000  HGR : HCOLOR= 2 : HPLOT 0,0 :
        CALL 62454 : HCOLOR= 0 : FOR A =
        0 TO 179 STEP 2 : HPLOT 0, A TO
        279, A : NEXT A
\20010  FOR I = BMEM% TO EMEM%
 20030  TMEM% = I : GOSUB 300
 20040  IF ELEV% = 0 OR SITE% = 0 THEN
        20090
 20050  E = 4
 20060  FOR DD = YMAP% - 3 + ELEV% TO
        YMAP% + ELEV%: GOSUB 20100 : EE
        = EE + 1 : E = E + 1 : NEXT DD
 20070  E = 8 : EE = EE - 1
 20080  FOR DD = YMAP% + 1 + ELEV% TO
        YMAP% + 4 + ELEV% : GOSUB 20100
        : EE = EE + 1 : E = E - 1 : NEXT
        DD
 20090  EE = 0 : NEXT I : RETURN
 20100  X1 = XMAP% + EE : IF X1 - E < 10
        OR X1 + E > 265 THEN POKE I, 100
        : PRINT G$ : RETURN
 20110  HPLOT X1 - E - 3,DD TO X1 + E -
        2, DD
 20190  RETURN

----------- DRAW RELIEF MAP ------------
21000  HOME
21010  FOR A = BMEM% TO BMEM% + 380
       STEP 20 : OELEV% = 0 : FOR AA =
       0 TO 19 : GOSUB 21100 : NEXT AA,
       A
21020  FOR A = BMEM% TO BMEM% + 19 :
       FOR AA = 0 TO 380 STEP 20 :
       GOSUB 21100 : NEXT AA, A
21090  RETURN
21100  TMEM% = A + AA : GOSUB 300 : IF
       XMAP% < 0 OR XMAP% > 270 THEN
       21180
21120  HCOLOR= 1 : IF OELEV% > 8 OR
       ELEV% > 8 THEN  HCOLOR= 3
21130  IF OELEV% < 1 AND ELEV% < 1 THEN
       21180
```

```
21140  IF  ABS (X% - XMAP%) > 14 OR ABS
       (Y% - YMAP%) > 18 THEN 21180
21150  IF X% < 10 OR Y% < 10 THEN RETURN
21160  HPLOT X%, Y% TO XMAP%, YMAP%
21180  DELEV% = ELEV% : X% = XMAP% : Y%
       = YMAP%
21190  RETURN

-------------- MENU -------------------
22000  TEXT : HOME : POKE 216, 0
22020  G$ = CHR$ (7) : D$ = CHR$ (4) :
       HIRES = 1
22030  BMEM% = 7790 : EMEM% = BMEM% +
       399 : REM  (7790 = $1E6E)
22040  DASH$ = "" : FOR A = 1 TO 40 :
       DASH$ = DASH$ + "-" : NEXT
22100  A$ = "ISLAND MAZE MENU" : GOSUB
       22950
22140  PRINT "< 0 >  HIRES/MENU SWITCH"
       : PRINT
22150  PRINT "< 1 >  MAKE NEW ISLAND" :
       PRINT
22155  PRINT "< 2 >  ADD MOUNTAINS" :
       PRINT
22160  PRINT "< 3 >  EDIT ISLAND DATA"
       : PRINT
22170  PRINT "< 4 >  LOAD ISLAND DATA"
       : PRINT
22180  PRINT "< 5 >  SAVE ISLAND DATA"
       : PRINT
22190  PRINT "< 6 >  EXIT PROGRAM" :
       PRINT
22195  PRINT "< 7 >  DRAW RELIEF MAP" :
       PRINT : PRINT
22200  PRINT "CHOOSE A NUMBER   "
22210  GET A$ : IF A$ = "0" THEN A = 0
       : GOTO 22220
22215  A =  VAL (A$) : IF A < 1 OR A >
       7 THEN 22210
22220  HOME : ON A + 1 GOTO 22750,
       22300, 22800, 22400, 22500,
       22600, 22700, 22900
22230  GOTO 22200
22300  A$ = "PROGRAM: ISLAND MAKER" :
       GOSUB 22950 : GOSUB 30000 : GOTO
       10000
22400  A$ = "PROGRAM: MAP EDITOR" : M1

       = 1 : GOSUB 22950 : GOSUB 30000
       : GOTO 10000
22500  A$ = "LOADING ISLAND DATA" :
       GOSUB 22950
22550  PRINT D$ "CATALOG" : PRINT
22560  INPUT "<FILE NAME?>  ISLE." ;
       AN$ : GOSUB 22850
22570  PRINT D$ "BLOADISLE." AN$
       ",A$1E6E"
22590  GOTO 22000
22600  A$ = "SAVING ISLAND DATA" :
       GOSUB 22950
22650  INPUT "FILE NAME: ISLE." ; AN$
22660  PRINT D$ "BSAVEISLE." AN$
       ",A$1E6E,L400"
22690  GOTO 22000
22700  HOME : END
22750  HIRES = - HIRES : IF HIRES < 1
       THEN POKE - 16297, 0 : POKE
       - 16304, 0 : GOTO 22210
22760  GOTO 22000
22800  IF HIRES > 0 THEN  HGR
22810  GOTO 1000
22850  IF AN$ = "" THEN  POP : GOTO
       22000
22860  ONERR  GOTO 22890
22870  RETURN
22890  POP : HOME : GOTO 22000
22900  HGR : GOSUB 20000 : GOSUB 21000
       : FOR A = 1 TO 1000 : NEXT :
       GOTO 22000
22950  HTAB 20 - LEN (A$) / 2 : PRINT
       A$ : PRINT DASH$ : RETURN
22990  END
30000  FOR I = 1 TO 1000 : NEXT :
       RETURN
```



INPUT

continued from page 5

authors profit is 500 * ($50-$10) = $20,000. At $25 per hour, did the author really spend 800 hours improving his product? If he only spent 200 hours, does he believe his effort worth $100 per hour?

In contrast, the most expensive selection in the Hardcore Program Library cost $12 of which $6 is postage and handling, $4 is the author's royalty, and $2 is the cost of printing documentation. Adding a second selection increases the cost by only $4, not another $12. This is software that's too cheap to steal. Even the maligned folks at Call-Apple offer few such bargains. If the program is copied from a friend, the Honor Royalty System requires only that the recipient send the author a $1 royalty. The minority of computerists who might use a copied program more than once and not send in the buck would steal at any price.

Software sellers must realize that overpriced products encourage piracy and that copy protection only serves to frustrate the honest user. I would have thought that a major software producer like Apple Computer, Inc. would recognize this. What's the best selling Apple word processor? AppleWriter. The best selling assembler? DOS Toolkit. Operating system? DOS 3.3. All reasonably-priced, well-documented and easily-archived. Alas, Apple has caved in to the fear and greed of independent software developers by copy-protecting and price-inflating its more recent offerings.

The software retailer, a mainstay of user-support, gets squeezed from both ends. To buy, he must pay stratospheric prices for decent software. To sell, he must either drastically cut margins or figure how to prepare the slow movers for dinner.

Until the software producers recognize that we're in a recession and inflation is no longer an excuse for premium prices, keep both hands on your pocketbook.

John B. Matthews, MD.
Dayton, OH

On The Soapbox

Dear Chuck,

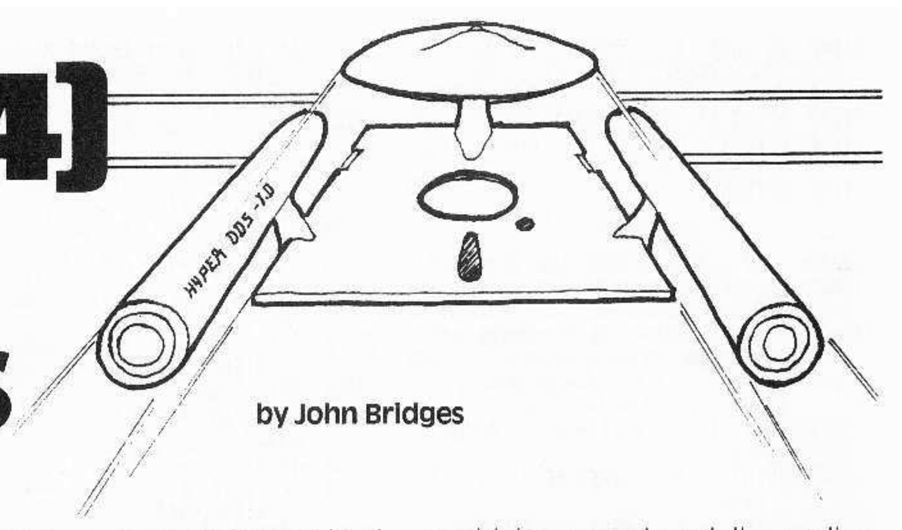
Here is your check for a year's subscription. I hope that there are enough of us to keep this excellent and long-needed publication alive.

Do not despair of advertising, even from software houses. If the magazine is successful, your kickoff article on censorship, and your other articles on reasons for creating backups will eventually be cited as excuses for their placing ads with you. Whatever their prejudices, they will not miss an opportunity to hawk their wares to the largest possible community. In fact, some of them might even rise to the challenge and revel in possessing

continued on page 44

CHR$ (4)

Speeding Up Disk I/O With HyperDOS



by John Bridges

At some time or other, most users have all noticed that FID will load a file it is copying a lot faster than DOS will load that same file. (Is everyone back from checking?) If FID can do it, why can't DOS?

When DOS loads a file, it calls a general routine that reads a range of bytes from the file that is open, which in turn calls the "Filemanager" to do the actual work. The loop for reading those bytes in the filemanager is extremely inefficient when it comes to a large range of sequential bytes. If the general "read routine" was replaced with a different routine that read the file sector-by-sector, as opposed to byte-by-byte, a large speed increase would result. Enter HYPER-DOS.

HYPER-DOS will link into the LOAD, RUN, BLOAD and BRUN routines. When a file is put into memory, the track/sector list and the first sector of the program are loaded into DOS and a check is performed to see if it is the last sector to be loaded. If so, DOS will complete the LOAD. Otherwise, HYPERDOS will move the data to the correct place in memory. Each subsequent sector is loaded directly into the correct location and a check is performed only once per sector to see if it is the last sector of the file. On the last sector, only the portion of the sector that is used is decoded.

Compared to standard DOS, HYPER-DOS offers a speed increase of up to 5 times for loading Binary and Applesoft BASIC files. It does not effect text file or Integer BASIC access, nor does it affect file save times.

A 32K file (131 sectors) takes about 30 seconds to load with normal DOS and about 7 seconds with HYPER-DOS.

All DOS commands are supported and it is compatible with most unprotected programs that run under DOS 3.3. In the writing of HYPER-DOS, compatibility with existing programs was the number one priority.

Unfortunately, a choice had to be made as to where to put the new load routine. The disk format (INIT) routines could be overwritten or the DOS buffers could be moved and the routine inserted below DOS. Moving the DOS buffers seemed the best choice.

```
                *--------------------------------
                * >HYPER-DOS<     BY JOHN BRIDGES
                *--------------------------------
                * THIS ROUTINE IS ASSEMBLED TO RUN BENEATH DOS THEN
                * POINTERS ARE CHANGED SO THAT DOS WILL REMEMBER THE
                * CODE WHEN A DISK IS INITIALIZED.
                * INSERT THE ADDRESS OF THE LOAD ROUTINE AT $A448
                * AND THE ADDRESS OF THE "BLOAD" ROUTINE AT $A38C
                *--------------------------------
                          .OR $9C01     PROGRAM ORIGIN
                          .TA $800      TARGET ADDRESS FOR OBJECT CODE
0067-           START     .EQ $67       APPLESOFT PROGRAM START ADDRESS
00FB-           PTR       .EQ $FB       A TEMPORARY ALL PURPOSE POINTER
00FD-           PTR2      .EQ PTR+2     POINTER USED IN MOVING DATA
00FF-           TEMP      .EQ PTR2+2    USED FOR SAVING THE YREG
03E3-           IOB.LOC   .EQ $3E3      RETURNS LOCATION OF IOB IN A,Y
B5CB-           DA.BUF    .EQ $B5CB     LOCATION TO PUT DATA
B7F0-           RW.BUF    .EQ $B7F0     WORK BUFFER
A702- ✓         PRT.ERR   .EQ $A702     DOS ERROR PRINTING ROUTINE
A2EA-           CLOSE     .EQ $A2EA     CLOSE FILE IN USE
B7EC-           RW.TRK    .EQ $B7EC     TRACK CURRENTLY ON
B7ED-           RW.SCT    .EQ $B7ED     CURRENT SECTOR
B7F9-           OFFSET    .EQ $B7F9     NUMBER OF BYTES TO SKIP
B7FA-           MOV.NUM   .EQ $B7FA     NUMBER OF BYTES TO MOVE
B7B5-           RWTS      .EQ $B7B5     READ IN TRACK/SECTOR
B5C9-           TS.BUF    .EQ $B5C9     BUFFER FOR THE T/S LIST
B7EB-           RWTS.VOL  .EQ $B7EB     VOLUME NUMBER OF DISK
AA60-           F.LEN     .EQ $AA60     LENGTH OF FILE IN BYTES
AA72-           B.ADDR    .EQ $AA72     START LOCATION OF BINARY FILE
A471- ✓         END.LD    .EQ $A471     READ IN BINARY FILE NORMALLY
                *--------------------------------
                * LOAD AN APPLESOFT PROGRAM
                *--------------------------------
9C01- A9 02     LOAD      LDA #2        LENGTH BYTES ON FIRST SECTOR
9C03- 8D F9 B7            STA OFFSET
9C06- A9 FD               LDA #$FD      NUMBER OF BYTES TO MOVE
9C08- 8D FA B7            STA MOV.NUM
                *--------------------------------
9C0B- A5 67               LDA START     PROGRAM START ADDRESS (LO-BYTE)
9C0D- 85 FD               STA PTR2      MOVE TO LOCATION
9C0F- A5 68               LDA START+1   PROGRAM START ADDRESS (HI-BYTE)
9C11- 85 FE               STA PTR2+1    MOVE TO LOCATION
9C13- 4C 41 9C            JMP SCT.LD    GOTO COMMON LOAD ROUTINE
                *--------------------------------
9C16- AE 72 AA  EXIT      LDX B.ADDR    GET ADDRESS OF BINARY PROGRAM
9C19- AC 73 AA            LDY B.ADDR+1  AND GOTO THE NORMAL LOAD
9C1C- 4C 71 A4            JMP END.LD    ROUTINE IN DOS
```

continued on facing page

Requirements:
48K Apple II with DOS 3.3

To enter HYPER-DOS, boot up a 3.3 DOS disk and do the following:

1. Type **CALL-151** return
2. Type **9DØØ:C9 9B** return (address of highest DOS buffer)
3. Type **9DØD:9C** return (high byte of bottom of DOS used by the INIT command)
4. Type **A38C:1F 9C** return
5. Type **A448:Ø1 9C** return
6. Type **3D3G** return (rebuilds buffers)
7. Type CALL-151 return
8. Enter the following HEX code:

```
9CØØ:ØØ A9 Ø2 8D F9 B7 A9 FD
9CØ8:8D FA B7 A5 67 85 FD A5
9C1Ø:68 85 FE 4C 41 9C AE 72
9C18:AA AC 73 AA 4C 71 A4 A9
9C2Ø:Ø4 8D F9 B7 A9 FB 8D FA
9C28:B7 AD 72 AA 85 FD AD 73
9C3Ø:AA 85 FE AD 61 AA C9 Ø1
9C38:BØ Ø7 AD 6Ø AA C9 FC 9Ø
9C4Ø:D5 18 AD CB B5 6D F9 B7
9C48:85 FB AD CC B5 69 ØØ 85
9C5Ø:FC AC FA B7 B1 FB 91 FD
9C58:88 DØ F9 B1 FB 91 FD 38
9C6Ø:A5 FD ED F9 B7 8D FØ B7
9C68:A5 FE E9 ØØ 8D F1 B7 18
9C7Ø:AD C9 B5 85 FB AD CA B5
9C78:85 FC AØ ØE 84 FF EE F1
9C8Ø:B7 B1 FB 8D EC B7 C8 B1
9C88:FB 8D ED B7 C8 84 FF B1
9C9Ø:FB C8 11 FB FØ 3E 2Ø E9
9C98:9C A4 FF DØ E1 C8 B1 FB
9CAØ:DØ Ø3 4C EA A2 8D EC B7
9CA8:C8 B1 FB 8D ED B7 AD FØ
9CBØ:B7 48 AD F1 B7 48 A5 FB
9CB8:8D FØ B7 A5 FC 8D F1 B7
9CCØ:2Ø E9 9C 68 8D F1 B7 68
9CC8:8D FØ B7 BØ 2A AØ ØC 84
9CDØ:FF B8 5Ø AA AD 6Ø AA 18
9CD8:6D F9 B7 8D 3D BE 2Ø E9
9CEØ:9C A9 ØØ 8D 3D BE 4C EA
9CE8:A2 A9 ØØ 8D EB B7 2Ø E3
9CFØ:Ø3 2Ø B5 B7 BØ Ø1 6Ø 68
9CF8:68 A2 Ø8 4C Ø2 A7 ØØ ØØ
```

9. Type **3DØG** return
1Ø. Insert a blank disk
11. Type **INIT HELLO**

This disk will now contain HYPER-DOS and may be used to initialize new HYPER-DOS disks.

Some problems have been encountered when using HYPER-DOS with binary programs. Binary programs which load exactly below DOS or relocate themselves below DOS and assume that DOS ends at $9DØØ will overwrite the HYPER-DOS code.

For those who encounter this problem, a version of HYPER-DOS that will reside inside DOS and overwrite the INIT code will be presented. More on that next next time.

```
                  *-------------------------------
                  * LOAD A BINARY PROGRAM
                  *-------------------------------
9C1F- A9 Ø4       BLOAD     LDA #4        LENGTH AND START LOCATION BYTES
9C21- 8D F9 B7              STA OFFSET
9C24- A9 FB                 LDA #$FB      NUMBER OF BYTES TO MOVE
9C26- 8D FA B7              STA MOV.NUM
                  *-------------------------------
9C29- AD 72 AA              LDA B.ADDR    GET MOVE TO ADDRESS (LO-BYTE)
9C2C- 85 FD                 STA PTR2
9C2E- AD 73 AA              LDA B.ADDR+1  GET MOVE TO ADDRESS (HI-BYTE)
9C31- 85 FE                 STA PTR2+1
                  *-------------------------------
9C33- AD 61 AA              LDA F.LEN+1   GET HI-BYTE OF FILE LENGTH
9C36- C9 Ø1                 CMP #1        MORE THAN ONE SECTOR?
9C38- BØ Ø7                 BCS SCT.LD    YES, GOTO COMMON LOAD ROUTINE
9C3A- AD 6Ø AA              LDA F.LEN     GET LO-BYTE OF FILE LENGTH
9C3D- C9 FC                 CMP #$FC      MORE THAN 252 BYTES?
9C3F- 9Ø D5                 BCC EXIT      NO, SO LOAD IN NORMALLY
                  *-------------------------------
                  * BEGIN LOADING SECTORS
                  *-------------------------------
9C41- 18          SCT.LD    CLC           SET UP FOR ADDITION
9C42- AD CB B5              LDA DA.BUF    GET LO-BYTE OF DATA BUFFER ADDR
9C45- 6D F9 B7              ADC OFFSET    SKIP NON-PROGRAM BYTES
9C48- 85 FB                 STA PTR       FROM LOCATION FOR MOVE (LO-BYTE)
9C4A- AD CC B5              LDA DA.BUF+1  GET HI-BYTE OF DATA BUFFER ADDR
9C4D- 69 ØØ                 ADC #$ØØ      ADD, IN CASE CARRY IS SET
9C4F- 85 FC                 STA PTR+1     FROM LOCATION FOR MOVE (HI-BYTE)
                  *-------------------------------
                  * MOVE FIRST CHUNK INTO PLACE
                  *-------------------------------
9C51- AC FA B7              LDY MOV.NUM   NO. OF BYTES TO MOVE
9C54- B1 FB       LOOP1     LDA (PTR),Y   MOVE (FROM) LOCATION
9C56- 91 FD                 STA (PTR2),Y  MOVE (TOO) LOCATION
9C58- 88                    DEY           GET NEXT BYTE
9C59- DØ F9                 BNE LOOP1     DONE? NO, SO GET NEXT BYTE
9C5B- B1 FB                 LDA (PTR),Y   GET THE LAST BYTE (Ø)
9C5D- 91 FD                 STA (PTR2),Y  MOVE IT
                  *-------------------------------
                  * SET RW.BUF TO POINT TO CORRECT MEMORY LOCATION
                  *-------------------------------
9C5F- 38                    SEC           SET CARRY FOR SUBTRACTION
9C6Ø- A5 FD                 LDA PTR2      START ADDRESS OF PROGRAM
9C62- ED F9 B7              SBC OFFSET    PAGE OFFSET FROM 1ST SECTOR
9C65- 8D FØ B7              STA RW.BUF    SET WORKBUFFER PNTR TO THIS LOC
9C68- A5 FE                 LDA PTR2+1    GET HI-BYTE OF PROGRAM START
9C6A- E9 ØØ                 SBC #$ØØ      TAKE CARE OF CARRY OVERS
9C6C- 8D F1 B7              STA RW.BUF+1  STORE HI-BYTE OF WORKBUFFER
                  *-------------------------------
9C6F- 18                    CLC           CLEAR ERROR FLAG
9C7Ø- AD C9 B5              LDA TS.BUF    GET ADDRESS OF T/S LIST
9C73- 85 FB                 STA PTR
9C75- AD CA B5              LDA TS.BUF+1
9C78- 85 FC                 STA PTR+1
                  *-------------------------------
9C7A- AØ ØE                 LDY #$ØE      NEXT T/S TO GET
9C7C- 84 FF                 STY TEMP      SAVE POSITION INTO T/S LIST
9C7E- EE F1 B7    LOAD.IT   INC RW.BUF+1  INCREMENT THE MOVE TO BUFFER
9C81- B1 FB                 LDA (PTR),Y   GET TRACK OF NEXT PART OF FILE
9C83- 8D EC B7              STA RW.TRK    PUT IT IN RWTS
9C86- C8                    INY           GET SECTOR
9C87- B1 FB                 LDA (PTR),Y
9C89- 8D ED B7              STA RW.SCT    PUT IN RWTS
9C8C- C8                    INY
9C8D- 84 FF                 STY TEMP      SAVE YREG
9C8F- B1 FB                 LDA (PTR),Y   GET TRACK
```

continued on page 7Ø

K E Y S
TO THE KEY BOARD

A New Column Examining AppleSoft B.A.S.I.C.

Bev. R. Haight

part1 Normal Keyboard Entry

KEYS TO THE KEYBOARD is the beginning of a TUTORIAL on Applesoft Programming Techniques. In this installment, normal keyboard entry is explored.

There are three ways to retrieve data from the keyboard while in the deferred mode (inside a RUNning Applesoft program):

1. Use INPUT
2. Use GET
3. PEEK (-16384)

Of course, you could use non-keyboard retrieval via game controls, the DOS EXEC command, BLOADing binary data, etc. But those are topics for other studies. I will examine only the former three means: INPUT, GET, and PEEK (-16384).

1. INPUTING

The INPUT statement is usually the first method we learn to use. It permits us to input a name, phrase, and a whole set of individual or grouped keyboard data (when followed by a press of the Return key). And the response is printed to the screen so that the input can be checked and corrected before finally being entered with a press of Return.

It could be something as simple as:

```
10 INPUT A$
20 INPUT A
30 INPUT A%
```

Or something as complex as:

```
40 INPUT A$, B$, C, D%, E$, F$, G, H
```

And it allows us to use our own "prompting statement":

```
50 INPUT "WHAT IS YOUR NAME, AGE, BIRTHDAY, HEIGHT, AND WEIGHT?";
NAME$, AGE, BIRTHDAY, HEIGHT, WEIGHT
```

It is the ability to easily accept complicated strings as well as serial data (and its re-prompting characteristic) that makes INPUT so flexible.

For example:

```
60 INPUT "PICK A NUMBER"; A
```

will prompt you with:

```
PICK A NUMBER
```

And if you should answer with any letter (except E), it will simply give you an error message of "?REENTER" followed by the same prompting string: PICK A NUMBER.

You can use:

```
70 INPUT "prompting statement"; A
```

(which demands a number, and may include exponents, decimal fractions, positive or negative...)

```
80 INPUT "prompting statement"; A%
```

(which demands only integers...)

```
90 INPUT "prompting statement"; A$
```

(which demands any string at all...)

However, using INPUT requires some care because:

1. Error messages can scroll the text page and might destroy the carefully formatted displays many programs require. This is easily prevented by controlling the dimensions of the text window while an INPUT is operating or by controlling the error messages with an ONERR GOTO statement.

2. Formatting can also be destroyed when the operator/user invokes the editing features (escape codes) and may lead to some confusion as to what is being asked by the INPUT statement. This is especially disconcerting when using multiple requests in a single INPUT statement. Using serial INPUTs in place of a multiple one can solve some of the problems.

3. The error messages are simple and do not really explain what was wrong with the prior response. These error messages consist of:

?REENTER or ?EXTRA IGNORED (which will not stop the program)

or even the dreaded: BREAK IN LINE # (which just stopped the program).

Table 1, "INPUT" shows what happens when you respond to that prompt in various ways. It shows that PRINTing the response can sometimes lead to undesirable results. If DOS is active, DOS commands preceded by ctrl D will be valid DOS commands when PRINTed and will be attempted, so beware of answering with:

ctrl D INIT!

Some things to remember about INPUT are:

1. A question mark is used as a "prompting" marker if you do not specify your own prompting "string" or statement. Only one string is allowed and it must be followed by a semicolon (;) and the variable(s).

2. Be sure to ONERR GOTO a line that RESUMEs before going back to the INPUT or else the 86th error may crash your program.

3. DO NOT USE ctrl X, ctrl M (Return) or a quotation mark (") inside a prompting string.

4. DO NOT USE ctrl X, ctrl M (Return), comma (,) or colon (:) inside a response to:

```
100 INPUT "prompting string"; A$
```

A ctrl M (or Return) is interpreted as "end of string input" even when nothing has been entered (in which case a "null" or empty string is assumed).

5. If the INPUT statement is not fully answered (as in a multiple INPUT such as:

```
110 INPUT MOS$,DAY%,YEAR
```

then double question marks are printed to prompt you to complete the answer.

6. Ctrl C can stop an INPUT if it is the first character typed. To continue, you must then reRUN the program. So beware. Ctrl C will NOT stop a program if it is preceded by any other response. Nor will ctrl C stop a multiple INPUT statement when it is entered as the first character to any but the first variable response.

Of course, the type of variable will affect the nature of the response accepted. This is clearly seen in table 1, "INPUT."

2. GETING

The second method, most often used for single-key selections because it doesn't require the pressing of Return, is the GET statement.

——— A word of warning ———

GET was meant to retrieve a string variable, that is, a letter of the alphabet. It was not meant to GET arithmetic variables. If you GET A or GET A%, and a non-numeric key is pressed, the program may crash with a resounding "beep!" and a ?SYNTAX ERROR. And if you try to bypass the error with an ONERR GOTO, the program may crash after the 43rd error. Using ONERR GOTO and then RESUME isn't much help either. It will crash after the second error. GETting a colon or comma will lead to ?EXTRA IGNORED. GETting a Plus or Minus sign, Space, Period, E, and ctrl "at" sign all are valid but equal to zero. A Return is like any non-numeric response and ends with ?SYNTAX ERROR.

DON'T GET A

If you must GET a number, first GET A$, then change the string into a number using A = VAL (A$).

GETting a string can also result in a few odd responses. A ctrl H (left arrow key), ctrl "at" and ctrl M (Return) all equal a null character (empty string). Ctrl C at this time does not stop the program.

DO GET A$

Use table 2, "GET A$," to determine what happens when various keys are pressed. The table shows the key pressed in the right-most column. Following it are: ALONE or when only that key is pressed, when the SHIFT key is also used, when the CONTROL key is also pressed, and finally when BOTH the shift and control keys are pressed. Each column shows first what you get when you PRINT A$ followed by the ASCii value of the keys' combinations (what you get when you PRINT ASC(A$).

The two columns on the right shows what key corresponds to a particular ASCii value. Values 1 through 3Ø are control characters and not printed to the screen. Some control letters, when PRINTED, result in certain operations (see the notes in table 2).

Some characters cannot be directly gotten from the keyboard. These include the underscore (95), left bracket (91) and the back slash (92).

And trying to PRINT ASC (A$) of a shift ctrl "at" will stop the program with ?ILLEGAL QUANTITY ERROR IN line number #.

In order to use GET to get an entire string, it is necessary to use concan-

INPUT

USER RESPONSE	PROGRAM RESPONSES				
	INPUT A or A%	PRINT A	PRINT A%	INPUT A$	PRINT A$
Ø	-ok-	Ø	Ø	-ok-	Ø
ØØØØØØØØØØ	-ok-	Ø	Ø	-ok-	ØØØØØØØØØØ
123456789	-ok-	123456789	<see note A>	-ok-	123456789
1111111111	-ok-	1.111111111E+Ø9	<see note A>	-ok-	1111111111
11223344556677	-ok-	1.12233446E+17	<see note A>	-ok-	11223344556677
+	-ok-	Ø	Ø	-ok-	+
-	-ok-	Ø	Ø	-ok-	-
.	-ok-	Ø	Ø	-ok-	.
:	?EXTRA IGNORED	Ø	Ø	-ok-	:
,	?EXTRA IGNORED	Ø	Ø	-ok-	,
;	?REENTER	<nothing>	<nothing>	-ok-	;
^]<>?/	?REENTER	<nothing>	<nothing>	-ok-	^]<>?/
#=)('&%$#"!	?REENTER	<nothing>	<nothing>	-ok-	#=)('&%$#"!
-1	-ok-	-1	-1	-ok-	-1
-1-1	?REENTER	<nothing>	<nothing>	-ok-	-1-1
+9	-ok-	9	9	-ok-	+9
+9+9	?REENTER	<nothing>	<nothing>	-ok-	+9+9
1+1	?REENTER	<nothing>	<nothing>	-ok-	1+2
E	-ok-	Ø	Ø	-ok-	E
+E	-ok-	Ø	Ø	-ok-	+E
-E	-ok-	Ø	Ø	-ok-	-E
-E7	-ok-	Ø	Ø	-ok-	-E7
6E+1Ø	-ok-	6E+1Ø	<see note A>	-ok-	6E+1Ø
6E+4	-ok-	6ØØØØ	<see note A>	-ok-	6E+4
6E+2	-ok-	6ØØ	6ØØ	-ok-	6E+2
6E-4	-ok-	6E+Ø4	<see note A>	-ok-	6E-4
6E-2	-ok-	.Ø6	Ø	-ok-	6E-2
,2	?EXTRA IGNORED	Ø	Ø	-ok-	,2
1,2	?EXTRA IGNORED	1	1	-ok-	1,2
1:2	?EXTRA IGNORED	1	1	-ok-	1:2
1 1 1 1	-ok-	1111	1111	-ok-	1 1 1 1
.999999999	-ok-	.999999999	Ø	-ok-	.999999999
.9999999999	-ok-	1	1	-ok-	.9999999999
.9.9	?REENTER	<nothing>	<nothing>	-ok-	.9.9
->->->5	-ok-	5	5	-ok-	5
ABCDEFGHIJ	?REENTER	<nothing>	<nothing>	-ok-	ABCDEFGHIJ
1A	?REENTER	<nothing>	<nothing>	-ok-	1A
A1	?REENTER	<nothing>	<nothing>	-ok-	A1
ctrl G	?REENTER	<nothing>	<nothing>	-ok-	<bell>
ctrl J	?REENTER	<nothing>	<nothing>	-ok-	<line feed>
ctrl C	<see note C>	<nothing>	<nothing>	<see note C>	<nothing>
ctrl S ctrl C	?REENTER	<nothing>	<nothing>	?REENTER	<nothing>
ctrl X	\	<nothing>	<nothing>	\	<nothing>
ctrl M (Return)	?REENTER	<nothing>	<nothing>	?REENTER	<nothing>
ctrl D ctrl D	?REENTER	<nothing>	<nothing>	-ok-	<see note B>
ctrl D (note D)	?REENTER	<nothing>	<nothing>	-ok-	<see note D>

A. program stops with ?ILLEGAL QUANTITY ERROR IN line number.
B. program stops with ?SYNTAX ERROR   BREAK IN line number.
C. program stops with ?BREAK IN line number.
D. DOS commands are valid if preceeded by ctrl D.
-ok- means that it is a valid entry and was acted upon.
<nothing> means that nothing was printed

tenation. For example, a simple (but error-prone) loop would be:

```
============== GETTER 1 ===============

9    REM RETRIEVE ONE CHARACTER

1Ø   GET A$

19   REM IF KEY PRESSED IS RIGHT ARROW
     AND IF B$ IS MORE THAN A SINGLE
     CHARACTER THEN DELETE THE LAST
     CHARACTER

2Ø   IF A$ = CHR$ (8) AND LEN (B$) > 1
     THEN B$ = LEFT$ (B$, LEN (B$) - 1) :
     PRINT A$" "A$ ;:

29   REM IF KEY PRESSED IS RIGHT ARROW OR
     A RETURN THEN NEW B$

3Ø   IF A$ = CHR$ (8) OR A$ = CHR$ (13)
     THEN PRINT A$" " : B$ = "" : GOTO 1Ø

39   REM GET RID OF THE REST OF THE CTRL
     CHARACTERS

4Ø   IF ASC (A$) < 32 THEN A$ = ""

49   REM ADD CHARACTER TO B$, PRINT THAT
     CHARACTER

5Ø   B$ = B$ + A$ : PRINT A$ ;

59   REM CONTINUE WITH STRING RETRIEVAL

6Ø   GOTO 1Ø

=======================================
```

This routine will not permit control characters into B$ and it traps out Returns. The left arrow can still be used to remove characters from B$.

As we have seen, both INPUT and GET will temporarily halt program execution while waiting for something from the keyboard. INPUT requires a Return, GET does not.

3. PEEKING

If we want the program to continue while waiting for something from the keyboard, we should simply PEEK (-16384) or (49152). If the value returned is greater than 127 (the high-bit is set) then a key has been pressed, in which case you should immediately clear the keyboard strobe by a PEEK (-16368) OR (49168). You can also POKE anything there because all that you need to do to "clear" the keyboard is to make a reference to that location (-16368) and the value at -16384 will drop by 128 (the high-bit is no longer set).

Table 3, "PEEK -16384," is similar in format to table 2 except that it shows values at that location in memory followed by what happens when you PRINT CHR$(A) where A = PEEK (-16384) and IF A > 127. Values of 128 through 159 are control characters. Again, the left bracket, back slash and the underscore cannot be directly keyed. And like GET, you must provide a loop in order to acquire a long string or statement.

GET A$

KEY	ALONE A$	ALONE ASC(A$)	SHIFT A$	SHIFT ASC(A$)	CONTROL A$	CONTROL ASC(A$)	BOTH A$	BOTH ASC(A$)
1 !	1	49	!	33	1	49	!	33
2 "	2	5Ø	"	34	2	5Ø	"	34
3 #	3	51	#	35	3	51	#	35
4 $	4	52	$	36	4	52	$	36
5 %	5	53	%	37	5	53	%	37
6 &	6	54	&	38	6	54	&	38
7 '	7	55	'	39	7	55	'	39
8 (	8	56	(	4Ø	8	56	(	4Ø
9)	9	57	)	41	9	57	)	41
Ø	........48........							
: *	:	58	*	42	:	58	*	42
- =	-	45	=	61	-	45	=	61
; +	;	59	+	43	;	59	+	43
, <	,	44	<	6Ø	,	44	<	8
. >	.	46	>	62	.	46	>	62
/ ?	/	47	?	63	/	47	?	63
<--	........8........							
-->	........21........							
SPACE	........32........							
RETURN	........13........							
ESC	........27........							
A	A	65	A	65	><	1	><	1
B	B	66	B	66	><	2	><	2
C	C	67	C	67	><	3	><	3
D	D	68	D	68	><	4	><	4
E	E	69	E	69	><	5	><	5
F	F	7Ø	F	7Ø	><	6	><	6
G	G	71	G	71	>bell<	7	>bell<	7
H	H	72	H	72	see a	8	see a	8
I	I	73	I	73	><	9	><	9
J	J	74	J	74	see b	1Ø	see b	1Ø
K	K	75	K	75	><	11	><	11
L	L	76	L	76	><	12	><	12
M]	M	77	]	93	see c	13	see c	13
N ^	N	78	^	94	><	14	><	14
O	O	79	O	79	><	15	><	15
P @	P	8Ø	@	64	><	16	see d	16
Q	Q	81	Q	81	><	17	><	17
R	R	82	R	82	><	18	><	18
S	S	83	S	83	><	19	><	19
T	T	84	T	84	><	2Ø	><	2Ø
U	U	85	U	85	see e	21	see e	21
V	V	86	V	86	><	22	><	22
W	W	87	W	87	><	23	><	22
X	X	88	X	88	><	24	><	24
Y	Y	89	Y	89	><	25	><	25
Z	Z	9Ø	Z	9Ø	><	26	><	26

ASC	A$		
1	>A<	48	Ø
2	>B<	49	1
3	>C<	5Ø	2
4	>D<	51	3
5	>E<	52	4
6	>F<	53	5
7	>G<	54	6
8	see a	55	7
9	>I<	56	8
1Ø	see b	57	9
11	>K<	58	:
12	>L<	59	;
13	see c	6Ø	<
14	>N<	61	=
15	>O<	62	>
16	>P<	63	?
17	>Q<	64	@
18	>R<	65	A
19	>S<	66	B
2Ø	>T<	67	C
21	see d	68	D
22	>V<	69	E
23	>W<	7Ø	F
24	>X<	71	G
25	>Y<	72	H
26	>Z<	73	I
27	ESCape	74	J
28	***	75	K
29	>]<	76	L
3Ø	>^<	77	M
31	***	78	N
32	SPACE	79	O
33	!	8Ø	P
34	"	81	Q
35	#	82	R
36	$	83	S
37	%	84	T
38	&	85	U
39	'	86	V
4Ø	(	87	W
41	)	88	X
42	*	89	Y
43	+	9Ø	Z
44	,	91	***
45	-	92	***
46	.	93	]
47	/	94	^
		95	***

```
    >< indicates that, when PRINTed to the screen, nothing gets
printed, not even a space.
    a. CTRL H and Shift CTRL H are the same as the backspace key (<--).
    b. CTRL J and Shift CTRL J are line feeds (moves down one line).
    c. CTRL M is the same as a RETURN (goes to beginning of next line).
    d. Trying to get the ASC() of a Shift CTRL P will end up in an
ERROR.
    e. CTRL U and Shift CTRL U are the same as the --> key.
    GETting a CTRL C will not end the program.
```

KEY PEEKER 1 is just such a subroutine:

```
*********** KEY PEEKER 1 ***************

1    ONERR GOTO 1020
10   B$ = "" : G$ = CHR$ (7) : Z$ = CHR$
     (136) : S$ = CHR$ (32) : S = -16336
     : GOTO 1000
```

```
********** PEEKER SUBROUTINE ***********

100  A = PEEK (-16384) ) : IF A < 128
     THEN RETURN
110  POKE - 16368, 0 : A$ = CHR$ (A)
120  IF A = 136 AND LEN (B$) > 1 THEN B$
     = LEFT$ (B$, LEN (B$) - 1) : PRINT
     Z$S$S$Z$Z$ ; : GOSUB 200 : RETURN
```

```
130  IF A = 136 THEN PRINT Z$S$S$ :
     GOSUB 210 : B$ = "" : RETURN
140  IF A = 141 THEN PRINT S$A$ : GOSUB
     210 : B$ = "" : RETURN
150  IF A < 160 THEN RETURN
160  B$ = B$ + A$ : PRINT A$ ;
190  GOSUB 220 : RETURN
```

```
************** TAP TAP ! ***************

200  FOR A = 1 TO 10 : N = PEEK (S) -
     PEEK (S) : NEXT : RETURN
210  PRINT G$;
220  N = PEEK (S) + PEEK (S) + PEEK (S)
     + PEEK (S) : RETURN
```

```
**********SHIMMERING CURSOR ***********

1000 HOME
1010 VTAB 10 : HTAB 1 : CALL -868
1020 PRINT "-"Z$ ; : GOSUB 100
1030 PRINT "<"Z$ ; : GOSUB 100
1040 PRINT S$Z$ ; : GOSUB 100
1050 GOTO 1020
```

PEEK -16384

	ALONE		SHIFT		CONTROL		BOTH	
KEY	VALUE	CHR$	VALUE	CHR$	VALUE	CHR$	VALUE	CHR$
1 !	177	1	161	!	177	1	161	!
2 "	178	2	162	"	178	2	162	"
3 #	179	3	163	#	179	3	163	#
4 $	180	4	164	$	180	4	164	$
5 %	181	5	165	%	181	5	165	%
6 &	182	6	166	&	182	6	166	&
7 '	183	7	167	'	183	7	167	'
8 (	184	8	168	(	184	8	168	(
9)	185	9	169	)	185	9	169	)
0	......176......							
: *	186	:	170	*	186	:	170	*
- =	173	-	189	=	173	-	189	=
; +	187	;	171	+	187	;	171	+
, <	172	,	188	<	172	,	188	<
. >	174	.	190	>	174	.	190	>
/ ?	175	/	191	?	175	/	191	?
<--	......136......							
-->	......149......							
SPACE	.......160......							
RETURN	......141......							
ESC	.........155......							
A	193	A	193	A	129	>A<	129	>A<
B	194	B	194	B	130	>B<	130	>B<
C	195	C	195	C	131	>C<	131	>C<
D	196	D	196	D	132	>D<	132	>D<
E	197	E	197	E	133	>E<	133	>E<
F	198	F	198	F	134	>F<	134	>F<
G	199	G	199	G	135	>G<	135	>G<
H	200	H	200	H	136	>H<	136	>H<
I	201	I	201	I	137	>I<	137	>I<
J	202	J	202	J	138	>J<	138	>J<
K	203	K	203	K	139	>K<	139	>K<
L	204	L	204	L	140	>L<	140	>L<
M]	205	M	221	]	141	>M<	157	>]<
N ^	206	N	222	^	142	>N<	158	>^<
O	207	O	207	0	143	>O<	143	>O<
P @	208	P	192	@	144	>P<	128	>@<
Q	209	Q	209	Q	145	>Q<	145	>Q<
R	210	R	210	R	146	>R<	146	>R<
S	211	S	211	S	147	>S<	147	>S<
T	212	T	212	T	148	>T<	148	>T<
U	213	U	213	U	149	>U<	149	>U<
V	214	V	214	V	150	>V<	150	>V<
W	215	W	215	W	151	>W<	151	>W<
X	216	X	216	X	152	>X<	152	>X<
Y	217	Y	217	Y	153	>Y<	153	>Y<
Z	218	Z	218	Z	154	>Z<	154	>Z<

cannot be directly keyed:

219	left bracket
220, 156	back slash
223, 159	underscore

VALUE	CHR$		
128	>@<	173	-
129	>A<	174	.
130	>B<	175	/
131	>C<	176	0
132	>D<	177	1
133	>E<	178	2
134	>F<	179	3
135	>G<	180	4
	bell	181	5
136	>H<	182	6
	<----	183	7
137	>I<	184	8
138	>J<	185	9
	line feed	186	:
139	>K<	197	;
140	>L<	188	<
141	>M<	189	=
	RETURN	190	>
142	>N<	191	?
143	>O<	192	@
144	>P<	193	A
145	>Q<	194	B
146	>R<	195	C
147	>S<	196	D
148	>T<	197	E
149	>U<	198	F
	---->	199	G
150	>V<	200	H
151	>W<	201	I
152	>X<	202	J
153	>Y<	203	K
154	>Z<	204	L
155	ESCAPE	205	M
156	> <	206	N
157	>]<	207	O
158	>^<	208	P
159	> <	209	Q
160	SPACE	210	R
161	!	211	S
162	"	212	T
163	#	213	U
164	$	214	V
165	%	215	W
166	&	216	X
167	'	217	Y
168	(	218	Z
169	)	219	***
170	*	220	***
171	+	221	]
172	,	222	^
		223	***

This shows how program execution can continue while periodically checking to see if a key has been pressed. In this case, the time between each keypress is used to animate a "shimmering cursor" consisting of intermittently PRINTing a "less than" sign and a dash (lines 1020-1040). This version accepts only non-control characters. Return (line 140) is used to scroll to the next line, and the left arrow key is used to back up over mistakes (line 120), and backing up over all the entries is the same as a Return (line 130).

An optional "key tapping sound" has been included (lines 200-220) with an extra beep for a Return.

non-scrolling key PEEKer

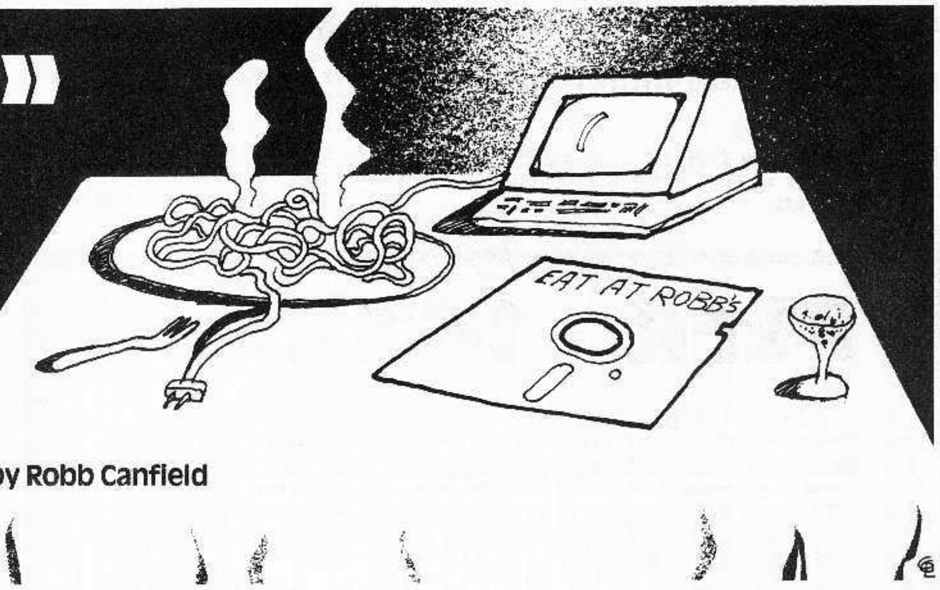
If you don't want scrolling to occur, make these changes in the program:

```
************ KEY PEEKER 2 *************

DEL 130, 140

130  IF A = 136 OR A = 141 THEN GOSUB
     210 : B$ = "" : POP : GOTO 1010

=======================================
```

Don't give in to the temptation to speed up the PEEK routine by making K = -16384 so that you can simple and quickly PEEK (K). If you do, you will lose the "shimmering" cursor effect.

There you have it: part one of KEYS TO THE KEYBOARD.

"MENU" Hello Program



by Robb Canfield

Requirements:
APPLE II with 32K or 48K
DOS 3.2, 3.2.1 or 3.3
Applesoft in ROM

MENU HELLO is a user-oriented program easily modified for individual needs. A menu Hello program is one that makes "turn-key" operation possible by providing a quick, simple and user-friendly way to LOAD, RUN, BLOAD or BRUN programs on a disk.

Most users have probably written a menu program of some sort, such as:

```
10 PRINT "1) DISKVIEW 1.0"
20 PRINT "2) DISKEDIT 2.5"
30 PRINT: INPUT "RUN "; A$
40 PRINT CHR$(4) "RUN" A$
```

This is great for a few programs on selected diskettes, but writing a new menu program for each disk can be tiring at best, and probably not worth it anyway. MENU is a more advanced type of menu program that allows the user to do a multitude of things with the directory (where DOS stores all the information printed when a CATALOG is done), from LOADing a program to LOCKing or UNLOCKing some or all of the programs on a disk.

Type the listing for MENU in the order explained in "How To Enter MENU." SAVE the program. If MENU is used for the "Hello" program, be sure that it is the one that RUNs first when the disk is booted. Do this by SAVEing MENU under whatever file name that is RUN when the disk boots, or INITialize new disks so that MENU is its "HELLO" program. This can be done by simply typing INIT MENU, followed by a V and the volume number desired (1 - 254).

Now RUN it. "READING CATALOG" will be visible in the center of the screen. At this time MENU is reading the catalog. After a few seconds the first part (page) of the catalog will be seen.

To "page" thru it (when there are a lot of files in the catalog), use the left and right arrow keys.

To select a program, just type its letter code. This will cause the MINI-MENU to be entered. The MINI-MENU is self-prompting (it asks all the questions) and simple to use.

The only confusing part that may arise is when a binary file is RUN or LOADed. The MINI-MENU will ask for the RUNning or LOADing location. This is an optional choice. To LOAD or RUN the binary file at other than the original address, enter the new address. Press return to use the original address.

REMEMBER: Always precede a hex location with a "$".

It is possible to DELETE, RENAME, UNLOCK, LOCK, LOAD (BLOAD), RUN (BRUN) a program and EXEC a file (depending on its type) using the MINI-MENU. The MINI-MENU also automatically updates the options for the user. This is done so that a locked file is not locked again, and a text file isn't RUN or LOADed. There is even an UNLOCK/LOCK ALL mode that will LOCK or UNLOCK every file on a disk. Whenever a file is selected, all valid commands are displayed along with their explanations. All commands are NORMAL keys. No control characters are used in the MINI-MENU.

WARNING:

MENU has a problem reading files that are in inverse or flashing. MENU will print garbage for the file name and will generate errors if anything is done to that file by the MINI-MENU. MENU will only work properly on a normal catalog (one that prints only the unmodified catalog header and the file names in a normal fashion). A catalog that prints either the amount of space left or that the disk is okay will not be read properly by MENU.

When a CATALOG is done, the file names and their status codes are displayed. This is a typical example:

***A 009 CALCULATING PI**

And this is an explanation of the various parts of the file status code preceding the actual file's name (CALCULATING PI):

***** - The asterisk means the file is locked. If the file wasn't locked this would be a space instead of an asterisk.

A - The "A" tells DOS that the program is in Applesoft. "I" means Integer, "B" means Binary, and "T" means a text file.

009 - These three numbers represent the length of the program or text file in sectors. This number will always be three digits.

The status section of the file takes up seven characters. The file name takes up thirty characters (DOS always reserves thirty characters for the file name. All characters after the actual file name are spaces and printed as such). DOS also prints a carriage return after it finishes printing the file name, so that the following file name is put on the next line down.

The machine language portion of MENU fools DOS into putting the file name and its status into the string array (NA$), instead of on the screen. Everything that is normally printed on the screen is put into the array instead, including the catalog header. The information can then be used as desired.

Line Ø: How To Enter MENU

MENU consists of two parts:
1. A machine language listing.
2. A BASIC listing

The machine language routine must be entered from the monitor (*) before the BASIC listing is typed.

First of all, make sure that the Applesoft pointers are set correctly by typing:
FP Return

Then enter the monitor by typing:
CALL -151

Now type the following lines:

```
*800: 00 38 08 00 00 B2 A9 28  Return
 808: 9D 53 AA A9 08 8D 54 AA  Return
 810: A9 36 8D 55 AA A9 08 8D  Return
 818: 56 AA A0 08 B1 6B 8D 2C  Return
 820: 08 C8 B1 6B 8D 2D 08 60  Return
 828: 38 E9 80 8D FF FF EE 2C  Return
 830: 08 D0 03 EE 2D 08 60 00  Return
 838: 00 00  Return
```

While still in the monitor, set the "End Of Program" counter to point to location $83A by typing:
AF: 3A 08 Return

Get back into Applesoft by typing:
3DØG Return

And then type:
LIST Return

There should be a line Ø that looks EXACTLY like:

```
Ø  REM  SPEED= ( PLOT S LET  SPEED=
     PLOT T LET  SPEED= 6 PLOT U LET
      SPEED= PLOT V LET  COLOR= RETURN
     + PLOT  +  RETURN + PLOT  8 RIGHT$
 END  PLOT  REM  INPUT  OUT OF DATA
     =  OUT OF DATA
```

Do NOT use a printer to LIST line Ø in order to compare it. The printer will probably LIST it quite differently from what appears on the screen.

If line Ø does not look like this, then there is an ERROR in the machine language subroutine and it must be re-entered (corrected).

If unfamiliar with the Monitor commands, the easiest way to make changes is to retype whatever line is in error. First, get back into the Monitor by typing:
CALL -151

The Monitor's asterisk (*) will be seen. Now type:
800.838

The figures on the screen are what is called a HEX DUMP. Compare it with the one shown in Figure 1 (the word "return" won't be seen after each line). If any line is not the same, re-enter the entire line followed, again, by RETURN. After the corrections are completed, exit to Applesoft by typing:
3DØG Return

(or, for those with the Autostart ROM, just press RESET.) Now type:
LIST Return

Again, compare the odd listing of line Ø to see if it is correct.

Line Ø should be the only line of the program at this time. The machine language subroutine is hidden behind the REM statement and will not be affected by RENUMBER or line changes as long as this is the FIRST line in the program.

All that is left is to type the BASIC listing as shown, and SAVE it to the disk. There is no need to type the other REMs in the BASIC listing, but it may help when modifying the program later.

CAUTION

Do NOT under any condition delete or modify line Ø. Any modification to line Ø will cause MENU to run incorrectly.

COMMANDS FOR MENU

(ctrl M or) RETURN =	**Read a new catalog into memory.**
(ctrl C) =	**CATALOG command. Doesn't update the catalog in memory.**
(ctrl H or) Left Arrow =	**Page backward thru the menu. Supports wrap around.**
(ctrl U or) Right Arrow =	**Page forward thru the menu. Supports wrap around.**
A through Z =	**Used to make single-key selections. It will enter the MINI-MENU.**
(ctrl X) =	**Exit the program.**

ESC = "Help Mode". Displays the commands in an abbreviated form.

```
NOTE: This listing has been edited to
make it easier to read and enter. To
do this, I have made some changes.
When entering the BASIC version, re-
member that spaces are only important
within a PRINT statement. This
listing also does not use semicolons
when they are not necessary.

Sometimes to avoid confusion, the
PRINT statement will look like this:

1 PRINT
 "SUPERCALIFRAGILISTICEXPIALIDOCIOUS"

Or when it contains normally spaced
words, it will look like this:

1ØØØ PRINT "THIS PRINT STATEMENT IS
     DIVIDED BETWEEN WORDS, SO JUST
     PUT A NORMAL SPACE BETWEEN THEM
     IN ORDER TO RECONSTRUCT THE
     STATEMENT.

Finally, please include the copyright
notice so that the original author is
given proper credit for the work
done.                           (BRH)
```

```
******* MENU LISTING *******************

Ø   ----see text----
1   REM   MENU II COPYRIGHT 1982
  BY HARDCORE COMPUTING
2   REM   WRITTEN BY ROBB CANFIELD
5   TEXT : HOME

>>>>>>> INITALIZE STRING STORAGE <<<<<<<

1Ø  DIM NA$(1Ø4)
2Ø  FOR X = 1 TO 4Ø: HE$ = HE$ + "=":
  NEXT X
3Ø  FOR X = 1Ø4 TO 1 STEP - 1: NA$(X) =
  LEFT$ (HE$, 38): NEXT
32  STZ = PEEK (984)
33  FOR X = 2Ø58 TO 2Ø73 STEP 5: POKE
  X, STZ: NEXT
4Ø  NA$(Ø) = LEFT$ (HE$, 19)
5Ø  D$ = CHR$ (4): G$ = CHR$ (7) +
  CHR$ (7)

>>>>>>>>> RE-READ  CATALOG <<<<<<<<<<<<
       -> HARD ENTRY POINT <-

55  TEXT : HOME
6Ø  VTAB 12: HTAB 12: PRINT "READING
  CATALOG ": VTAB 12: HTAB 29

>>>>>>> CALL MACHINE SUBROUTINE <<<<<<<<
         AND GET A CATALOG

19Ø  ONERR GOTO 135Ø
2ØØ  CALL 2Ø54: PRINT D$ "CATALOG"

>>>>>>>>>> GET MAXIMUM PAGE <<<<<<<<<<<

2Ø6  FOR X = 1 TO 3: IF MID$ (NA$ (X *
   26 + 1), 2, 1) = "=" THEN 21Ø
```

```
2Ø8  NEXT
21Ø  MA = X - 1: POKE 216, Ø: PRINT
   D$ "PR#Ø": PRINT D$ "IN#Ø": PRINT
22Ø  X = Ø

>>>>>>>>> PRINT  FILENAMES <<<<<<<<<<<<
      -> SOFT ENTRY POINT <-

23Ø  HOME
24Ø  VTAB 1: HTAB 1Ø: PRINT MID$ (NA$
   (Ø), 3, 15)
25Ø  VTAB 3: PRINT HE$: VTAB 19: PRINT
   HE$
27Ø  IF X < Ø THEN X = MA
28Ø  IF X > (MA) THEN X = Ø
29Ø  POKE 34, 4: POKE 35, 18
3ØØ  VTAB 5: HTAB 1: PRINT
31Ø  HOME

>>>>>>>>>> PRINT FILENAMES ON <<<<<<<<<<
         LEFT SIDE OF SCREEN

32Ø  FOR Y = 1 TO 13
322  A$ = MID$ (NA$ (26 * X + Y), 8, 17)
325  IF MID$ (A$, 2, 1) = "=" THEN Y =
   Y - 1: GOTO 44Ø
33Ø  A$ = CHR$ (64 + Y) + " " + A$
35Ø  PRINT A$
36Ø  NEXT
37Ø  VTAB 5

>>>>>>>>>> PRINT FILENAMES ON <<<<<<<<<<
         RIGHT SIDE OF SCREEN

38Ø  FOR Y = 1 TO 13
382  A$ = MID$ (NA$ (26 * X + Y + 13),
   8, 17)
385  IF  MID$ (A$, 2, 1) = "=" THEN MA
   = X: Y = Y + 12: GOTO 44Ø
39Ø  A$ = CHR$ (65 + Y + 12) + " " + A$
41Ø  HTAB 21: PRINT A$
42Ø  NEXT

>>>>>>> CHECK FOR LEGAL SELECTION <<<<<<

43Ø  Y = 26

>>> PRINT PAGE #, GET FILE SELECTION <<<

44Ø  POKE 34, 2Ø: POKE 35, 24
45Ø  VTAB 22: HTAB 3Ø: PRINT "PAGE "
   X + 1 " OF " MA + 1;
46Ø  VTAB 24: HTAB 1Ø: INVERSE : PRINT
   "PRESS 'ESC' FOR HELP";: NORMAL
47Ø  VTAB 21: HTAB 1: PRINT
   "SELECT ONE (PRESS A KEY ) ";: GET
   A$: PRINT
475  IF A$ = CHR$ (8) AND MA < > Ø THEN
   X = X - 1: GOTO 27Ø
48Ø  IF A$ = CHR$ (21) AND MA < > Ø THEN
   X = X + 1: GOTO 27Ø
49Ø  IF A$ = CHR$ (24) THEN 13ØØ
5ØØ  IF A$ = CHR$ (27) THEN 3Ø1Ø
51Ø  IF A$ = CHR$ (13) THEN 55
52Ø  IF A$ = CHR$ (3) THEN TEXT : HOME :
   PRINT D$"CATALOG": GET B$: PRINT :
   PRINT : HTAB 8: PRINT
   "HIT ANY KEY TO CONTINUE ";: GET
   B$: GOTO 23Ø
53Ø  IF A$ > CHR$ (Y + 64) THEN 47Ø
54Ø  IF A$ < "A" OR A$ > "Z" THEN 47Ø
55Ø  A = ASC (A$) - 64

>>>>> INITALIZE MINI-MENU DEFAULTS <<<<<
        AND PRINT OUT CHOICES

56Ø  TEXT : HOME : VTAB 2: HTAB 16:
   PRINT "MINI-MENU"
```

How MENU works: A

This is a line-by-line description of Menu. The subtitles are the same as the ones used in the listing itself. Line numbers precede their explanations.

- LINE Ø: MACHINE LANGUAGE PROGRAM -

The machine language part of Menu is hidden within a REM statement on line Ø. Line Ø is typed with the machine subroutine (see "How to enter MENU").

- INITIALIZE STRING STORAGE (1Ø-5Ø) -

1Ø: First, reserve memory for 1Ø4 file names and the catalog header (NA$).

2Ø: Each element of NA$ must be set to 38 characters. This is done by creating a string (HE$) consisting of 4Ø "='s and running through a loop that sets each element of NA$ to the first thirty-eight characters of HE$ (using LEFT$), except for NA$(Ø) which is the catalog header and, as such, only needs nineteen characters.

3Ø: The loop runs backward so that the NA$(Ø) is the first element filled when a CATALOG is done.

32-33: Get the amount of memory available in the machine (48K or 32K) by checking location 984 (dec). This value is then POKEd into the machine subroutine hidden in line Ø.

4Ø: Set enough room aside for the catalog header.

- REREAD CATALOG (55-22Ø) - (HARD ENTRY POINT)

55-2ØØ: CALL the machine subroutine and read the catalog into memory. This is the HARD re-entry point. Going to this line will cause all information about the current catalog to be deleted and a new one read.

2Ø6-22Ø: Find the last page of the catalog. This page minus one is stored in the variable MA.

- PRINT FILE NAMES (23Ø-31Ø) - (SOFT ENTRY POINT)

23Ø, 24Ø: Clear window and print the catalog header.

25Ø: Print the boundary (HE$).

27Ø: Control the wrap-around feature.

32Ø-37Ø: Look for a blank file name

```
565  VTAB 7: HTAB 3: A$ = NA$- (X * 26
   + A): PRINT A$
57Ø  L = Ø: B = Ø: L$ = "K) LOCK": B$ =
   "L) LOAD": B1$ = "R) RUN"
575  IF LEFT$ (A$,1) = "*" THEN L = 1:
   L$ = "U) UNLOCK"
58Ø  IF MID$ (A$, 2, 1) = "B" THEN B =
   1: B$ = "L) BLOAD": B1$ = "R) BRUN"
585  IF MID$ (A$, 2, 1) = "T" THEN B =
   2: B$ = "E) EXEC"
59Ø  PRINT L$: PRINT B$: IF B < > 2 THEN
   PRINT B1$
595  PRINT "D) DELETE": PRINT
   "C) CHANGE PROGRAM NAME"
6ØØ  PRINT "A) LOCK/UNLOCK ALL"
61Ø  PRINT "X) EXIT TO CATALOG"
615  ONERR GOTO 137Ø
62Ø  B$ = "": FOR Y = 37 TO 8 STEP - 1:
   IF MID$ (A$, Y, 1) = " " THEN  NEXT
63Ø  A$ = MID$ (A$, 8, Y - 7)
64Ø  PRINT
65Ø  PRINT "ENTER YOUR CHOICE > ";: GET
   B$: PRINT : PRINT

>>>>>>>>>>>>>> LOCK FILE <<<<<<<<<<<<<<

66Ø  IF B$ < > "K" OR L < > Ø THEN 71Ø:
   REM  LOCK FILE ONLY IF UNLOCKED NOW
67Ø  PRINT "LOCK "A$;: GOSUB 3ØØØ
68Ø  IF B$ < > "Y" THEN 56Ø
69Ø  HOME : VTAB 12
692  Y = 4Ø - LEN (A$) - LEN (B$) - 8:
   IF Y < 2 THEN Y = 2
694  HTAB (Y / 2)
699  PRINT "LOCKING "A$: PRINT D$ "LOCK"
   A$
7ØØ  NA$(X * 26 + A) = "*" +  RIGHT$
   (NA$ (X * 26 + A), 37): GOTO 23Ø

>>>>>>>>>>>>> UNLOCK FILE  <<<<<<<<<<<<<

71Ø  IF B$ < > "U" OR L < > 1 THEN 76Ø:
   REM  UNLOCK FILE ONLY IF NOW LOCKED
72Ø  PRINT "UNLOCK "A$;: GOSUB 3ØØØ
73Ø  IF B$ < > "Y" THEN 56Ø
74Ø  HOME : VTAB 12: Y = 4Ø - LEN (A$)
   - 11: IF Y < 2 THEN Y = 2
745  HTAB Y / 2: PRINT "UNLOCKING " A$:
   PRINT D$ "UNLOCK" A$
75Ø  NA$ (X * 26 + A) = " " + RIGHT$
   (NA$ (X * 26 + A), 37): GOTO 23Ø
76Ø  IF B = 2 THEN 94Ø

>>>>>>>>>>>> RUN/BRUN FILE <<<<<<<<<<<<

765  IF B$ < > "R" THEN 9ØØ
77Ø  L$ = "": B$ = "": C$ = "BRUN"
8ØØ  IF B = Ø THEN C$ = "RUN": GOTO 83Ø

>>>>>>>>>> IF FILE IS BINARY, <<<<<<<<<<
        GET OPTIONAL LOAD ADDRESS

81Ø  PRINT : PRINT "WHERE TO "C$;: INPUT
   " (DEC $HEX ) > "; L$: PRINT
82Ø  IF L$ < > "" THEN B$ = " AT "
83Ø  PRINT C$" " A$ B$ L$;: E$ = B$:
   GOSUB 3ØØØ: T$ = E$: E$ = B$:
   B$ = T$
84Ø  IF E$ < > "Y" THEN 56Ø
845  ONERR GOTO 137Ø
85Ø  PRINT : HOME : VTAB 12
852  Y = 4Ø - LEN (A$) -  LEN (C$) - LEN
   (L$) - LEN (B$) - 5
854  IF Y < 2 THEN Y = 2
856  HTAB (Y / 2)
86Ø  PRINT C$ "ING " A$ B$ L$
87Ø  IF L$ < > "" THEN L$ = ", A" + L$
```

line-by-line explanation

(all "='s). If it is blank, then get a file selection. Otherwise continue printing file names until both columns are filled.

Line 23Ø is also known as the SOFT entry point.

- GET FILE SELECTION (44Ø-55Ø) -

44Ø-46Ø: Set text window and print page number and other info.

47Ø: Get a file selection and check if user wishes to:

475, 48Ø: Page thru the catalog.

5ØØ: Need help.

51Ø: Exit the menu.

52Ø: Read another catalog.

If a file selection was entered, make sure that the file exists.

53Ø: If it doesn't, ignore this selection and get another.

55Ø: Otherwise, go to the Mini-Menu.

- THE MINI-MENU -

The MINI-MENU is divided into 12 main routines.

1. Initialize Defaults
2. Center Printing
3. UNLOCK File
4. LOCK File
5. RUN/BRUN File
6. DOS Control
7. LOAD/BLOAD File
8. DELETE File
9. RENAME File
1Ø. LOCK/UNLOCK All
11. Exit Mini-Menu
12. EXEC Text File

NOTE: The word "FLAG" is used to denote a variable whose value will cause certain actions to be taken. The flags are:

B is the "Binary/ Text" flag.
B = Ø Applesoft or Integer file.
B = 1 Binary file.
B = 2 Text file.
L is the "Lock/Unlock" flag.
L = Ø Unlocked.
L = 1 Locked.

1) INITIALIZE DEFAULTS (56Ø-65Ø)

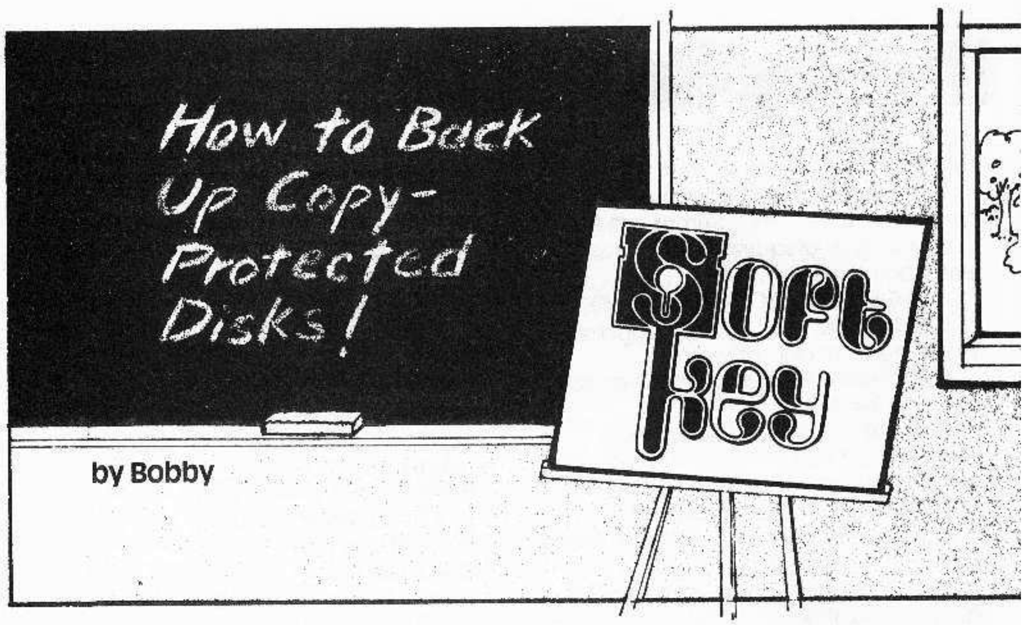
56Ø,565: Print "MINI-MENU", then set A$ equal to the proper file name and print it.

57Ø: Set default options (LOCK, LOAD, RUN).

continued on page 70

```
880  PRINT D$ C$ A$ L$
890  GOTO 230

>>>>>>> LOAD/BLOAD SELECTED FILE <<<<<<<

900  IF B$ < > "L" THEN 940
910  B$ = "": L$ = "": C$ = "BLOAD"
920  IF B < > 1 THEN C$ = "LOAD": GOTO
    820
930  GOTO 810

>>>>>>>>>>>>> DELETE FILE <<<<<<<<<<<<<

940  IF B$ < > "D" THEN 990
950  IF L = 1 THEN PRINT G$
    "THIS FILE IS LOCKED"
960  PRINT : PRINT "DELETE " A$;: GOSUB
    3000
965  IF B$ < > "Y" THEN 560
970  PRINT : HOME : VTAB 12
972  Y = 40 - LEN (A$) - 9
974  IF Y < 2 THEN Y = 2
976  HTAB (Y / 2)
979  PRINT "DELETING " A$: PRINT
    D$ "UNLOCK"A$: PRINT D$ "DELETE" A$
980  RUN

>>>>>>>>>>>>> RENAME FILE <<<<<<<<<<<<<

990   IF B$ < > "C" THEN 1160
1000  ONERR GOTO 1310
1010  IF L = 1 THEN PRINT G$
     "THIS FILE IS LOCKED"
1020  PRINT
1030  PRINT "RENAME " A$;: GOSUB 3000
1040  PRINT
1050  IF B$ < > "Y" THEN 560
1060  PRINT
1070  PRINT "CHANGE " A$ " ";: INVERSE :
     PRINT "TO";: NORMAL : PRINT " ";:
     INPUT B$
1075  IF  LEN (B$) > 30 THEN HOME : VTAB
     12: GOTO 1320
1080  IF B$ = "" THEN 560
1090  HOME : VTAB 12
1092  Y = 40 - LEN (A$) - LEN (B$) - 12
1094  IF Y > 38 OR Y < 3 THEN Y = 2
1096  HTAB (Y / 2)
1100  PRINT "CHANGING " A$ " ";: INVERSE
     : PRINT "TO";: NORMAL : PRINT " "
     B$: PRINT D$ "UNLOCK" A$: PRINT D$
     "RENAME" A$ "," B$
1110  IF L = 1 THEN PRINT D$ "LOCK" B$
1115  NA$ (X * 26 + A) = LEFT$ (NA$
     (X * 26 + A), 7) + B$
1120  GOTO 230

>>>>>>>> UNLOCK/LOCK ALL FILES <<<<<<<<<

1160  IF B$ < > "A" THEN 1280
1170  PRINT : PRINT "LOCK/UNLOCK ALL
     FILES ";: INVERSE : PRINT "L/U";:
     NORMAL : GET B$
1180  IF B$ < > "L" AND B$ < > "U"
     THEN 560
1190  IF B$ = "U" THEN B$ = "UNLOCK":
     T$ = " "
1200  IF B$ = "L" THEN B$ = "LOCK":
     T$ = "*"
1210  PRINT : HOME : VTAB 11: HTAB
     (40 - LEN (B$) - 6) / 2: PRINT B$
     "ING ALL"
1220  FOR Y = 1 TO 105
1230  IF MID$ (NA$ (Y), 8, 2) = "=="
     THEN 230
1240  VTAB 13: HTAB 10: PRINT MID$ (NA$
     (Y), 8, 30)
1250  PRINT D$ B$ MID$ (NA$ (Y), 8, 30)
1252  NA$ (Y) = T$ + RIGHT$ (NA$(Y), 37)
1260  NEXT
1270  GOTO 230

>>>>>>>>>>>>> EXIT MINI-MENU? <<<<<<<<<<<

1280  IF B$ = "X" OR B$ = CHR$ (27) OR
     B$ = CHR$ (13) OR B$ = CHR$ (24)
     THEN 230

>>>>>>>>>>>>> EXEC TEXT FILE <<<<<<<<<<<<

1282  IF B < > 2 OR B$ < > "E" THEN
     560
1284  C$ = "EXEC": L$ = "": B$ = "":
     GOTO 830
1300  TEXT : HOME : END
>>>>>>>>>>>> ERR ROUTINES <<<<<<<<<<<<<<<

1310  IF L = 1 THEN PRINT D$ "LOCK" A$
1320  PRINT : PRINT G$
     " ERR IN NEW NAME. PLEASE TRY
     AGAIN"
1330  ONERR GOTO 1350
1340  PRINT : HTAB 7: PRINT
     "PRESS ANY KEY TO CONTINUE";: GET
     B$: GOTO 560
1350  POKE 216, 0: PRINT D$ "PR#0":
     PRINT D$ "IN#0": PRINT
1360  HOME : VTAB 12: HTAB 3: PRINT G$
     "!!! - UNABLE TO READ
     DIRECTORY !!!": POKE 216, 0: END
1370  HOME : VTAB 12: IF LEFT$ (L$, 3)
     = ",A$" THEN HTAB 10: PRINT G$
     "ERR IN LOAD ADDRESS": GOTO 1340
1380  PRINT G$ "STRANGE ERR. I WILL
     RE-READ THE CATALOG": PRINT : HTAB
     6: PRINT "PRESS ANY KEY TO
     CONTINUE";: GET B$: GOTO 55

>>>>>> SUBROUTINE TO GET KEYPRESS <<<<<<

3000  PRINT " ";: INVERSE : PRINT
     "Y/N";: NORMAL : GET B$: PRINT :
     RETURN

>>>>>>>>>> PRINT INSTRUCTIONS <<<<<<<<<<

3010  TEXT : HOME : VTAB 2: HTAB 16:
     PRINT "COMMANDS": PRINT : PRINT
     HE$: PRINT
3020  PRINT "> RIGHT ARROW MOVE FORWARD
     THRU MENU": PRINT
3030  PRINT "< LEFT ARROW MOVE BACKWARD
     THRU MENU": PRINT
3040  INVERSE : PRINT "M";: NORMAL :
     PRINT " 'RETURN' GET'S A NEW
     CATALOG": PRINT
3050  INVERSE : PRINT "C";: NORMAL :
     PRINT " CTRL 'C' GET A NORMAL
     CATALOG": PRINT
3060  INVERSE : PRINT "X";: NORMAL :
     PRINT " CTRL 'X' EXIT MENU":
     PRINT
3070  PRINT "   ANY LETTER A-Z GO'S TO
     MINI-MENU": PRINT
3080  VTAB 20: HTAB 7: PRINT "PRESS ANY
     KEY TO CONTINUE";: GET B$: PRINT :
     GOTO 230
```

Softkey 3: Controlling the I.O.B.



by Bobby

Softkey III will deal with disks that change the address and data marks from track to track, or use the standard DOS marks but change the sector numbers. None of the "MUFFIN's" will work on these disks. They require a special program to directly access the disk.

This program accomplishes the task by doing a "brute force', sector-by-sector copy of each track. On a 13 sector disk, it will cause the disk drive to recalibrate (the racheting sound) on any sector that doesn't contain data. This is nothing to worry about and does not mean the program is not working.

The program calls the RWTS (Read or Write a Track/Sector) directly and uses its own IOB (Input/Output Block). The procedure for this is very well docmented in the DOS manual (page 94-98).

Basically, the IOB is a list of parameters that is used by the RWTS whenever a disk access is necessary. This list includes the Slot, Drive, Track, Sector and Volume number. Each call to the RWTS will, depending on the Command code, position the Read/Write head only, Read or Write a single sector, or INITialize the disk.

SETTING UP THE CONTROLLER

The controller is the heart of the IOB program. It starts at line 1000 and defines the Track/Sector pairs that you wish to copy and the address and data mark values you will use to read or write them.

Two basic controllers are listed. The first assumes that only the sector numbers are changed. The second assumes that only the address and data marks are changed. They are presented as samples of what you can do with the program. The controller can be as simple or as complex as necessary.

Controller #1

This controller is set up to copy a disk that uses standard address and data marks, but changes the sector numbers so that they step by even increments.

NOTE: Use the 4+4 Conversion Chart in Disklocks (issue 2) to convert the encoded sector bytes for each address mark to hex. Verify that the sector numbers step by even number increments (ie. 0,2,4,6,8, 10 etc.)

```
1000 FOR TK = 3 TO 34
1010 DV = 1 : CD = RD : GOSUB 50 :
     GOSUB 85
1020 DV = 2 : CD = WR : GOSUB 50 :
     GOSUB 80
1030 NEXT
```

Often the checksum on the original disk will be altered. If you think this is the case then add line 200 to the IOB program.

```
200  GOSUB 55
```

This change will cause DOS to look at the start of address and start of data marks and ignore the checksum and end marks.

Controller #2

This controller is set up to copy a disk that uses different address and data marks on each track. It assumes that all six bytes (3 address/3 data) are changed. If only one or two are changed, the controller can be altered to show this. (IE. If address mark A2 is unchanged then the 'READ A2' could be omitted from line 60 and the 'POKE 47455,A2' omitted from line 70.)

Both controllers set up a FOR/NEXT loop for the track numbers and GOSUBs to the appropriate routine to READ/WRITE a track.

```
1000 FOR X = 3 TO 34
1010 DV = 1 : CD = RD : GOSUB 50 : GOSUB
     60 : GOSUB 80
1020 DV = 2 : CD = WR : GOSUB 50 : GOSUB
     90 : GOSUB 80
1030 NEXT
```

When typing the data for line 63050, be sure to maintain the proper order. The routine that READs the data assumes that the address mark information is first and the data mark information is second.

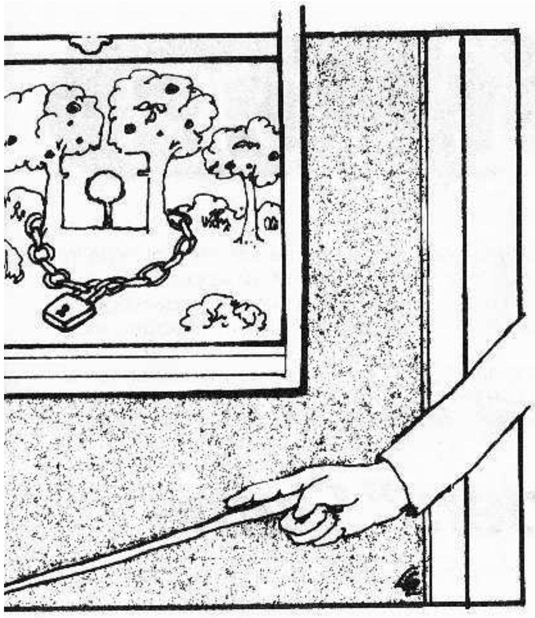
How To Use the

Requirements:
Apple II or Apple II+ with 48K and Applesoft in ROM
13 sector or 16 sector DOS (as appropriate)
Blank disk
IOB program
DiskView or other nibbler

Perform the following steps:
1. RUN DiskView
2. Remove the DiskView disk and insert the back-up copy of your program disk.
NEVER USE THE ORIGINAL DISK
3. READ track 3 and look for the start of address and the start of data marks. The normal values are D5 AA 96 (address) and D5 AA AD (data). Write

I.O.B. Pro

```
     >> set LOMEM above buffer <<
10  TEXT : HOME : LOMEM : 16385 : GOSUB
    63000 : GOTO 100
    >> print track and sector info <<
20  HOME : VTAB 12 : HTAB 12 : PRINT
    "TRACK "TK"  SECTOR "ST : RETURN
   >> center text on screen and print <<
30  HTAB 20 - ( LEN (A$) / 2) : PRINT A$
    ;: RETURN
   >> wait for key press then return <<
40  HOME : VTAB 12 : GOSUB 30 : VTAB 14
    : A$ = "PRESS ANY KEY TO CONTINUE "
    : GOSUB 30 : GET AN$ : RETURN
     >> poke variables into IOB <<
50  POKE BUF,32 : POKE CMD,CD : POKE
    TRK,TK : POKE SCT,ST : POKE DRV,DV :
    POKE VOL,VL : RETURN
```



I.O.B. Program

down the values that you find.

NOTE: See Disklocks (issue 2) for more information on address and data marks. Practice on a normal DOS disk until you are familiar with the disk format and can easily find the address and data marks. 4. Repeat step 3 for each track (3 to 34).

5. Use the Byte Cross-reference Chart to convert the hex bytes to decimal.

6. Load the IOB program.

7. Make the appropriate changes in the controller (lines 1000 thru 2000) Insert the decimal values from step 5 into the data statement(s) at line 63050.

8. Run the program.

gram Listing

```
     >> get address and data marks <<
60  READ A1 : READ A2 : READ A3 :
    READ D1 : READ D2 : READ D3
       >> poke the info into DOS <<
70  POKE 47445,A1 : POKE 47455,A2 :
    POKE 47466,A3 : POKE 47335,D1 :
    POKE 47345,D2 : POKE 47356,D3 :
    RETURN
          >> read/write a track <<
80  FOR ST = 0 TO DOS : POKE SCT,ST :
    GOSUB 20 : CALL IO : POKE BUF, PEEK
    (BUF) + 1 : NEXT : RETURN
 >> read a track by even no. sectors <<
85  FOR ST = 0 TO DOS * 2 STEP 2 : POKE
    SCT,ST : GOSUB 20 : CALL IO : POKE
    BUF, PEEK (BUF) + 1 : NEXT : RETURN
```

continued on page 53

Softkey 4: Boot Code Tracing

Requirements:
A copy of Apple Galaxian
48K slave Disk (with the hello program deleted)
Knowledge of machine code/assembly language

Boot code tracing is the most difficult of all the SoftKeys presented. It is also the most effective method of transferring single load binary programs, which are programs which load into the Apple at Boot and do not access the disk again.

This SoftKey is definitely not for beginners. You must have some knowledge of machine code or assembly language.

The initial Boot or Boot 0 is determined by a program on the disk controller card. This program is stored in ROM (Read Only Memory) and cannot be changed by the program on a disk. A non-standard format may be used on a disk to prevent copying, but that disk will not Boot on your Apple unless track 0, sector 0 is readable by the disk controller card.

This means that you can also read it, and by controlling the boot process you can determine where the program is and save it to disk.

In order to demonstrate this process, the Boot of the binary program Apple Galaxian will be traced. This is a familiar game and almost everyone owns a copy.

The Boot 0 code is at $C600. If your controller card is not in slot 6, then change the number after the $0C to correspond to the appropriate slot (IE. Slot 5 would be $C500). First the code will be moved down into RAM (Random Access Memory) where a portion of it can be modified. Turn on your Apple, then enter the monitor (CALL 151) and type:

9600< C600.C6FFM

This will move the Boot code to page $96 ($9600), where the changes will be made.

NOTE: Commands will be on a separate line and printed exactly as you should enter them. Press return after each line entry. If a command has already been listed, it will be referred to but not listed again.

Be sure you move the code to a page boundary that corresponds to the slot that your controller card is in (IE. for slot 6 - $9600, $8600, $7600 etc. or for slot 5 - $9500, $8500 etc.). The reason for this is that the Boot 0 code contains a routine that finds which slot the controller card is in. It does this by calling a return code in the F8 ROM and extracting the return address to locate the page boundary. The code itself is relocatable (will run anywhere in memory).

The purpose of Boot 0 is to transfer the code stored on disk at track 0, sector 0 into memory and to execute it. Then the disk code (Boot 1) will take over the Boot process. Here is where the first change will be made.

Examine the code at the end of page $96. At $96F8 you will find a JMP to $801. This is the next Boot stage (Boot 1). Boot 0 needs to load and not run the Boot 1 code at page $08, so the code at $96F8 will be changed to JMP to the monitor after turning off the drive motor. In order to save having to reenter the same code, put the routine at $9500.

96FA:95 N 9501:AD E8 C0 4C 59 FF

The base address $C088 is used to turn the drive motor off. You must add the slot number of your controller card to this address in the form $S0 where "S" is the slot (IE. slot 5 would be $C088 + $50 = $C0D8). For those of you not familiar with hexadecimal numbers, there is a table on page 82 of the Apple II reference manual with the base addresses and conversions for each slot.

After making the changes and checking that they are correct, you are ready to begin. Run the code at $9600 (Boot 0).

$9600G

When the drive stops, Boot 1 will be in memory at page $08. If you examine the code at $800 you will see that it is written to run at page $02. The first part of the code is a move routine that transfers the code from page $08 to page $02, then JMPs to page $02.

Boot 1 should be moved to page $98 so you can make changes. The code must be moved or else the next time you run Boot 0 it will be overwritten.

9800< 800.8FFM

Now the modified Boot 0 can be linked at $9600 to the Boot 1 code at $9800. The move routine should then be changed so it will work at its new address, and the JMP to the next Boot stage should be changed to the routine at $9500.

96FA:98 N 9805:98 N 9843:95

The "N" is used as a null command to separate changes. It is the monitor command to set normal mode and does nothing, since you are already in normal mode.

After you have made the changes and checked that they are correct, run the code at $9600 again.

continued on page 53

apple softies

Apple Softies are programs presented in a way that will reveal certain aspects of AppleSoft BASIC. These programs are not only valuable in themselves (whether they are instructive, business or entertainment software) but are also valuable as examples of how to utilize AppleSoft in your own programs. By typing in these programs and then altering lines to fit your own needs and esthetics, you will refine your own knowledge of AppleSoft. More importantly, we hope that you will make creative improvements. If you do, please send them in to us so that other readers may enjoy and learn from them.

NOTE: When entering AppleSofties, remember that only the first two characters of a variable name are important. You may wish to shorten some of the long variable names. If so, please do not remove the variable suffix (%, $, or array notation). Use the editing features of AppleSoft as well as some of its input shortcuts (such as using "?" for "PRINT", and ignoring spaces except in PRINT, DATA, or REM statements.

A Hi-Res "Shoot The Invaders" Game

Zyphyr Wars 2.Ø is an Applesoft hi-res 'shoot the invaders' type of arcade game (which abound in Apple-dom). This particular one is not only a complete game in itself, but is also a 'core' program that you are encouraged to alter and adapt so that you get a 'feel' for the ease with which such games are created. (It is an egotistical myth that the writing of computer games is a difficult or a special art!)

special features

This program is a fast single-player game with some interesting features worthy of publication in apple-softie. Instead of using a hi-res text/character generator (but easily adapted to use one, by the way), this version flashes the 'hit points' by switching screens (text and hi-res) in rapid, repetitive bursts that give the illusion of text on the hi-res page. Also, the UFOs or Zyphyrs (henceforth called 'Zs') appear to 'zip' across the screen, much faster than is really possible. And you have an entire cityscape to protect rather than simple bunkers to hide behind. Plus, this game uses SHAPE TABLES.

A city is built on the bottom and your ship, a satellite, is DRAWn at the top. Ten Zs appear between your ship and the city skyline below and you must destroy them before they destroy you and your city. If they succeed in creating deep craters, the game will end. These Zs zap your city and your ship with death rays, while you must zap back using your own ray. But because you will be shooting down at them, a miss will destroy parts of your own city. If you miss too many times, the game ends dramatically.

When you destroy the first ten Zs, you are elevated to the next level and another ten will appear. Each higher level that you achieve will subject your ship to more frequent attacks. And if your ship is hit, it plummets to the ground, destroying the building beneath it. You have only a few ships.

This Apple-softie will show you that arcade-type games can be written in Applesoft, too. However, I do use several 'tricks' that allow it to play quickly.

1. illusions of speed

First of all, I don't really 'move' the Zs around 'bit by bit.' Instead, I give the illusion of fast movement by connecting their old and new positions with a line and then adding a 'zip!' sound, hence the name 'Zyphyr,' not zipper. (I do, however, make your ship move 'bit-by-bit' whenever it crashes.)

2. paddles only

Secondly, when I DO move the ship back and forth, I do not make it step across smoothly. I use absolute paddle positions and the ship 'materializes' at whatever value the paddle happens to be turned to when its value is checked by the program. That means that the player can also make the ship 'zip' from one side to the other, a necessary feature if the player wishes to zap the Zs.

3. rays, not bombs

Thirdly, I don't move 'missiles' or 'bombs' because that slows everything down (remember Text Invaders?). I HPLOT or XDRAW lines (rays) instead, giving the appearance of an immediate 'hit' or 'miss.'

4. hit or miss?

And fourth, I don't make the program go through long searches just to determine if a ray has hit or missed. The program only permits the Zs to appear directly over the center of each column of buildings, and then only one Z per building. The player's ship moves across in a similar fashion, appearing only over the center of each building. Therefore, the only search that is necessary is to simply determine if the column in which the ship presently resides is also occupied by a Z!

a shape table

The buildings consist of 'shapes' stacked up in columns. The Zs are also conglomerate shapes (allowing the programmer to change their shapes). But the player's ship is a single 'shape.'

The shapes are poked into memory page 3 ($3ØØ hex, or 768 in decimal) along with a short sound routine. DRAW and XDRAW are used extensively, along with HPLOTs in order to give the readers examples of various hi-res graphic possibilities. The reader may wish to turn all HPLOTs into XDRAWS.

sound effects

There are a variety of noises used in this game, including a soft 'zip' sound. Should you have a favorite sound routine, you might want to use it instead.

This is how the program works:

the program

First off, the 4Ø columns (representing the 4Ø character-column text format) must be reserved by DIMensioning them at the very start.

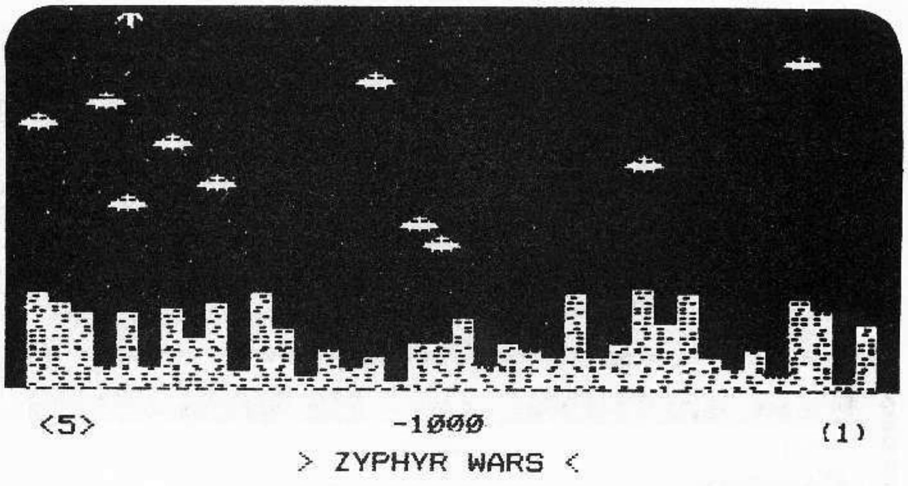
COL%(4Ø) . . . This integer variable array will store which Z is above which building (column). If empty, it will be equal to zero.

HEIGHT(4Ø) . . . This array will store the height of the building. When struck by a ray or a falling ship, the building will diminish in height until a crater is dug into the ground.

XUFO%(1Ø), YUFO%(1Ø) . . . These arrays store the X and Y positions of all ten Zs.

Zyphyr Wars 2.0

by Bev R. Haight



Also, to keep track of the player's ship, there are:

PN%, or the New Paddle position, and PO%, or the Old Paddle position.

hi-res text trickery

After the variables are defined, the game itself is set up in lines 12000 through 12090. It HPLOTS the earth (EARTH = 152) on hi-res page 1 (13000-13030). Then, on text page 1 (13100-13130), it sets the bottom of the text window to 19 (13180). (That means that the 4 rows seen below the hi-res screen in the mixed-mode are actually outside of the window. This will pose a small problem later when we try to print below the window.) The ground is continued into the text page. This gives the illusion of having text on the hi-res page. This is an effect we will later enhance.

the city shapes up

Next, the buildings are calculated and XDRAWn (14000-14090). Only the inner 38 columns are used for buildings (14000). A random height is chosen and subtracted from EARTH. Why subtraction? Well, remember that as the height of the building increases, the actual Y value gets LOWER, not higher, because the Y value at the very top of the screen is zero!

Now, one of the 5 building shapes from the shape table is chosen at random (14040). These 5 shapes show a window (or windows) at various positions (see SHAPES FOR ZYPHYR WARS). When stacked together (14060), something resembling a modern building is created. Shape #6 is a plain and simple vector dash, but at SCALE 7 it is a solid line (14050). This line separates the building levels.

stars in 4 colors

After the buildings are up, stars are drawn from top to bottom (14100). Because stars are just PLOTed points (14130), color is inevitable even when specifying HCOLOR #3 or HCOLOR = 7. By alternating between the two types of white, all 4 colors are PLOTted.

The Zs are calculated next. Each will appear on a specific row corresponding to a text page row. I do this for a reason that you will see later. Only one Z per column and per row is permitted (15010). COL%(n) is then filled with the row number of the Z that occupies it.

ready to play......

The game is now ready to play. The text displays are PRINTed. This is accomplished in a unique way because it must be PRINTed below the text window. Normally, if you tried to PRINT a string of more than one character outside the window, only the first character will be printed outside, the rest will appear inside. Therefore, a long string must be VTABed and HTABed into place one character at a time.

The main game sequence is controlled in lines 100 to 190, a loop with GOSUBs. This allows you to add your own routines. For example, you may want to make this a two player game with a tank or gun moving along the bottom that shoots upward at the other player's ship or at the Zs.

GOSUB 1000

The first GOSUB moves the satellite in response to the paddle. I use paddle 1. It DRAWs and reDRAWs the ship. But drawing and redrawing anything on a single hi-res page will make it flicker. So, instead of using two entire hi-res pages to remove the flicker effect (which uses up too much memory), I remove it by allowing it to erase and redraw only when the position of the ship must be changed (1100).

GOSUB 2000

The second GOSUB is more complex and is composed of its own series of GOSUBs. It first moves the Zs, but only one of them. I originally moved them sequencially from top to bottom, but it made the game too predictable, so they now move at random. To move them, the program must first check if the random Z it selected still exists (it could have been shot). If it doesn't exist then that's the end of that. If the Z is still around, the program then generates another random number and checks if that particular column is empty. If not then it runs off to a routine that allows the Z to shoot down at the city. But if the column is empty, the program checks to see if the Z should shoot at your ship. The chances of it shooting at you will increase as you ascend the various levels (by wiping out all ten Zs).

10 zipping zyphyrs

Finally, the Z gets to move. It deletes itself from its old COL% and places its number in the new COL%. It changes its XUFO% value accordingly. Now comes the illusion of movement, the 'zip.' A FOR-NEXT loop is initiated that goes from 3 to 4 because I let it determine the HCOLOR (white is 3 and black is 4). Drawing and undrawing the Z is a GOSUB inside a GOSUB. And, instead of storing a Z 'shape' in the vector table (as I did with the ship), I opted for a more flexible but slower format (for variety) that would permit you to alter the Z shape (200-260) more easily without changing the shape table itself.

GOSUB 3000

The third main GOSUB is conditional and depends on whether the paddle button is depressed at the time that the routine checks it. If not depressed, the program loops back on itself. If pressed, it means that the ship is shooting. It checks to see if a Z occupies its column. If so, then the Z vanishes and U% (UFO counter) is raised by one. If over 9, a new level has been reached and ten new Zs are created. If not, then too bad because any building below will be completely wiped out and your score will be decremented by a thousand points. And if you accumulate a negative 100,000 points then the game ends.

Switching screens

One of the things you should notice whenever you hit a Z is that the hit points are flashed in the same place where the Z expired. That effect is accomplished by switching screens rapidly (9030-9080). In this way, one can project text on the hi-res page for brief moments without using a character (or block-graphics) generator.

This version is small enough to fit comfortably beneath hi-res page 1. However, if you add any more routines, or if you add more remarks, you should consider loading it above hi-res page 1 or you might just wipe out the tail end of your program when you try to RUN it. Use a 'loader program' similar to the one used with B. Bryte's Island Map Editor. Also, if you or your club has access to one of the Applesoft compilers now available, you should try compiling it to make it more challenging.

EXPLANATIONS OF THE VARIABLES

Counters___

```
A, AA, B, I, R, RR, S, Z
U%---------Zyphyr counter, if 10 then
   increment LEVEL%.
```

Strings___

```
A$---------used to print messages to
   screen and to GET an answer from
   the player. <850,880>
GUN$-------number of guns. <350>
LEVEL$-----level number. <360>
SC$--------score number. <9020-9050>
```

Permanent Values___

```
BUZZ = -16336 (speaker). <300,310,320,
   610,740,810,820,9040,10010>
B1 = -16286 (button #1). <120,10010>
EARTH = 152, the game ground level.
   <630,810,820,830,950,6010,10000,
   13010,14020,14030,14100>
START = 768  (HEX=$300), a convenient
   place to put the noise routine and
   shape table. <11100,11110>
```

Flags___

```
AGAIN -----to avoid "OUT OF DATA".
   <10090>
FUFO% -----0 or 1, to flash text page.
   <9000,9040,9090>
HC---------1 or -1, to select HCOLOR.
   <10020,14110>
SF%--------0 or 1, to shoot or not to
   shoot at the satellite. <600,2400>
```

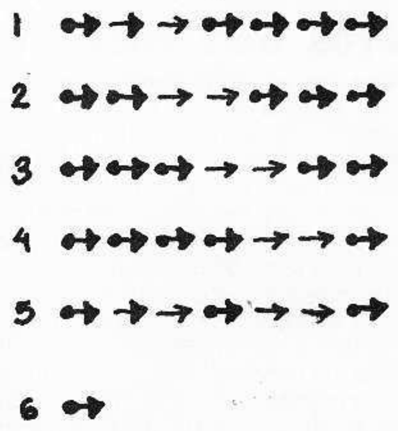
Arrays___

```
COL%(40)----what Zyphyr is over each
    building (column), a number from
    zero to ten. <10,260,2200,2750,
    2900.3010-3030,15010-15050>
HEIGHT%(40)-the height of the 40
    buildings.
    <10,600,640-650,720,810-830,3010,
    6010,14020-14030>
XUFO%(10)---what building the zyphyr
    is over, a number from 2 to 39.
    <10,200,260,600,2300,2750.2800,
    8010,15020>
YUFO%(10)---the Zyphyr's Y location,
    equal to (Zyphyr * 8) +12
    <10,200,260,600,2050.2700,3020-
    3030,8000-8010,15030>
```

Others___

```
ADD--------Points to add to score.
   <840,940-960,9010-9020>
   <2500-2600>
GUN%-------Number of satellites.
   <350,750,10000>
HEIGHT-----Used to calculate HEIGHT%.
   <14010-14020>
LEVEL%-----Level number. <360,550,840,
   940-960,2400,9010,1000>
MEM--------Used to POKE in DATA.
   <11110-11150>
N----------Noise maker. <300-320,610,
   810-820,9040>
PN%--------New value of Paddle.
   <260,650-660,710-740,810-830,1000-
   1300,3010-3030,9010>
PO%--------Old paddle value.
   <1100-1300>
PTS--------Points, in a STRing it is
   called SC$. <9020-10000>
Q%---------New Zyphyr number, equal to
   RND(1) * 37 +1. <2100-2900>
QUANTITY---DATA statement value to
   POKE in to MEM. <11120-11140>
R%---------what building shape to
   XDRAW, a number from 1 to 5.
   <14040,14060>
STAR-------X location of random stars.
   <14120-14130>
T%---------1 to 38, temporary column
   number to calculate Zyphyrs.
   <15010-15050>
TX%--------X location of the satellite
   <410-420,3000>
TY%--------Y location of the satellite
   usually 5. <410-420,3010-3020,9010>
UX%--------X location of the center of
   Zyphyr, used to draw ray. <600-670>
UY%--------Y location of Zyphyr.
   <600-670>
X----------X location of super BOOM!
   <8120,8130>
X%---------X location of the left edge
   of the Zyphyr. <200-260>
Y----------Y location of super BOOM!
   <8120,8130>
Y%---------Y location of the left edge
   of the zyphyr. <200-260>
ZH%--------The bottom of the ray, or
   Y, usually a building top.
   <600-670,910-950>
ZX%--------Same as above, but the X.
   <600-660,910-930>
```

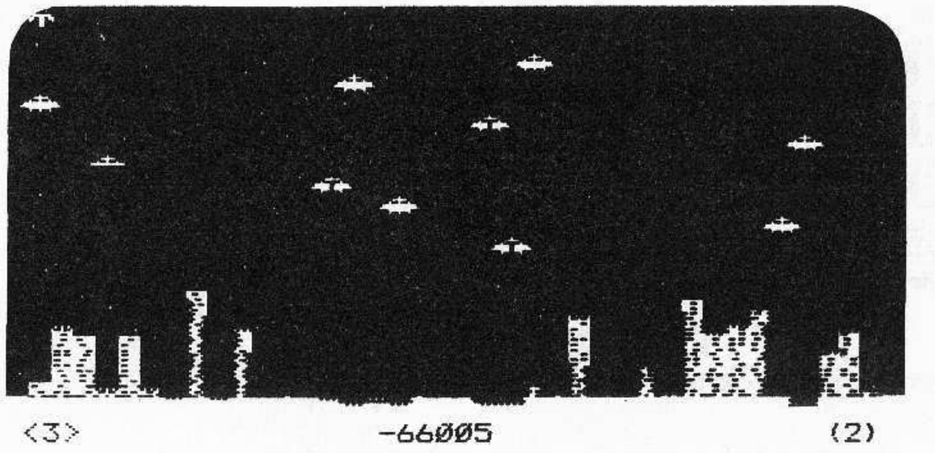
VECTOR SHAPES



AppleSoft LISTing for ZYPHYR WARS

```
======== COPYRIGHT CREDITS =============

1  REM  ZYPHYR WARS BY BEV R. HAIGHT
2  REM  COPYRIGHT 1982 BY
        SOFTKEY PUBLISHING
        14404 EAST D STREET,
        TACOMA, WA 98445

10 TEXT : HOME : HGR : DIM HEIGHT%(40),
   XUFO%(40), YUFO%(40), COL%(40) : GOTO
   10000


======== MAIN GAME SEQUENCER ==========

100  GOSUB 1000
110  GOSUB 2000
120  IF PEEK (B1) > 127 THEN GOSUB 3000
190  GOTO 100
```

```
============== DRAW ZYPHYRS ============

205  SCALE = 1 : DRAW 6 AT X% + 3, Y% -
     2
21Ø  SCALE = 5 : DRAW 6 AT X% + 1, Y% -
     1
22Ø  SCALE = 1 : DRAW 5 AT X%, Y%
225  SCALE = 11 : DRAW 6 AT X% - 2, Y% +
     1
23Ø  SCALE = 13 : DRAW 6 AT X% - 3, Y% +
     2
235  SCALE = 7 : DRAW 6 AT X%, Y% + 3
24Ø  SCALE = 1 : DRAW 5 AT X%, Y% + 4
25Ø  RETURN
26Ø  X% = XUFO% ( COL% ( PN% ) ) * 7:
     Y% = YUFO% ( COL% ( PN% ) ) : ROT =
     Ø : GOTO 2Ø5

================== BUZZ! ===============

3ØØ  FOR S = 1 TO 3 : N = PEEK (BUZZ) :
     NEXT : RETURN
31Ø  FOR S = 1 TO 1Ø : N =  PEEK (BUZZ)
     : NEXT : RETURN
32Ø  N = PEEK (BUZZ) -  PEEK (BUZZ) :
     RETURN

=============== DISPLAY GUNS ===========

35Ø  HTAB 3 : GUN$ = "<" + STR$ (GUN%) +
     ">  " : INVERSE : VTAB 21 : FOR A =
     1 TO LEN (GUN$) : PRINT MID$ (GUN$,
     A, 1) ;: NEXT : NORMAL : RETURN

=========== DISPLAY LEVEL ==============

36Ø  VTAB 21 : HTAB 35 : LEVEL$ = "(" +
     STR$ (LEVEL%) + ")" : INVERSE :
     FOR A = 1 TO LEN (LEVEL$) : PRINT
     MID$ (LEVEL$, A, 1) ;: NEXT :
     NORMAL : GOSUB 53ØØ : RETURN

============= DRAW RAY =================

4ØØ  ROT= 16 : SCALE= 3
41Ø  HCOLOR = 3 : FOR A = 8 TO TY% STEP
     TY% / 1Ø : DRAW 6 AT TX%, A : POKE
     6, A : POKE 7, 3 : CALL 768 : NEXT
42Ø  HCOLOR = Ø : FOR A = 8 TO TY% STEP
     TY% / 1Ø : DRAW 6 AT TX%, A : NEXT
44Ø  RETURN

=============LEVEL COUNTER =============

55Ø  U% = U% + 1 : IF U% > 9 THEN U% =
     Ø : LEVEL% = LEVEL% + 1 : GOSUB
     15ØØØ
56Ø  RETURN

============== ZYPHYR SHOOTS ===========

6ØØ  UX% = XUFO% (Z) * 7 + 3 : UY% =
     YUFO% (Z) : ZX% = XUFO% (Z) : ZH%
     = HEIGHT% (ZX%) : IF SF% > Ø THEN
     SF% = Ø : GOTO 66Ø
61Ø  FOR AA = 3 TO 4 : HCOLOR = AA :
     HPLOT UX%, UY% TO UX%, ZH% : NEXT :
     GOSUB 1ØØØ
62Ø  GOSUB 9ØØ
63Ø  ZH% = ZH% + 2 : IF ZH% > EARTH
     THEN  GOSUB 95Ø
64Ø  HEIGHT% (ZX%) = ZH% : RETURN
65Ø  UX% = PN% * 7 + 3 : UY% = 7 : ZX%
     = PN% : ZH% = HEIGHT% (ZX%) : GOTO
     62Ø
```

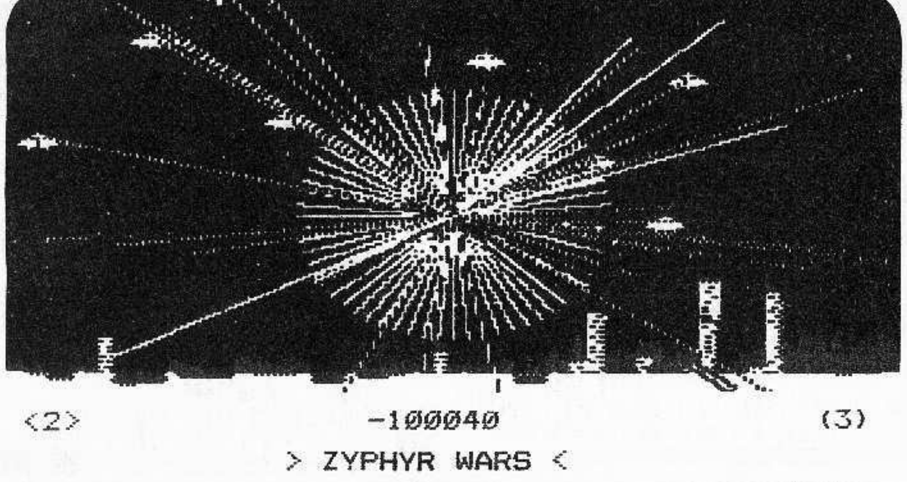


> ZYPHYR WARS <

```
66Ø  SCALE = 3 : ROT = 16 : ZH% = Ø :
     IF PN% = ZX% THEN ZH% = 5
67Ø  FOR AA = 3 TO 4 : HCOLOR = AA :
     HPLOT UX%, UY% TO UX%, ZH% : GOSUB
     32Ø : NEXT : GOSUB 52ØØ : GOSUB 3ØØ
     : IF ZH% = 5 THEN 7ØØ
68Ø  RETURN

============ SATELLITE FALLS ==========

7ØØ  ROT = Ø : SCALE = 1
71Ø  HCOLOR = Ø : DRAW 8 AT PN% * 7, 5
72Ø  ROT = 32 : FOR A = Ø TO HEIGHT%
     (PN%) STEP 2
73Ø  FOR AA = 1 TO 2
74Ø  XDRAW 8 AT PN% * 7 + 7, A : POKE
     6, A : POKE 7, 6 : CALL 768 : N =
     PEEK (BUZZ)
75Ø  NEXT : NEXT : GUN% = GUN% - 1 :
     GOSUB 5ØØØ : IF GUN% = Ø THEN 85Ø

=========== GROUND EXPLOSION ==========

8ØØ  HCOLOR = Ø
81Ø  IF HEIGHT% (PN%) >  = EARTH THEN
     FOR AA = 1 TO 2 : FOR A = 2 TO 6 :
     SCALE= 3 * A : FOR R = Ø TO 2Ø
     STEP 2 : FOR RR = - 1 TO 1 STEP 2
     : ROT = 112 - R * RR : XDRAW 6 AT
     PN% * 7 + 3, EARTH : NEXT : N =
     PEEK (BUZZ) : NEXT : NEXT : NEXT :
     GOSUB 35Ø

=========== WHOLE BLDG. GONE ==========

82Ø  FOR AA = HEIGHT% (PN%) TO EARTH
     STEP 2 : HPLOT PN% * 7 - 1, AA TO
     PN% * 7 + 7, AA : HPLOT PN% * 7 -
     2, AA - 1 TO PN% * 7 + 8, AA - 1 :
     N =  PEEK (BUZZ) : NEXT
83Ø  HEIGHT% (PN%) = EARTH : GOSUB 35Ø
84Ø  ADD = - 1ØØØ * LEVEL% : GOSUB 9Ø2Ø
     : RETURN

============ END OF GAME ==============

85Ø  A$ = "  END OF GAME !" : FOR B = 1
     TO 15 : FOR AA = 1 TO B : VTAB 1Ø
     : HTAB 13 : PRINT  LEFT$ (A$, B)
86Ø  POKE - 163Ø3, Ø : GOSUB 31Ø : POKE
     - 163Ø4, Ø : NEXT AA : A = B * 13
     : GOSUB 53ØØ
```

```
87Ø  NEXT B : TEXT : VTAB 1 : HTAB 1Ø
     : PRINT "ANOTHER GAME? <Y><N> " ;
88Ø  GET A$ : IF A$ = "Y" THEN CLEAR :
     GOTO 1Ø
89Ø  HOME : END

========= BUILDING LOSES A LEVEL ======

9ØØ  HCOLOR = Ø : ROT = Ø
91Ø  SCALE = 7 : DRAW 6 AT ZX% * 7,
     ZH% - 1 : GOSUB 3ØØ
92Ø  SCALE = 9: DRAW 6 AT ZX% * 7 - 1,
     ZH% : GOSUB 3ØØ
93Ø  HCOLOR = 5 : SCALE = 7 : DRAW 6 AT
     ZX% * 7, ZH% + 1
94Ø  ADD = - 1Ø * LEVEL% : GOSUB 9Ø2Ø :
     RETURN
95Ø  IF ZH% > EARTH + 4 THEN ZH% =
     EARTH + 4 : CC = CC + 1 : IF CC >
     9 THEN  GOSUB 6ØØØ
96Ø  ADD = - 2Ø * LEVEL% : GOSUB 9Ø2Ø :
     RETURN

========== MOVE SATELLITE =============

1ØØØ  PN% =  PDL (1) / 5 + 1 : IF PN% >
      38 THEN PN% = 38
1Ø5Ø  ROT = Ø : SCALE = 1
11ØØ  IF PN% <  > PO% THEN  HCOLOR = Ø
      : DRAW 8 AT PO% * 7, 5
12ØØ  HCOLOR = 3 : DRAW 8 AT PN% * 7,
      5
13ØØ  PO% = PN%
14ØØ  RETURN

=========== MOVE ZYPHYRS ==============

2ØØØ  Z =  RND (1) * 1Ø + 1
2Ø5Ø  IF YUFO% (Z) < 1 THEN  RETURN
21ØØ  Q% =  RND (1) * 37 + 1
22ØØ  IF COL% (Q%) > Ø THEN  GOTO 6ØØ
23ØØ  X1% = XUFO% (Z) * 7 : X2% = Q% *
      7
24ØØ  IF RND (1) * 1Ø < LEVEL% THEN
      GOSUB 1ØØØ : SF% = 1 : GOSUB 6ØØ
27ØØ  FOR AA = 5 TO 4 STEP - 1 : HCOLOR
      = AA : HPLOT X1%, YUFO% (Z) TO
      X2%, YUFO% (Z) : NEXT : GOSUB
      1ØØØ : GOSUB 54ØØ
275Ø  COL% (XUFO% (Z) ) = Ø
28ØØ  HCOLOR = Ø : GOSUB 2ØØ : XUFO%
      (Z) = Q% : HCOLOR = 3 : GOSUB 2ØØ
```

```
29ØØ  COL% (G%) = 1 : RETURN

=========== SATELLITE SHOOTS ==========

3ØØØ  TX% = PN% * 7 + 4
3Ø1Ø  IF COL% (PN%) = Ø THEN TY% =
      HEIGHT% (PN%) - 2 : GOSUB 4ØØ :
      HCOLOR = Ø : GOSUB 82Ø : RETURN
3Ø2Ø  TY% = YUFO% (COL% (PN%) ) : GOSUB
      4ØØ : HCOLOR = Ø : GOSUB 26Ø
3Ø3Ø  YUFO% (COL% (PN%) ) = Ø : COL%
      (PN%) = Ø : GOSUB 9ØØØ : GOSUB
      55Ø : RETURN

========= SOUND OF DESTRUCTION ========

5ØØØ  FOR A = 1 TO 1Ø : POKE 6, RND (1)
      * 5Ø + 1 : POKE 7, 25Ø : CALL 768
      : NEXT : RETURN
51ØØ FOR A = 1ØØ TO 2ØØ STEP 2Ø : POKE
     6, A : POKE 7, 2 : CALL 768 : NEXT
     : N = PEEK (BUZZ) : GOSUB 1ØØØ :
     NEXT : RETURN
52ØØ FOR A = 2ØØ TO 25Ø STEP 1Ø : POKE
     6, A : POKE 7, 4 : CALL 768 : NEXT
     : RETURN
53ØØ  FOR AA = 251 TO 1 STEP - 1Ø :
      POKE 6, A : POKE 7, 5 : CALL 768
      : NEXT : RETURN
54ØØ  FOR A = 1 TO 3Ø STEP 3 : POKE 6.
      A : POKE 7, 2 : CALL 768 : NEXT :
      RETURN

============ CRATER COUNTER ============

6ØØØ  CC = Ø : FOR I = 1 TO 38
6Ø1Ø  IF HEIGHT% (I) < EARTH THEN 6Ø9Ø
6Ø5Ø  CC = CC + 1
6Ø6Ø  IF CC > 5 THEN GOSUB 8ØØØ : GOTO
      85Ø
6Ø9Ø  NEXT : RETURN
8ØØØ  FOR B = 1 TO 1Ø : IF YUFO% (B) <
      1 THEN 8Ø5Ø
8Ø1Ø  HCOLOR = 1 : HPLOT XUFO% (B) * 7
      + 4, YUFO% (B) TO 14Ø, 9Ø : GOSUB
      54ØØ
8Ø5Ø  NEXT B : SCALE = 5Ø : FOR A = Ø
      TO 64 : ROT = A : XDRAW 6 AT 14Ø,
      9Ø: NEXT A
81ØØ  FOR A = 1 TO 5Ø
811Ø  HCOLOR = RND (1) * 5 + 1
812Ø  X = RND (1) * 28Ø : Y = RND (1) *
      19Ø
813Ø  HPLOT 14Ø, 9Ø TO X, Y
814Ø  POKE 6, A : POKE 7, 3 : CALL 768
815Ø  NEXT A
819Ø  RETURN

============DISPLAY SCORE ==============

9ØØØ  HOME : FUFO% = 1
9Ø1Ø  ADD = LEVEL% * 1ØØ : VTAB TY% / 8
      + 1 : HTAB PN% : PRINT ADD
9Ø2Ø  PTS = PTS + ADD : SC$ = STR$
      (PTS) : VTAB 21 : HTAB 17 :
      INVERSE
9Ø3Ø  FOR A = 1 TO  LEN (SC$)
9Ø4Ø  IF FUFO% THEN  POKE  - 163Ø3, Ø :
      N =  PEEK (BUZZ)
9Ø5Ø  PRINT  MID$ (SC$, A, 1) ;
9Ø7Ø  POKE  - 163Ø4,Ø
9Ø9Ø  NEXT : PRINT "   " : NORMAL : IF
      PTS < - 1ØØØØØ THEN  GOSUB 8ØØØ :
      GOTO 85Ø
9Ø9Ø  HOME : FUFO% = Ø : RETURN

===========INIT VARIABLES =============

1ØØØØ  EARTH = 152 : LEVEL% = 1 : GUN%
       = 5 : PTS = Ø : CC = Ø
1ØØ1Ø  BUZZ = - 16336 : KEY = - 16384 :
       KBOARD = - 16368 : B1 = - 16286
1ØØ2Ø  HC =  - 1
1ØØ3Ø  Z = 1
1ØØ8Ø  SCALE = 1 : ROT = Ø
1ØØ9Ø  IF AGAIN > Ø THEN 12ØØØ
1Ø1ØØ  DATA 166,7,164,6,173,48,192,136,
       2Ø8,253,2Ø2,2Ø8,245,96,Ø

============ SHAPE TABLE ==============

11ØØØ  DATA 9,Ø,2Ø,Ø,24,Ø,29,Ø,33,Ø,37,
       Ø,41,Ø,43,Ø,45,Ø,Ø,Ø
11Ø1Ø  DATA 77,45,45,Ø
11Ø2Ø  DATA 45,9,45,5,Ø
11Ø3Ø  DATA 45,77,45,Ø
11Ø4Ø  DATA 45,1Ø9,41,Ø
11Ø5Ø  DATA 77,77,5,Ø
11Ø6Ø  DATA 5,Ø
11Ø7Ø  DATA 32,Ø
11Ø8Ø  DATA 44,44,53,54,46,36,36,45,46,
       54,Ø
11Ø9Ø  DATA -1
111ØØ  START = 768
1111Ø  MEM = START
1112Ø  READ QUANTITY
1113Ø  IF QUANTITY < Ø THEN 112ØØ
1114Ø  POKE MEM, QUANTITY
1115Ø  MEM = MEM + 1
1116Ø  GOTO 1112Ø
112ØØ  POKE 232, 15 : POKE 233, 3

============ SET UP GAME ==============

12ØØØ  GOSUB 13ØØØ
12Ø1Ø  GOSUB 14ØØØ
12Ø2Ø  GOSUB 141ØØ
12Ø3Ø  GOSUB 15ØØØ
12Ø4Ø  GOSUB 35Ø
12Ø9Ø  GOTO 1ØØ

============== MAKE EARTH =============

13ØØØ  HCOLOR= 3
13Ø1Ø  FOR A = EARTH TO EARTH + 1Ø
13Ø2Ø  HPLOT Ø, A TO 279, A
13Ø3Ø  NEXT
131ØØ  INVERSE
1311Ø  FOR A = 21 TO 23 : FOR AA = 1 TO
       4Ø
1312Ø  VTAB A : HTAB AA : PRINT " " ;
1313Ø  NEXT : NEXT
1314Ø  VTAB 23 : HTAB 12 : PRINT
       " > ZYPHYR WARS! <" ;: NORMAL :
       VTAB 24 : HTAB 3 : PRINT
       "COPYRIGHT 1982 BY SOFTKEY
       PUBLISHING" ;
1318Ø  POKE 35, 19
1319Ø  RETURN

========== CALCULATE BLDGS ============

14ØØØ  FOR A = 1 TO 38
14Ø1Ø  HEIGHT = ( RND (1) * 2Ø) * 2
14Ø2Ø  HEIGHT% (A) = EARTH - HEIGHT
14Ø3Ø  FOR B = EARTH TO HEIGHT% (A)
       STEP  - 2
14Ø4Ø  R% =  RND (1) * 5 + 1
14Ø5Ø  SCALE = 7 : XDRAW 6 AT A * 7, B
       - 1
14Ø6Ø  SCALE = 1 : XDRAW R% AT A * 7,
       B
14Ø7Ø  POKE 6, B : POKE 7, 5 : CALL 768
14Ø9Ø  NEXT : NEXT : RETURN

============ DRAW STARS ===============

141ØØ  FOR A = Ø TO EARTH
1411Ø  HC =  - HC : HCOLOR = 3 : IF HC
       < Ø THEN  HCOLOR = 7
1412Ø  STAR =  RND (1) * 279 + 1
1413Ø  HPLOT STAR, A
1414Ø  POKE 6, A + 1 : POKE 7, 9 : CALL
       768
1415Ø  NEXT
1419Ø  RETURN

=========== CALCULATE UFOS ============

15ØØØ  FOR A = 1 TO 1Ø
15Ø1Ø  T% =  RND (1) * 37 + 1 : IF COL%
       (T%) > Ø THEN 15Ø1Ø
15Ø2Ø  XUFO% (A) = T%
15Ø3Ø  YUFO% (A) = A * 8 + 12
15Ø4Ø  HCOLOR = 3 : Z = A : GOSUB 2ØØ
15Ø5Ø  COL% (T%) = A
15Ø6Ø  NEXT
15Ø9Ø  GOSUB 36Ø : RETURN
```



```
<2>          -1ØØØ4Ø          (3)
          > ZYPHYR WARS <
```

HGR : huey's hi-res corner

by jack hewitt

Last issue's column presented to you my program 'Artist's Easel' as an introduction to the world of Apple Computer high resolution graphics. This column will cover how to access the graphics screen from the monitor and from BASIC programs.

2 Hi-Res Screens

The Apple has two hi-res screens, each of which occupies 8K of RAM. Screen 1 resides from $2000-$3FFF (8192-16383) immediately followed by screen 2 at $4000-$5FFF (16384-24575). These two areas do not become active or displayed until certain 'soft' switches are thrown. These switches are special memory locations which, when referenced, activate the hardware necessary to view these screens. Table 1 shows the locations of the switches and their functions.

set	hex	decimal	function
1	C050	-16304	display graphics
	C051	-16303	display text
2	C052	-16302	full screen
	C053	-16301	mixed text/graphics
3	C054	-16300	display page 1
	C055	-16299	display page 2
4	C056	-16298	display lo-res
	C057	-16297	display hi-res

Apple's normal configuration, which exists on power-up or after a Reset, is:

text mode,
page 1,
mixed text,
and lo-res switches 'ON'

(Since the text switch is 'ON' the lo-res screen is not displayed).

Each pair of switches is a complementary set. That is, when one is switched ON the other is switched OFF, therefore only one instruction is necessary to change configurations.

BASIC POKEs

To activate a switch from BASIC requires a simple POKE.

A POKE -16304,0 (POKEing any value will suffice) will change from text to graphics - but exactly which graphics mode depends on which switch from set 4 is ON.

From the monitor it is even easier; just type the hex value of the switch and hit Return. The monitor will return a value (meaningless) and the switch will be thrown.

Try this example.

Soft Switches

Power up and hit Reset. If not in the monitor (asterisk prompt '*'), enter CALL -151.

Now type C050 and hit Return.

You are now in graphics mode (lo-res-mixed-page 1).

Now type each of the following instructions and observe the changes:

```
C057  return     --  Hi-res page 1
C055  return     --  Hi-res page 2
C052  return     --  Full screen
C051  return     --  Text page 2
C054  return     --  Text page 1
```

Notice how you went from hi-res to text and from page 1 to page 2 (page 2 text does not show keyboard entry).

From The Monitor

Now enter this short program from the monitor:

```
0300:8D 50 C0 8D 52 C0 8D 57 C0 8D 54
     C0 8D 55 C0 20 09 03
```

Now type:

```
300L return
```

You should see:

```
0300: 8D 50 C0  STA C050
0303: 8D 52 C0  STA C052
0306: 8D 57 C0  STA C057
0309: 8D 54 C0  STA C054
030C: 8D 55 C0  STA C055
030F: 20 09 03  JSR 0309
```

This will set graphics, full screen, hi-res, page 1, page 2 then loop continuously, alternately displaying page 1 and page 2.

Now type:

```
300G  Return
```

To stop it, hit Reset.

Notice that only a simple STA instruction is necessary to access each switch.

Program 1

Now from BASIC enter and RUN the following program. This does the same as the machine language program but at BASIC's slower speed.

```
100 POKE -16304,0
200 POKE -16302,0
300 POKE -16297,0
400 POKE -16300,0
500 POKE -16299,0
600 GOTO 400
```

HGR To HGR2

Here is what occurs when a command of HGR is entered from Applesoft:

a. The switches at C050, C053, C054, C057 are thrown;

b. A routine which places all zeroes from $2000-$3FFF is executed which clears the screen to all black;

c. And a $20 (32) is stored in memory location $E6 (230) which tells the drawing routines that page 1 is to be drawn on.

A HGR2 throws the fullscreen switch C052 and page 2 switch C055 and places a $40(64) at $E6 (230) as well as clearing the screen.

Because HGR and HGR2 clear the screen to black, these commands are generally used to first initiate a graphics program.

However, it is often desirable to load a picture before displaying the screen so as to avoid the delay encountered in clearing the screen and loading a picture from the disk while the screen is displayed. This is where it is necessary to utilize the screen switches.

Lines 100-400 of program 1 will accomplish this without erasing the screen.

Another example is shown in the following two programs which demonstrate different ways of animating a moving figure.

Program 2 is a short, fast routine which suffers from the characteristic flicker

continued

HGR

continued from page 43

resulting from single-screen animation. The other one is much more complicated but results in smoother motion. It also shows an alternate way of turning on the screens which does not show the screens until after they are cleared.

Examine these programs and see how the various methods of screen switching have been employed and how you can use them in your programs. Next issue I will present some methods of screen manipulation including some assembly routines which can be used to combine several pictures or to change colors or even turn pictures upside down.

Until then, happy drawing!

Program 2

```
100  REM  SIMPLE ANIMATION
     ON ONE SCREEN
150  HGR : POKE - 16302, 0 : REM
     CLEAR SCREEN AND SET FULL SCREEN
200  GOSUB 550 : REM  ENTER SHAPE TABLE
250  FOR XC = 0 TO 268 STEP 2
300  XDRAW 1 AT XC, 100 : REM  DRAW CAR
350  XDRAW 1 AT XC, 100 : REM  ERASE
     CAR
400  NEXT
450  END
500  REM  FOLLOWING DATA IS A SHAPE
     TABLE WHICH IS POKED INTO MEMORY
     AT $0300
550  DATA  1,0,4,0,45,45,54,54,45,45,
     54,62,55,62,39,60,63,63,54,63,36,
     63,36,36,37,37,37,45,0
600  FOR X = 768 TO 796 : READ SV :
     POKE X, SV : NEXT
650  POKE 232, 0 : POKE 233, 3 : REM
     ENTER SHAPE TABLE START ADDRESS
700  SCALE = 1 : ROT = 0
750  RETURN
```

Program 3

```
100  REM  MORE COMPLEX ANIMATION
     USING TWO SCREENS
150  POKE 230, 32 : CALL 62450 : POKE
     230, 64 : CALL 62450 : POKE
     - 16304, 0 : POKE - 16297, 0 :
     POKE - 16302, 0 : POKE - 16300, 0
     : REM  CLEAR BOTH SCREENS BEFORE
     DISPLAYING THEM
200  GOSUB 950 : REM  ENTER SHAPE TABLE
250  FOR XC = 0 TO 266 STEP 4
300  HCOLOR = 3 : REM  WHITE
350  POKE 230, 32 : DRAW 1 AT XC, 100 :
     REM  DRAW ON SCREEN 1
400  POKE 230, 64 : DRAW 1 AT XC + 2,
     100 : REM  DRAW ON PAGE 2
450  POKE - 16299, 0 : REM  DISPLAY
     PAGE 2
500  HCOLOR = 0
550  POKE 230, 32 : DRAW 1 AT XC, 100 :
     REM  ERASE PAGE 1
600  HCOLOR = 3
650  DRAW 1 AT XC + 4, 100 : REM  NEW
     PAGE 1
700  POKE - 16300, 0 : REM  DISPLAY
     PAGE 1
750  HCOLOR = 0
800  POKE 230, 64 : DRAW 1 AT XC + 2,
     100 : REM  ERASE PAGE 2
850  NEXT
900  END
950  DATA  1,0,4,0,45,45,54,54,45,45,
     54,62,55,62,39,60,63,63,54,63,36,
     63,36,36,37,37,37,45,0
1000  FOR X = 768 TO 796 : READ SV :
      POKE X, SV : NEXT
1050  POKE 232, 0 : POKE 233, 3 : REM
      ENTER SHAPE TABLE START ADDRESS
1100  SCALE = 1 : ROT = 0
1150  RETURN
```

REBUTTALS

continued from page 10

when a computer owner copies a disk and passes it out to all the members of his family? Or to his best friend? Or to his best friend's best friend?

Also, where does a corporation stand? A corporation, whether it be an incorporated user's group or an actual business, is recognized by law as an individual. As an individual, can corporation purchase one copy of a program and make copies for all of its employees or members? Isn't that within the constitutional rights of the corporation as an individual?

I am not saying that we should go from one extreme or the other. All I am trying to point out is that we should not be so blinded by our own selfish views that we fail to see the legitimate concerns of the 'other side.' This applies to all concerned.

If we attempt to look at it logically and really be fair, it seems that we could come up with a rather equitable solution for all concerned. If we adopt an attitude of 'damn the companies' or turn our noses up at users, we only hurt ourselves. Perhaps it is time to take BOTH sides into account.

As an afterthought, how would you feel about a company photocopying HARDCORE for all of its employees, thus depriving you of many possible subscriptions? The analogy is about the same, even if the media is different. I would definitely be interested in an answer.

Sincerely,
Allen L Wyatt
Advanced Operating Systems
Michigan City, IN

INPUT

continued from page 25

an "uncrackable" program.

I would like to take some time to join you on the soap box. There is a significant difference between copying a program disk, and "breaking" the disk.

To copy a disk, one only has to figure a way to recreate the same pattern of bits on the new disk that was on the old.

To "break" a disk is another matter entirely. This refers to removing the nonstandard encoding, and "normalizing" the programs.

A normalized program can be run with normal DOS. A normalized program can be modified (and corrected if need be). A normalized program can be integrated easily into a system.

The procedure to normalize is far more complex than a simple copy.

Copying in some manner will always be possible, and some software manufacturers have learned to live with it. The point of this is embodied in a program called ASCII Express. This program is heavily protected and yet is copyable, within limits. The copy program is on board, and so is a copy counter. The user may make up to 4 copies of the original and thus have 4 backups. Had every programmer of commercial software been so enlightened, Locksmith, Nibbles Away, and other copy programs would never have been necessary. (Even Hardcore would likely not have come into existence.) The worst that can happen is that the copies of the software fall into 4 unauthorized hands.

There are some users who get considerable enjoyment from just "breaking" a program. They usually don't do it to deliberately pirate programs; it is simply an intellectual exercise and sometimes an ego trip. They seldom use commercially available copy software, preferring to write their own. They are thus constantly adding to their own knowledge of computing, and through their writings, they add to others. These people are a minority, and probably always will be, but they do exemplify the meaning of the term "Personal Computer."

I try to avoid pirated audio recordings, and pirated video, and if I pay for a program, I would feel like a fool to give it away to someone else, so I don't.

I have on occasion though, borrowed (and lent) software.

I have an extensive fiction library (several hundred volumes) and have never hesitated to lend a book from it to someone else, nor have I hesitated to borrow one. By the strict letter of the law, I am in violation of copyright law each time I either lend or borrow a book. If anyone thinks that I am going to stop borrowing or lending books (or programs), they couldn't be more wrong! If that makes me a pirate, so be it.

I sympathize with the programmer who has spent hundreds of hours creating a piece of

continued on page 63

Vector Graphics part I: shape tables

Fun and Facts Concerning DRAW and XDRAW

Bev R. Haight

The Applesoft graphics commands are familiar to most of you. In this column I'll be covering the least-used of those commands: those that manipulate Applesoft "vector graphics".

These commands are:

1. DRAW n
2. DRAW n AT x, y
3. XDRAW n
4. XDRAW n at x, y
5. SCALE= n
6. ROT= n

We'll also be using those commands that are familiar to you:

7. HCOLOR= n
8. HGR and HGR2
9. HPLOT x, y
10. HPLOT x1, y1 TO x2, y2 TO x3, y3
11. HPLOT TO x3, y3

And we'll be POKEing the following addresses (with zero). These are the locations of the Apple SCREEN SOFT SWITCHES and include:

(switch)	(address)
1. Hi-Res display	**-16297 or 49239**
2. Lo-Res display	**-16298 or 49238**
3. Page 2 display	**-16299 or 49237**
4. Page 1 display	**-16300 or 49236**
5. Mixed screen	**-16301 or 49235**
6. Full screen	**-16302 or 49234**
7. Text display	**-16303 or 49233**
8. Graphics display	**-16304 or 49232**

In order to use a shape table, we'll also be POKEing the address of the start of the shape table in location 232 and 233 (pointer to the shape table index). We might also be setting LOMEN or HIMEM. Furthermore, to make the programs more aurally interesting, we'll also be PEEKing the speaker (-16336 or 492ØØ).

And in later articles, we'll also be using these addresses:

decimal	(hex)	(purpose)
234	**($EA)**	**Collision counter**
228	**($E4)**	**Hi-Res color byte**
226	**($E2)**	**Hi-Res Y coordinate**
225	**($EØ)**	**Hi-Res X coordinate**

Before we can use the DRAW and XDRAW commands, however, it will be necessary to first make a shape for the commands to DRAW. Neglecting this step will have interesting effects especially if, when it tries to DRAW on one of the Hi-Res pages, your program also resides there. In fact, your system may even "hang" if it DRAWS over page one of memory and over your DOS.

A shape table consists of a shape index and the various shape definitions. The shape table index lists the number of shapes in the table and then consecutively lists how far the starting address of each shape definition is from the beginning of the table. Each shape definition consists of a list of vectors (either plot or non-plotting). It might help to have your Applesoft Reference manual open to page 92 while I run through the preliminaries.

Vectors are simply directions of travel. In this case, we have only 4 vectors or directions:

(direction)	(definition)
1. go UP	**ØØ**
2. go RIGHT	**Ø1**
3. go DOWN	**1Ø**
4. go DOWN	**11**

The definitions are the bits (remember, 8 bits make up a byte?) that tell the DRAW and XDRAW routines which direction to move. To this definition we must add another bit to tell the routine whether or not to plot:

1 PLOT
Ø DON'T PLOT

These three bits are then combined to fill out a byte. But since a byte has only 8 bits, we are left with a problem: three sets of three bits make 9 total. There's one bit too many!

Here's a sample shape definition byte: 1Ø111Ø1Ø

The DRAW routine reads this byte from right to left in this manner:

1Ø ← 1 11 ← Ø 1Ø ←

These are the three "steps" it takes when it reads a byte.

1. It first sees "Ø1Ø" as "Ø" and "1Ø" which means to move down without plotting. (1Ø = go down, and Ø = not to plot)

2. It then reads '111' as '11' and '1' which means to move left while plotting. (1 = plot, and 11 = move left)

3. Finally, it sees '1Ø' which it interprets as 'Ø' (implied by the absence of that mythical 9th bit) and '1Ø' which together means to move down again, without plotting.

As you can see, the third step can never be a plotting vector. That means that if you did want to plot while moving down, you'd have to put 'ØØ' in its place and go to the next byte. The first step in that next byte would then be '11Ø.'

Remember, zeros are important, but zeros in certain instances will be ignored, or worse, translated as the 'End of shape.' For example, whenever the DRAW/XDRAW routine reads a pair of zeros in step 3, it ignores them and proceeds to the next byte. But it will also ignore a set of three zeros in step 2 if a pair of zeros occupy step 3. Now that situation poses some problems.

Some Problems

1. According to the definition table, 'ØØ' means to 'go up.' But if that vector is placed in step 3, it will be ignored! So if one must go up, it will be necessary to place that 'ØØ' in step 1 of the next byte.

2. Also according to the definition table, 'ØØØ' means to 'go up,' too, without plotting. But if that vector is placed in step 2 and is followed by a 'ØØ' in step 3, it will also be ignored. Again, one must place the 'ØØØ' in step 1 of the next byte.

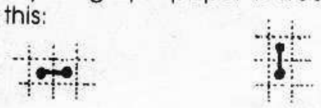
3. A byte consisting of all zeros means 'End of shape,' so one cannot end with a non-plotting vector going up. Nor can one have a series of non-plotting vectors going up. So how does one vector up a distance without plotting if it is translated as 'that's all folks!.' That will be one of the problems solved next time when I delve into more complex shapes using a 'shape-table maker' program. Meanwhile, think about it.

Without that program to help make the 'shapes,' you'll have to do it the hard way. But, interestingly enough, the hard way is the best way to learn about shape tables in general.

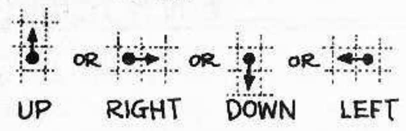
Have some graph paper handy (with 4 to 5 squares to the inch). The shapes will be mercifully simple: a plotting dash, and a plotting square. The first shape will be a dash . . .

continued on next page

On your graph paper it would look like this:



We'll adopt your manual's symbols and write it like this:



As you can see, there are 4 ways to draw the dash vectors: the 4 directions. We'll do it all 4 ways and translate it into the actual vector codes representing it, and the actual shape definition byte.

Vector:	UP	RIGHT	DOWN	LEFT
Code:	1ØØ	1Ø1	11Ø	111
Byte:	ØØØØØ1ØØ	ØØØØØ1Ø1	ØØØØØ11Ø	ØØØØØ111
Hex:	$ Ø4	$ Ø5	$ Ø6	$ Ø7
Decimal	4	5	6	7

We arrived at the shape definition byte by filling in steps 2 and 3 with zeros. (The hex translation is provided in case you want to use the monitor). Its decimal equivalent is provided for those who want to POKE in the values from Applesoft. Use the 'NIBBLE/HEX' table when translating the byte into its hex equivalent. To get the decimal equivalent, use the BYTE CONVERSION CHART in Update 2.1, your own hex-decimal conversion program or, with some work on your part, the table on page 15 of the Apple II Reference Manual.

CONVERSION CHART

NYBBLE-HEX-DECIMAL

ØØØØ	$Ø	Ø
ØØØ1	$1	1
ØØ1Ø	$2	2
ØØ11	$3	3
Ø1ØØ	$4	4
Ø1Ø1	$5	5
Ø11Ø	$6	6
Ø111	$7	7
1ØØØ	$8	8
1ØØ1	$9	9
1Ø1Ø	$A	1Ø
1Ø11	$B	11
11ØØ	$C	12
11Ø1	$D	13
111Ø	$E	14
1111	#F	15

Vector Codes

--- NON-PLOTTING ----

Vector	Code
UP	ØØØ
RIGHT	ØØ1 OR Ø1
DOWN	Ø1Ø OR 1Ø
LEFT	Ø11 OR 11

------- PLOTTING ----

Vector	Code
UP	1ØØ
RIGHT	1Ø1
DOWN	11Ø
LEFT	111

There! You now have a shape definition. But to use it, we'll have to put it into a shape table. Because we are using small shapes of relatively few bytes, let's put our shape table on page three of memory. We first have to make a shape table index consisting of the total number of shapes and then, consecutively, a list of each shape definition's starting position relative to this 'start of table.' In this example, there is only one shape. Reading down the column, you will see a sample shape table:

hex	decimal	explanation of byte
Ø1	1	total number of
ØØ	Ø	shapes
Ø4	4	index to first byte of
ØØ	Ø	shape #1
Ø4	4	shape definition #1 **
ØØ	Ø	end of shape

(** This is the dash going up. You can put any of the four dashes in this position.)

There are two ways to enter this table:
1. Use the Monitor.
2. POKE it in directly or via a program.

The memory location we'll use for our experimental shape table is $3ØØ hex, 768 decimal.

To use the monitor:
1. Type: **CALL -151**
2. Type: **300:01 00 04 00 04 00 Return**
3. Press **Reset** to return to Applesoft (autostart ROM only)

In this example, we'll POKE in the table from inside a program. In this way, you'll immediately see if you have made an error in translating your shape vectors into the proper byte code.

```
10 REM FUN WITH SIMPLE SHAPES
100 POKE 232,Ø : POKE 233,3
110 DATA 1,Ø,4,Ø
120 DATA 4
130 DATA Ø,-1
140 ADDRESS = 768
200 READ A
210 IF AK Ø THEN 250
220 POKE ADDRESS, A
230 ADDRESS = ADDRESS +      1
240 GOTO 200
250 HGR
270 HCOLOR = 3
280 HPLOT Ø, Ø
290 CALL 62454
500 FOR A =     1 TO 20
510 SCALE = A * 2
540 ROT = Ø
550 XDRAW 1 AT 10 * A, 80
580 NEXT A
```

In line 1ØØ, the pointer to the start of the shape table was POKEd. Although the decimal location is 768, we must poke two locations with the double byte equivalent of the decimal number of 768. This is easy to do since we know that the hexadecimal location is $3ØØ. Breaking it into two bytes, we have $3 as the upper byte and $ØØ as the lo-byte. That easily translates to decimal equivalents of 3 (hi-byte) and Ø (lo-byte). Addresses are stored in 'reverse order,' so we put the lo-byte in location 232 and the hi-byte at 233.

Lines 11Ø - 13Ø is our shape table (and line 12Ø is our lonely shape definition) conveniently stored in DATA statements. As we go through this exercise, you should make changes in line 12Ø and RUN the program whenever you want to change the shape definition.

If you RUN this program, you'll get 2Ø images of your dash oriented just the way you originally specified the vector . . . in 25 sizes or SCALEs.

You can now experiment with the shapes. Plug in a 5, 6 or 7 in place of the '4' in line 12Ø. As you can see, they are all dashes, but pointing in the four directions, pure vectors . . .

To see what a simple dash can do, add in these lines:

```
520 FOR B = 1 TO 64
540 ROT = B
550 XDRAW 1 AT 139, 79
570 NEXT
```

Now you have a rotating dash that, after one full revolution, gets a little larger and rotates again, and so on.

Make this change and you'll mysteriously get vanishing lines:

540 ROT = B * 2

If you like, try this change:

550 HCOLOR = RND (1) * 7 + 1
560 DRAW 1 AT 139, 79

Now the color will be solid, and colorful...but may sometimes be black.

Your dash can also go crazy and dash all over the place. For a crazy dash, make these changes:

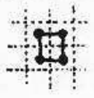
```
400 SC = RND (1) * 20 + 1 : SCALE = SC
410 RO #RND (1) * 64 + 1 : ROT = RO
420 XDRAW 1
450 GOTO 400
```

Now RUN it. The first thing you should notice is that when the dash runs off to one side, it reappears on the other. By purposely not specifying the X and Y locations, the XDRAW command will tack one dash onto the tail of the other, making worm tracks all over the screen.

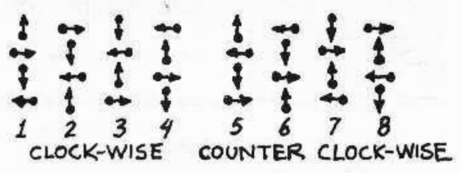
For your own amusement, you can add noise:

```
430 NOISE = PEEK (-16336)
```

So much for dashing dashes . . . Now on to another simple shape: an off-centered square. The smallest such square looks like this on your graph paper:



It is called an 'off-centered' square because one begins drawing it's vectors at a corner and not at the center. This imparts certain characteristics to this square. The square can be drawn using all four different vectors. Like the dash, we can begin at any of the four vectors and end up with a square as long as we use them in the proper order. The four different squares are:



Let's take the first square and change it into a shape table:

(code)	(byte)	hex	decimal
100	0000 0100	$04	4
101	0000 0101	$05	5
110	0000 0110	$06	6
111	0000 0111	$07	7

From this, it appears that the shape definition of a square consists of four bytes:

4567

But we only use a third of each byte to indicate a vector. Let's use the entire byte and conserve memory:

00 + 101 + 100 = 00101100 = $2C = 44
00 + 111 + 110 = 00111110 = $3E = 62

Remember, we fill in the byte from right to left and, since all four vectors are plotting, we've had to leave step 3 empty (00). Now our shape definition takes up only two bytes. And believe me, when you have a lot of shapes to store, every byte you save leaves more room for something else (program space, graphics, etc.).

Make this change in your program:

```
120 DATA 44,62
300 GOTO 500
```

your program will now rotate the square instead of the dash. Here's an expanding square (if you make this change):

```
300 FOR A = 1 TO 50
310 ROT = 0
320 SCALE = A
330 XDRAW 1 AT 179, 80
340 NEXT
390 END
```

Try making this change:

```
310 ROT = A
```

Or this:

```
310 ROT = A * 2
```

Table With 2 Shapes

In order to use both the dash and the square, your shape table must be modified to include both definitions and both indices.

table values	bytes away from start	explanation
2	0	number of shapes
0	1	in this table
6	2	number of bytes to
0	3	the first shape
8	4	number of bytes to
0	5	the second shape
* 4	6	first shape
0	7	end of first shape
28	8	second shape
38	9	
0	10	end of table

If you examine the column titled 'Bytes away from start' you will note that the beginning of a shape definition corresponds to the value place in its index. For shape 1 that number is 6, and for shape 2 that number is 8. When this table grows to include a great many shapes (maximum is 255), all the other shape indices are calculated in a similar manner. Of course, the start of the first shape must move as the list of indices grows.

As you have seen, there are many ways to 'make' any particular shape depending upon how you intend to use the shape. This will be the topic of the next installment.

FINANCIAL MANAGEMENT SYSTEM II

A HOME AND BUSINESS ACCOUNTING SYSTEM
FAST – EASY TO USE

Enter an entire month's CHECKING, CHARGE CARD, and CASH accounts in just a few minutes using your own personalized macro lists. INSTANT ERROR CORRECTION. Audit all files by Code and Month with year-to-date totals. PERFECT FOR TAX ACCOUNTING.

- SELF PROMPTING, ERROR AVOIDING ENTRY SYSTEM. 1 to 3 KEYSTROKE entries with AUTOMATIC DATE, CODING and NUMBERING.
- SCREEN and PRINTER routines for listing files, balance, reconcile, search, and audit reports. Configure system to match almost any printer.
- BUDGET MANAGER to plan, review and balance your budget, generate COMPLETE reports for any 1 – 12 month period.
- SEARCH all files with specific and expanded searches.
- ACCOUNT AUDITOR totals all files by Code and any 1 – 12 month period with year-to-date totals.
- Enter your own Item and Code Macros, up to 100 each.
- 48K with ROM APPLESOFT and DISK required. SINGLE or DUAL DISK compatible. (printer optional)
- **PRICE $39.95**

*******NOW AVAILABLE*******

CHECK WRITER II for FINANCIAL MANAGEMENT SYSTEM II. Print your checks directly from FMS II files.

- **PRICE $19.95 – ($15.00 if ordered with FMS II)**

GROCERY LIST

A USEFUL HOUSEHOLD PROGRAM DESIGNED TO ORGANIZE SUPERMARKET SHOPPING

Shoppers will INSTANTLY be able to use this easy, self-prompting program. Scan a file of up to 500 USER DEFINED ITEMS. Choose those needed with a single keystroke. Then print a shopping list ORGANIZED BY TABLE NUMBER, SECTION, or four letter code such as "DARY", "BAKE", or "DELI".

- 32 – 48K with disk and printer required, (APPLESOFT)
- **PRICE $19.95**

COMPUTERIZED MANAGEMENT SYSTEMS

by D R JARVIS COMPUTING INC.
1039 Cadiz Dr. - Simi, CA 93065
PHONE (805) 526-0151

Check, VISA or MASTER CARD accepted.
DEALER INQUIRIES INVITED

SOFTWARE WRITER'S MARKET

HOW TO MARKET YOUR SOFTWARE

BY BRH

Three publishers of software are examined in this column:

1. HAYDEN BOOK COMPANY

They requested inclusion in this column and sent a sample contract.

2. BRODERBUND SOFTWARE

A well-done Author's kit was supplied upon request and included a sample contract, an illustrated booklet revealing their products, introductory letter explaining 10 reasons why you should join their team, sample game documentation, and a personal letter explaining that they are very flexible with their royalty policy which is 20%-25% on games, more for business program, may include outright purchases, flat price per copy sold, and even advances.

3. INSTANT SOFTWARE

Their submission package included a sample contract and literature on submission procedure, style guides, commonly asked questions, and even hints on where to get ideas for new software packages.

Because most software houses have similar contracts, I will try to contrast these three under the categories of:

1. Rights sold
2. Royalty
3. Contract time
4. Program Submission

Copyrights

All three purchase all and exclusive rights (both copyrights and intellectual rights to your program(s). That means that the publisher owns that program and any program you write that is similar enough to it to qualify as infringement of those rights. The publisher also gets all versions, updates, etc. of that program.

Royal $$

Royalty is usually based upon a percentage of the cash received for the sale, lease, etc. of the program minus cost of returns, allowances, discounts, collection fees, transportation and postage, taxes, etc. though these vary from contract to contract. Royalty can also be a stated amount per program actually sold regardless of product price or gross cash receipts. Of course, when more than one program makes up a product, royalty must be divided up among the authors.

Contracts

Some publishers require a signed contract before examining your program, others after evaluation and acceptance. Some take an awful long time to decide, others will tell you within a few days whether they want your program or not.

Also some publishers require you to make revisions and alterations, or they will get someone else to do it for you and pay him out of the royalty that you would have gotten.

Contracts are a part of business, so

HAYDEN BOOK COMPANY
50 Essex Street
Rochelle Park, NJ 07662

—Rights sold:
All rights including exclusive rights and subsidiary rights. Also first options on all versions for any system and all future versions.
—Royalty:
AMOUNT NOT STATED, based on actual cash received by Publisher and paid quarterly to author.
—Contract time:
NOT STATED, however, if the program is not kept "in print" and "for sale" for any two year period, then 90 days after written demand from author to reprint the program, the contract is terminated and all rights revert back to author.
—Program submission:
Author supplies Publisher with three copies of program within 2 weeks of signing contract, along with necessary documentation and source code. Publisher has 12 weeks to determine publication acceptability.

BRODERBUND SOFTWARE
2 Vista Wood Way
San Rafael, CA 94901

—Rights sold:
Exclusive rights.
—Royalty:
NOT STATED, a flat royalty per copy of work sold regardless of selling price of copy and paid monthly to author (according to the sample contract).
—Contract time:
Ten years; or 6 months after either publisher or author gives written notice of termination.
—Program submission:
Send them a copy of your program by Registered Mail. There is a brief two day evaluation period. Please include necessary documentation, your address and phone.
—Other:
Author must make revision at own expense, and failing to do so, publisher may have another party make the revisions and charge the cost against the author's royalties.

INSTANT SOFTWARE INC
Peterborough, NH 03458

—Rights sold:
All rights, exclusive rights, and options on derivative works and versions.
—Royalty:
20% of gross cash receipts, paid quarterly to author.
—Contract time:
NOT STATED, but if the publisher does not publish, distribute and sell the program within 1 year after signing contract, then author need only give written notice to terminate the contract.
—Program submission:
You must sign the contract first, then send a copy of the program with contract and documentation. The evaluation period is up to 12 weeks (filled in a blank space in contract).
—Other:
If author does not make changes as specified by publisher, then publisher can get someone else to do it and deduct the cost from the author's royalties.

examine yours carefully. Get a lawyer to explain what the terminology really means. And if you have complaints or praises as a result of your experiences with various publishers, let me know. This column is for you, the software writer. Your experiences can help others make decisions regarding who shall, and how to, market their programs.

All three state in their contracts that if the author is accused of copyright infringement, it is up to the author to defend himself, and if the publisher is harmed, then the author must indemnify the publisher for that loss.

Dealers

One of the best ways to see how your program will be marketed by the publisher is to see how they have treated others. Examine the ads in magazines, check out your local computer outlet, and watch this column space. Ads in magazines sells your program, and interests others to buy them at their dealer outlets. Dealers can show off your program directly to customers, so it is important that publishers have many dealer outlets as well.

According to Instant Software, "our authors earn an average of 3 times as much from wholesale orders we get from our dealers." Broderbund boasts over 700 dealers.

Advance $

Another item worth investigating is the subject of advances, or prepayment of royalties before the program is marketed. In this way, the author gets financial support ($$$) and encouragement (because $$$ up front means that the publisher is certain of making a profit from your program.) Of the three, only Broderbund, in its author's kit, comes out and says, "Advances are standard for most new products."

However, before sending your program off to any publisher, look around, ask for their submission packages and sample contracts, compare and contrast them, check with a lawyer, and then choose.

FOCUS: AAA

continued from page 16

computer stores use my library, but for another reason. They give away the volumes to Apple buyers to help stimulate their interest."

"My goal for AAA, however incomplete, is to continue to provide Apple owners with not only public domain software, but an entire range of commercial software and accessories at very low prices."

For more information about the AAA, call Ron Maleika at (307) 632-8561 or write to the Apple Avocation Alliance, Inc., 721 Pike Street, Cheyenne, Wy 82009.

(Editor's note: I have not actually used any of the programs from Ron's library and cannot make judgments on the quality of the programs. From reading some of the literature, I garner that some programs need work and are listed as such. The quality of the public domain library continues to increase as subscribers write in with fixes and completions to such programs. I am impressed by the quantity. Let me know about the quality!)

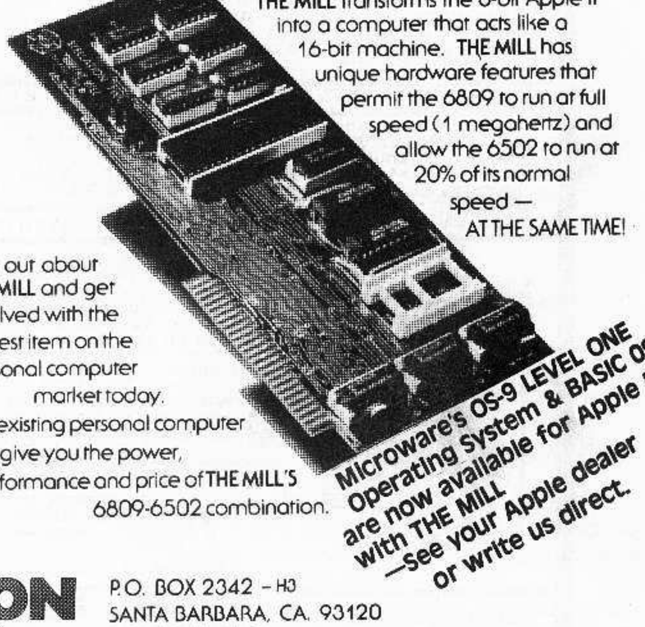
Your Apple too slow? Not anymore...

Now you too can write 6809 programs for your Apple II that are DOS 3.3 compatible. But you don't have to stop there, you can also program your Apple II's 6502 and the 6809 of THE MILL to run SIMULTANEOUSLY.

THE ASSEMBLER DEVELOPMENT KIT, including THE MILL, is a full feature assembler, designed to use the text editing system of your choice. The system will also boost your computer programming productivity, since the 6809 is today's easy to learn and program computer. Take advantage of the 8-bit 6502 and the 16-bit abilities of the 6809 running at the same time, create your own MULTIPROCESSING ENVIRONMENT on the Apple II.

Put THE MILL into your Apple II and get power, performance and price matched by no other personal computer. STELLATION TWO offers a full 1 year warranty and a 60 day money back guarantee, if you're not completely satisfied with the power of THE MILL.

Shop around, then compare the service, quality, price and power of THE MILL. Take this ad to your local Apple Dealer and see the endless possibilities of adding THE MILL to your Apple II.



THE MILL transforms the 8-bit Apple II into a computer that acts like a 16-bit machine. THE MILL has unique hardware features that permit the 6809 to run at full speed (1 megahertz) and allow the 6502 to run at 20% of its normal speed — AT THE SAME TIME!

Find out about THE MILL and get involved with the hottest item on the personal computer market today.

No existing personal computer can give you the power, performance and price of THE MILL'S 6809-6502 combination.

Microware's OS-9 LEVEL ONE Operating System & BASIC 09 are now available for Apple II's with THE MILL —See your Apple dealer or write us direct.

STELLATION TWO

P.O. BOX 2342 – H3
SANTA BARBARA, CA. 93120
(805) 966-1140

Apple is a trademark of Apple Computer, Inc.

continued from page 18

2 were too unreasonable to play. To this I say 'Hogwash'! Now don't get upset. The only reason that I can say this is because I have just recently obtained maps of both. So now you can say 'Hogwash,' too!

ADVENTURE IN TIME

Adventure In Time is rated by Phoenix as a class 4 adventure. I don't know what they base this rating on, but I found the adventure to be relatively easy. I also found it to be rather entertaining, and on that note I do recommend it. A map is available for that program too. As for a couple of hints, don't stay in any time period too long. You may find yourself sticking around longer than you had planned. The entire building that the time machine is in will go with you in time. With that in mind, everything you find can be stored in the building quite safely. The picture on the wall is worth checking out.

TIME ZONE

Time Zone looks to be an interesting program, aside from a couple of bugs. It seems that the 'Backup Program Disks' option in the main menu does not function properly. Another bug lies in Tokyo in the year 2Ø82 A.D. It seems that there is an error in the copy of the program that I viewed. However the possibility exists that the version (which I played the week it came out) that I have may just have been a lemon. I would be interested in any input concerning problems or just plain comments concerning the program in general.

On the bright side, I found Time Zone to be an interesting prospect within the realm of adventuring, but only time (no pun intended) will determine whether or not the program justifies the $1ØØ (hak kaf) price tag. At any rate, here's a couple of clues: In London in 2Ø82 A.D. it's a good idea to leave behind anything you are carrying by the time machine. After you have mapped out this section of the game, the effect of your actions will be very obvious. Be careful of what you are carrying. If you go into a time period that the particular item(s) cannot be carried due to technological limitations, you will lose them!

ULYSSES AND THE GOLDEN FLEECE

Ulysses and the Golden Fleece has proved to be a fair challenge. Even though bribing public officials is generally frowned upon, you won't get arrested for trying it.

IS IT ADVENTURE?

One issue I would like to touch on is 'how adventures should be classed' with respect to this column. With such programs as 'Castle Wolfenstien,' 'Crush, Crumble and Chomp,' 'Dark Forest,' 'Ruski Duck' and the like, it is becoming more and more difficult to determine exactly what programs to cover in this column. To get to the point, what I need is some input from my readers. Give me an idea of what sort of software you would like to see covered. (Unless otherwise requested, your letter may be printed in the next issue of HARDCORE.

In closing I would like to leave you with a little food for thought:

TREBOR SUX!

Until next time,

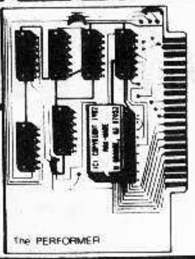
Mike Flynn

HARDWARE

APPLE

201 - 839-3478

Dealer and Distributor Inquiries Invited.

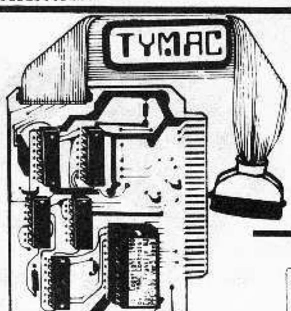


THE PERFORMER PRINTER FORMATTER BOARD for Epson, OKI, NEC 8023, CITOH 8510 provides resident screen dump and print formatting in firmware. Plugs into Apple slot and easily accessed through PR# command — Use with standard printer cards. $49.00 specify printer.

THE MIRROR FIRMWARE FOR NOVATION APPLE CAT II® The Data Communication Handler ROM Emulates syntax of another popular Apple Modem product with improvements. Plugs directly on Apple CAT II Board. Supports Videx and Smarterm 80 column cards, touch tone and rotary dial, remote terminal, voice toggle, easy printer access and much more.

List $39.00 — Introductory Price $29.00

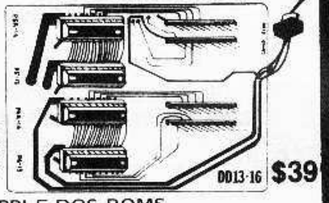
PARALLEL PRINTER CARD



A Universal Centronics type parallel printer board complete with cable and connector. This unique board allows you to turn on and off the high bit so that you can access additional features in many printers. Use with EPSON, C.ITOH, ANADEX, STARWRITER, NEC, OKI and other with standard Centronics configuration.

$139.00

DOUBLE DOS Plus



$39

DOUBLE DOS Plus — a piggyback board that plugs into the disk-controller card so that you can switch select between DOS 3.2 and DOS 3.3

DOUBLE DOS Plus requires APPLE DOS ROMS

SOFTWARE

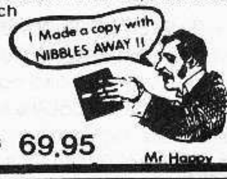
201 - 839-3478

APPLE 48K

NIBBLES AWAY II

AGAIN! Ahead of all others.

- **AUTO-LOAD PARAMETERS** . . . Free's the user from having to Manually Key in Param values used with the more popular software packages available for the Apple II.
- **EXPANDED USER MANUAL** . . Incorporates new Tutorials for all levels of expertice; Beginners Flowchart for 'where do I begin' to 'Advanced Disk Analysis' is included.
- **TRACK/SECTOR EDITOR** . . . An all new Track/Sector Editor, including the following features: Read, Write, Insert, Delete Search, and impressive Print capabilities!
- **DISK DIAGNOSTICS** . . . Checks such things as: Drive Speed, Diskette Media Reliability, and Erasing Diskettes.
- **HIGHEST RATED**. . .Best back up Program in Softalk Poll (Rated 8.25 out of 10).
- **CONTINUAL UPDATES** . . . Available from Computer Applications and new listings on the source.



69.95

Mr. Lister Customer Contact Profiler & Mailer

A Super Mail List Plus more up to 1000 Entries on single 3.3 Disk (only 1 Drive required) 2 second access time to any name full sort capabilities Dual Index Modes supports new 9 digit Zip. Easy to follow manual Not Copy Protected 4 user defined tables with 26 sort selections per table Beta tested for 6 months user defined label generation.

INTRODUCTORY PRICE ~~$135.~~ $99.00 Dealer & Dist. Inquiries Invited.

Super PIX HIRES SCREEN DUMP ◆◆◆◆

The Software package that will allow your printer to dump page 1 or page 2 of the Apple Hires screen horizontally or vertically. Use with EPSON® MX-80 with or without GRAFTRAX® Roms, MX-70 — OKI® Microline 80, 82, 83, 82A, 83A — C. ITOH® 8510 and NEC 8023A. Requires Tymac Parallel Printer Board PPC-100, . . . $24.95

THE APPLE CARD—Two sided 100% plastic reference card Loaded with information of interest to all Apple owners $3.98

MasterCard VISA

MICRO-WARE DIST. INC. P.O. BOX 113 POMPTON PLAINS, N.J. 07444

Dealer and Distributor Inquiries Invited.

a new users' column

Using VisiCalc For: Job Costing

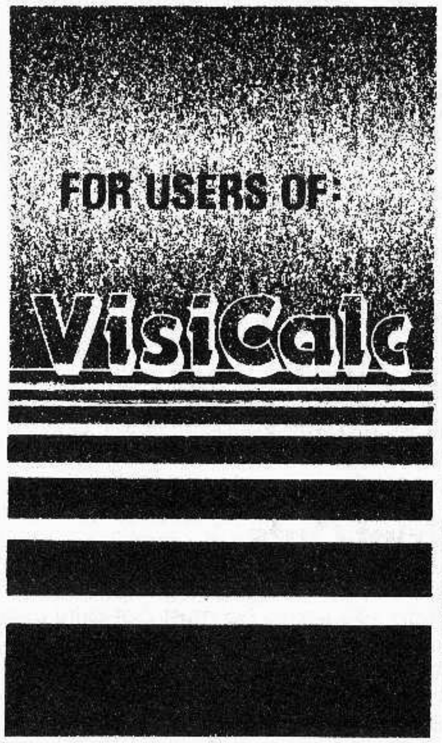
by Jerry Scott, PhD

The VISICALC program from Personal Software is the constant leader in Apple business software sales. This writer feels that every Apple owner should have and use the VisiCalc program. Ten years programming experience has taught this author that in programming, it is 'a far better thing' to learn a few important things really well. For the Apple user, VisiCalc is a very important thing.

I often tell beginning Apple owners to buy:

1. a Word Processor Program,
2. VisiCalc,
3. and a Data Base Program,

and then spend the time necessary to learn how to use these programs well. Beginners should do this even before learning to program. It takes about a month or so for the dedicated user to learn to use these three programs. After that, he really has gained some power with his Apple. If his needs still aren't met, then he will definitely understand his real needs much better. Then he might begin programming, or contact an experienced programmer to begin the process of getting the necessary software written.



Visicalc is a trademark of Visicorp

A. a 3-day introduction

This article will not attempt to fully teach someone who has never seen VisiCalc how to use it. Personal Software has provided a comprehensive 200 page manual for that purpose. (The manual is hard to follow in some places, yet is truly excellent in others. It must be worked through to learn all the 'ins and outs' of VisiCalc. Each of my columns will teach some important aspect of VisiCalc use.

Most important actions in VisiCalc begin with a COMMAND.

VisiCalc pioneered the 'type ahead buffer' for the Apple, and all commands begin with a slash (/). This means that many commands can be strung together. For example:

To globally format the screen so that all numeric entries are in 'dollars and cents', simply type:

/GF$

The **/G** means "do something globally."

The **F** means 'format'

And the **$** means 'use the dollar and cents' option of the Format command.

This command has three levels, and VisiCalc uses 'dynamic prompting' at the top of the screen to tell its users exactly what is happening.

Thus, the quickest way to learn VisiCalc is to experiment with all the commands. Most of them are nmemonic and are thus easy to remember:

/P Print
/S Storage
/I Insert
/D Delete

Here is my standard advice for quickly learning VisiCalc:

day 1: FUNCTIONS

Play with the commands for an hour or so, then get out the short reference guide (5 pages) and look at the FUNCTIONS.

These begin with the 'at' sign (@) and are also mnemonic.

@ S Sum
@ P Product
@ A Average

With these experiences behind you, study the Command Reference Chart (3-3,3-4) in the manual. And make sure that you learn the **/R** Replicate command the first night. This should end the first days work.

day 2: EDITING

On the second day you can read through several chapters of the manual. During that day, you should become familiar with the **/E** Edit command.

The **/E** command is one of the big improvements in the new 3.3 version. At the end of this study period, you should try building a simple four column application.

I recommend: CHECKBOOK BALANCING

Use these columns:

1. CHECK #
2. CHECK AMOUNT
3. DEPOSIT AMOUNT
4. BALANCE

I still use a similar format to balance my checkbook at the start of each month. Once you are really familiar with VisiCalc, such an application takes about 5 minutes to write.

day 3: PRINTING

Then on the third day, learn how to use the **/P** Print command, and how to spruce up the outputs of your worksheet.

The **/-** Repeat command is useful here.

You should also learn to use the **/W** Window and the **/T** Title commands to help improve the way the screen looks during data entry.

After the third day, you should be able to do a complicated balance sheet or

continued

budget. Surely, you can impress your friends with your checkbook application and VisiCalc's recalculation.

Once the program's commands are learned, it usually only take a few hours to set up balance sheets, budget charts, etc. The reference manual is still there, and the dynamic prompting and mnemonic nature of the commands make them easy to remember and use. These same balance sheets would take even an astute programmer several days to program in AppleSoft and much longer to completely debug and verify.

B. an estimation application

Now I will outline how to use VisiCalc to do Job Costing.

The VisiCalc worksheet has 254 rows and 63 columns. With only 64K of main memory, all of these can't be used at once. Experience has shown that at least 150 rows of 8 to 10 columns can be used.

Most companies estimate how much the material for a job will cost by looking up items in catalogs and calculating cost estimates from them. Therefore, the first step in the Estimation Process is to load the catalog items into VisiCalc worksheets of about 150 items each.

Each item must be complete with:
1. Parts numbers
2. Descriptions
3. Costs

Each item will occupy one row of the worksheet. Once the formulas for the first line are correct, use the **/R** Replicate command to put the same formulas on all the remaining 149 lines. Then SAVE the worksheet with a very clear and suggestive file name. Do the same again for each succeeding 150 items until done. Obviously, this approach will not work for someone with thousands of items or for someone whose items change daily. Instead, this application works best for the estimator who has only about 500 to 1000 relatively stable items from which to choose.

The individual lines containing the items are entered in such a way that all the job estimator has to do is to enter the number of each item he wants to order, and the VisiCalc formulas will calculate the total price and other associated formulas for that item. Then the estimator uses the **/M** Move command to move each row with the item just ordered to row 250. He does this for each item he orders in the first file.

The **/M** Move command works perfectly for this application. For example, if one whole row has been moved to line 250, moving another row to line 250 causes the first moved row to be relocated to line 249 and the newly moved row to be at 250. Thus you never have to worry about displacing an important row.

Once all the lines from the first file have been entered and moved, the second file is loaded using the **/SL** Load command. The **/SL** command loads a new file over the old in each position of the worksheet. Each position is thus what is in the new file in that position, except that if the new file entry is blank, then the old file entry is kept. In our example, our ordered entries have been moved above line 150, so they are unaffected by the loading of the new file.

Instead, all that really happens is that the 150 original rows are replaced with the 150 rows from the second file. The action is just like turning a page in a catalog. Then the whole process is repeated with the second file, and all other files until all items have been ordered.

Print-Outs

If the estimator is going to order more than 100 items, he must get print-outs with about 50 items on it. This is actually a better procedure since a normal 8 1/2 by 11 page contains only 66 lines. The @SUM command can be entered when the screen is built, summing the line totals from lines 200 to 250. This command is simply:

@SUM(C200 . . . C250)

where **C** is the column the line totals are in. Thus, the job estimator doesn't have to worry about how many items he has entered. If some of the lines from 200 to 250 are blank, then the **@SUM** command ignores them.

The **/P** command can be used to print the rectangle of the worksheet which begins at line 200 and extends to line 252. The first 199 rows won't be printed.

Moving the items to line 250 thus serves two necessary purposes:

1. It gets the ordered items out of the way of the next file load, and
2. It saves a lot of useless printing.

The only way to print out odd numbered rows is to do it one at a time or move them so they are continguous. The 'one at a time' method won't preserve totals, so we move the items. Besides this, the estimator only wants to print out the items he has ordered, not the entire catalog, and the company receiving the order only wants the same.

VisiCore?

(This same trick can be used to copy some 'protected' diskettes. Boot the protected disk, then get into the monitor and move the file into RAM unaffected by a warm boot. Next do a warm boot, and then move the file back and save it onto the nonprotected diskette. This is often quicker than going to cassette. Sorry for the digression, but since this is HARDCORE, I couldn't resist it!!)

This estimation application illustrates the use of two useful commands, the **/M** Move command and the **/SL** Load command.

Problems

Using VisiCalc for this application causes some problems, mostly in the data entry. A good typist has trouble keeping from overrunning the field widths, which the **/E** Edit command helps ease.

The **/R** Replicate command makes the formulas very easy and quick to enter.

And because of the recalculation ability of VisiCalc, changes are easy. This is very important to the estimator, as prices change often. Also, simple changes like a percentage increase in the cost of items, or a larger percent profit, can be easily accomplished with recalculation. All of these are important to the job estimator, since they will all happen in the future.

The real benefit to the estimator is that he doesn't have to worry about the correctness of the arithmetic, and if changes are made (as they usually will be), new updates and print-outs are routine and easy.

Conclusion

This type of VisiCalc application will appeal to a small businessman who handles less than a thousand products. VisiCalc costs about $200 and will do this application and many others like Checkbook balancing, small budgets and other calculations which a calculator would do. Estimation programs can cost several thousand dollars, so this application using VisiCalc is very cost-effective.

In future columns, I will present applications which use other commands. Also, I will review some of the useful Visicalc utilities, like VersaCalc, which makes VisiCalc even easier to use and even more useful. I would like to hear from readers about other uses being made of VisiCalc and will use the best of them in this column.

continued from page 37

When the drive stops, Boot 2 will be in memory at page $Ø3. Move the code at $3ØØ to $93ØØ.

9300<300.3FFM

The exit from Boot 2 is via an indirect JMP at $9343. This JMP normally points into itself. Rather than write any additional code to check when this JMP is changed, allow the code at $3ØØ to be called as a subroutine and change the indirect JMP at $9343 to point to $95Ø1. This works because the JMP at $9343 is not seen until Boot 3 is completely loaded.

Link Boot 1 to Boot 2 and run Boot Ø again.

9343:4C 01 95 N 9843:93 N 9600G

Boot 3 is now in memory. Location $93CC holds the page minus 1 of the start of Boot 3. You should find a $B6 there. That means that Boot 3 starts at page $B7 or $B7ØØ in memory. This is the final Boot. This stage will load the main program.

Boot 3 is large and would be cumbersome to modify if moved. It can be left right where it's at and the Boot 2 code can be changed so that next time it will load in a different location. The page number that Boot 2 uses when it loads Boot 3 from page $B6 to page $D6 will be changed.

9313:A9 D6 EA

This will cause Boot 2 to try to load the Boot 3 code into the space occupied by the monitor. Since the monitor is in ROM (Read Only Memory), nothing will change and the Boot 3 code at $B7ØØ is protected.

At $B749 is a routine that garbages the Boot 3 code after it loads in the program.

At $B759 is a JMP to $6ØØ. This is the actual start of the program. Since $6ØØ is in the text page, the boot with the code can't just be stopped at $95Ø1. If this was done, the code on the text page would scroll or be over-written. The code needs to be moved up to some safe place in memory. Also, The end of the program is at $AØØØ (my guess), which is the same place that DOS resides on a normal disk.

The Galaxian Logo is on HIRES page 1 ($2ØØØ-3FFF). It is a reasonable assumption that this is a safe place where no code exists. The logo will be lost, but the memory is needed.

The following code is a routine that compresses memory by moving page $ØØ thru page $Ø8 up to page $2Ø, and moving page $4Ø thru page $9F down to page $29.

```
B000:A2 00 BD 00 40 9D 00 29
B008:E8 D0 F7 EE 04 B0 EE 07
B010:B0 AD 04 B0 C9 A0 90 EA
B018:A2 00 BD 00 00 9D 00 20
B020:E8 D0 F7 EE 1C B0 EE 1F
B028:B0 AD 1C B0 C9 09 90 EA
B030:4C 59 FF
B000.B032
```

After entering the code, link Boot 3 to it and link Boot 2 to Boot 3.

B749:4C 00 B0 N 9343:4C 00 B7

Check to see that all the changes are correct, then run Boot Ø. When the drive stops you should see a lot of inverse and flashing characters on the screen. This is the portion of the program that is loaded into the text screen area. An image of this code is safely stored at $2ØØØ.

Remove the back-up copy of Apple Galaxian and insert the 48K slave disk in the drive. Boot the disk, then enter the monitor. There are a few more changes to make before the file can be BSAVEd.

The binary program that you are going to save is 132 sectors long. DOS will not normally allow you to save a file this long, so change the range limitation in DOS from 32K to 64K.

A964:FF

When the slave DOS was Booted, it over-wrote the code on page $Ø8. An image of the page $Ø8 code was saved at $28ØØ. Move this code down to page $Ø8.

800<2800.28FFM

Now all that is left is to enter the routine that will move the program back to where it will run. This routine also disconnects DOS and selects the hi-res screen before it JMPs to $6ØØ.

```
2800:A2 00 BD 00 88 9D 00 9F
2808:E8 D0 F7 CE 04 28 CE 07
2810:28 AD 04 28 C9 29 B0 EA
2818:BD 00 20 95 00 BD 00 21
2820:9D 00 01 BD 00 22 9D 00
2828:02 BD 00 23 9D 00 03 BD
2830:00 24 9D 00 04 BD 00 25
2838:9D 00 05 BD 00 26 9D 00
2840:06 BD 00 27 9D 00 07 E8
2848:D0 CE 20 89 FE 20 93 FE
2850:AD 50 C0 AD 57 C0 AD 52
2858:C0 4C 00 06
```

Add a JMP at $7FD to the move routine, and you are ready to BSAVE the file.

7FD:4C 00 28

BSAVE GALAXIAN, A$7FD, L$8103

Now that it is safely stored on the disk, you can run it and discover how much of the memory that was saved is really used. More often than not, all of memory is not used. A large part of what was saved is not required by the program. BLOADing the file and erasing pages before you run the program is the simplest way to determine what sectors are necessary. You could determine if there is sufficient memory that is not used and retrace the Boot code in order to save the HIRES picture. This is a fine point that is not required and is left up to the reader.

I.O.B.

continued from page 37

```
          >> restore normal DOS values <<

90   POKE 47445,213 : POKE 47455,170 :
     POKE 47466,150 : POKE 47335,213 :
     POKE 47345,170 : POKE 47356,173 :
     RETURN

            >> get original disk <<

100  A$ =
     "INSERT ORIGINAL DISK IN DRIVE 1."
     : GOSUB 40

            >> read sector zero <<

110  CD = RD : DV = 1 : GOSUB 50 : CALL
     IO

      >> reset IOB variables for INIT <<

120  VL = PEEK (OVL) : DV = 2 : CD = IN
     : GOSUB 50

      >> INITialize destination disk <<

130  A$ =
     "INSERT BLANK DISK IN DRIVE 2. " :
     GOSUB 40 : CALL IO : VL = 0

     >> CONTROLLER portion of program <<

1000  FOR TK = 3 TO 34
1010  DV = 1 : CD = RD : GOSUB 50 :
      GOSUB 60 : GOSUB 80
1020  DV = 2 : CD = WR : GOSUB 50 :
      GOSUB 90 : GOSUB 80
1030  NEXT

            >> end routine <<

62990  A$ = "COPY COMPLETED" : GOSUB 40
       : END

       >> poke machine subroutine <<

63000  FOR X = 768 TO 796 : READ A :
       POKE X,A : NEXT
63010  DATA 169,3,160,8,32,217,3,96,1,96
       ,1,0,0,0,25,3,0,32,0,0,1,0,0,96,1
       ,0,1,239,216

         >> initialize variables <<

63020  TK = ST = VL = CD = DV
63030  TRK = 780 : SCT = 781 : CMD = 788
       : RD = 1 : WR = 2 : SLT = 777 :
       DRV = 778 : BUF = 785 : ERR = 789
       : VOL = 779 : IO = 768 : INIT =
       4 : OVL = 790
63035  DOS = 15
63040  RETURN

>> address and data mark information <<

63050  DATA 0
```

As copy-protection schemes have changed and improved, so have bit-copiers. Most bit-copiers have several updated versions, and the list of parameters for each copy program is long.

Because protection methods change so rapidly, it would be useless to rate bit-copiers according to which copier is able to duplicate a certain program. By the time this article reached print, it wouldn't be accurate.

Instead, the bit-copiers have been reviewed on the basis of their documentation, ease of use, and versatility.

Documentation is the most important part of a bit-copy program. It must be clear and understandable, yet give complete instruction on how to use the program. It should also go into detail about the use of parameters,showing how to determine which ones are suitable.

Ease of use is a rating based on both how obvious the screen prompts are and how many keystrokes are necessary to achieve a desired result. Screen format is also taken into consideration.

Versatility is reviewed on the basis of what a bit-copier will do aside from copying a disk (it has nothing to do with the number of parameters that a particular program may use). It is an important factor in how useful a bit-copier will actually be, depending on individual purposes, and may contribute to how easy the program is to use.

Review of the "New" Bit Copy Programs

Locksmith 4.1

Locksmith 4.1 comes with an 11-page booklet which contains explanations of its use and features, and a simple description of copy-protection methods. Generally, it is assumed that the reader has a basic knowledge of computer operating procedures and terminology. Each program function is clearly explained, but the text would benefit from additional diagrams and examples.

A separate list of parameter values and their definitions is supplied with the manual, but no attempt is made to show how to determine which parameter should be changed for a particular disk format. Parameters are changed by typing the parameter number from the list, and then entering the desired value.

Locksmith 4.1 has a screen format that is easy to understand. A single keystroke is sufficient to respond to most of the displayed instructions, and it is usually not necessary to press "return."

During a disk copy, a great deal of information is displayed on the screen. Some of it is not explained, however, and is therefore useless. As new information enters the screen, older information scrolls off. A routine is included that will print the status display, allowing a permanent record of the track copy of each disk to be saved.

Locksmith 4.1 comes with a full nibble editor. It is possible to read, edit, and write complete tracks with a process that is simple and straightforward. The internal analyze routines can be used, but the procedure is more involved.

Other program features include a fast disk erase, disk certify, and a disk speed test.

It is explained in the manual that Locksmith 4.1 will work with another program called "The Inspector," which must be purchased separately. This program can save parameter changes, and read and write sectors from a 13 or 16 sector disk. "The Inspector" was not evaluated, since it was not included with the 4.1 disk.

Nibbles Away][

The Nibbles Away II bit-copier is documented with a 40-page booklet which provides detailed information on all of its functions. The use of tables and diagrams contributes to the relatively good quality of the text.

A special beginners appendix gives a brief introduction to copy-protection, and explains what steps to take when the default values are inadequate for copying a particular disk. Another appendix for advanced users describes protected disk formats and discusses parameters. An appendix on nibble counting is also included.

Nibbles Away II screen prompts are easy to understand. They are displayed in a single menu along with their corresponding default values, which enables viewing of all the information at once. If the default values are used for a copy, it takes only a single keystroke to start the program.

The system modifier is responsible for parameter changes, and may be entered into by typing "M" from the master menu. Another menu of parameters appears, from which the set to be changed is selected. This set is then displayed along with the parameter names and their current values in a menu that is both convenient and easy to use.

The Track/Bit Editor (TBE) is a normal DOS editor. A sector from a 13 or 16 sector disk can be read. Changes can be made to the sector data in memory, and the sector can be written back to the disk. This is required to copy some disks.

The exec files included with Nibbles Away II are a unique feature. These are files that make automatic changes to the parameters in memory so that a particular disk can be copied more easily. Not all of the exec files proved accurate. Some files loaded properly, but the disk did not copy.

A special filing system allows parameter changes to be saved on a data disk created with the program, so that the next time a particular disk is backed-up it is only necessary to load the appropriate parameter file.

A disk diagnostic utility is also included on Nibbles Away II that will erase a disk, check a blank disk for flaws, and test drive speeds.

Parameters for Locksmith and Nibbles Away have been included in this issue along with their explanations and uses. Parameters for Back It Up II will probably be in the Update or following issue (as soon as we convert the files and get more detailed explanations on their uses).—EDITOR

Back It Up][+

Back It Up II+ is accompanied by a 14-page manual that is detailed enough to be a decent guide for someone who is new to the bit-copying process. It gives a good description of the functions of a nibble copier, and the step-by-step directions are aided by examples. A section on copy-protection is also included.

As far as parameters are concerned, the text gives little in-depth information. An attempt is made to explain how to use selected individual parameters, but it falls short from lack of hard examples. The manual lists parameter values with brief definitions in an appendix, and a section on "Advanced Techniques" contains instructions on changing parameters.

The process of changing parameters is fairly simple. A "/" is entered in response to any question from the screen display. Then the program will ask which "parm#" should be changed and what "value:" is desired.

The Back It Up II+ screen prompts are clear and understandable. The extensive use of default answers make hitting the "return" key the only response that is usually needed.

While the copy is being made, single letters are used to follow the track copy status. At this time the screen display is rather lifeless, as there is just one point changing at a given moment, but setting parm $0C to $F0 helps by causing the program to display more information.

A nibbler is included with Back It Up II+ as an aid in finding the proper parameter values. By pressing the "R" key in response to any screen prompt, the program goes into the "Track Examination Mode." The number of the track to be studied can then be entered

This is a "read only nibbler," and does not allow editing and writing to the disk.

Closing Remarks:

Out of the three bit-copiers reviewed, Nibbles Away II has the most complete documentation. Although more step-by-step directions could have been used in describing the process of simple copying, the material given is the most detailed and informative.

As for screen prompts, Nibbles Away II has an advantage over the other two programs because all the information is displayed at once. Back It Up II+ and Locksmith 4.1 allow most prompts to scroll off the screen after a response is given. This method is simpler in appearance than the Nibbles Away II menu of prompts, but the latter takes little getting used to and has far more benefits.

During the copying process, Locksmith 4.1 has an active screen display that provides the most information. The Back It Up II display is easier to understand, but more static. Nibbles Away II has a screen display that is equally easy to understand and very lively. It will print "READING" or "ANALYZING," rather than just the letter "R" or "A." The program will beep when the copy is finished.

Nibbles Away II has many more features than either of the other bit-copiers. The auto-load or exec files make copying some disks very easy. This copier may also be booted from any slot, while Locksmith 4.1 and Back It Up II+ must be booted from slot 6. The system filer is a real advantage to someone who doesn't use bit-copiers often and may forget how to use the parameters (they can be stored on the special data disk for future use).

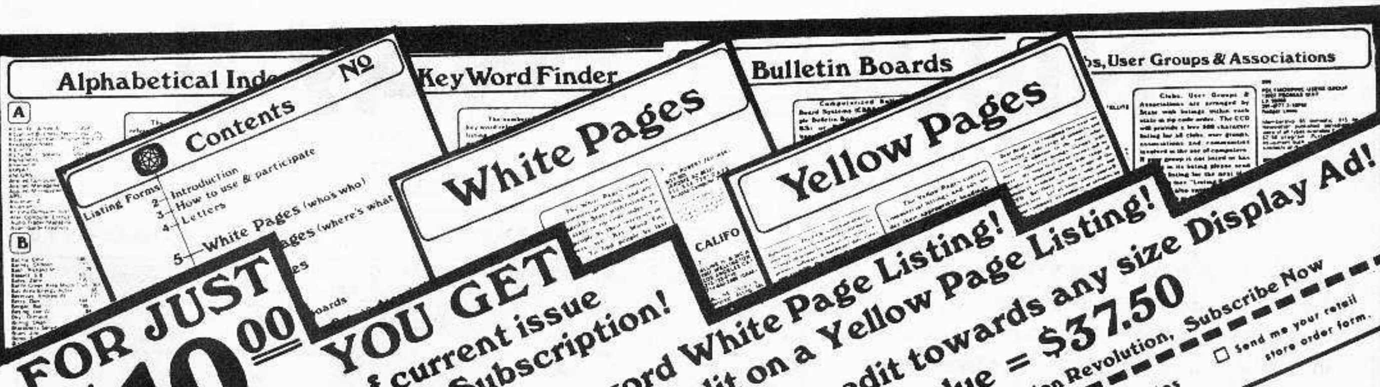
As a final note on parameters, detailed instructions on their use can be found on "The Source," a telecommunications network accessible by MODEM. However, only paid subscribers can use the network. (Reprints of these articles can be found elsewhere in this issue.) It would be more convenient if complete parameter instructions were supplied with the bit-copy programs.

The national phone book of computing

the Computerist's Directory

$7.50

"Let your fingers do the talking"



FOR JUST >>> $10.00 YOU GET

1) a Free copy of current issue
2) a One Year Subscription!
3) a Free 25 word White Page Listing!
4) $5.00 Credit on a Yellow Page Listing!
5) $10.00 Credit towards any size Display Ad!

Total Value = $37.50

Save $18.50 and Join the Information Revolution, Subscribe Now

Send me ☐ Display ad rates ☐ Send me your retail store order form.

ENCLOSED $

☐ VISA ☐ MASTER CHARGE ☐ AMERICAN EXP

EXPIRES

ACCOUNT NO.

SIGNATURE

NAME

ADDRESS

CITY

STATE

ZIP

the Computerist's Directory
PO BOX 405
FORESTVILLE, CA 95436
(707) 887-1857

The Computerist's Directory is becoming the one reference to your products.

The Computerist's Directory is published twice annually in January and July with quarterly up dates planned for April '82.

WHY YOU NEED LOCKSMITH.

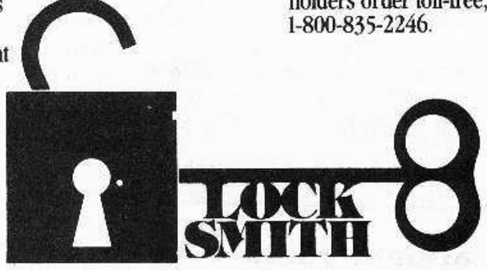
You've invested some money and a lot of time in a commercial software program for your Apple. It works well, to the point that you are *dependent* on its day-to-day functioning. But the disks are copy-protected. So you are also dependent on the vendor's back-up (if furnished), on his living up to vague promises of support, even on his ability to stay in business.

No computer user can live with that. So until the situation changes (and it will), you need Locksmith.

Locksmith (new 4.0 version) will copy almost all "protected" diskettes for the Apple. It is the most reliable nibble-copy program you can buy. *Locksmith is suitable only for backups,* because the copies include all serial numbers, codes and protection features of the original (under the new copyright law, you'd have to be pretty foolish to try bootlegging software that is traceable back to the purchaser).

Locksmith includes nine other utilities, of which these five are vital to the integrity of your system: 1. Media surface check — Never commit data to a flawed diskette again. 2. Disk-drive speed calibration — the most frequent cause of communication bugs between Apples. 3. Degauss and Erase — Make sure no stray data is left over. 4. Nibble-Editor — sophisticated read/write tool for repairing blown disks. 5. Quickscan — Check for unreliable data, find used and unused tracks.

All for just $99.95 at your local dealer or direct. You don't just need Locksmith. You can't afford to be without it. Mastercard and Visa holders order toll-free, 1-800-835-2246.



OMEGA MICROWARE, INC.
222 SO. RIVERSIDE PLAZA
CHICAGO, IL 60606
312-648-1944

Apple is a registered trademark of Apple Computer, Inc.

★ ★ ★ INTRODUCING ★ ★ ★

DEALER INQUIRIES INVITED

NIBBLES AWAY II

SIMPLY THE FINEST BIT COPIER EVER CREATED!!

Nibbles Away II provides a means to protect your software investment by allowing you to make additional copies (back-ups) of almost all Apple "protected" diskettes.

Nibbles Away II is user orientated; included are tutorials for all levels of expertise, beginners flowchart for "Where Do I Begin" to "advanced disk analysis." Nibbles Away II is the ultimate bit copier for the Apple.

ONLY $64.95

COMPARE FOR YOURSELF	NIBBLES AWAY II	LOCKSMITH 4.0
AUTO-LOAD PARAMETER FILES - PROVIDES EASIER ACCESS TO PARAMETER CHANGES.	YES	NO
DISASSEMBLE FUNCTION - ALLOWS ANY SECTOR TO BE LISTED IN ASSEMBLY LANGUAGE FORM.	YES	NO
TRACK/SECTOR EDITOR - ALLOWS USER TO READ/WRITE/EDIT ANY SECTOR ON DOS 3.2 OR 3.3 DISKETTES.	YES	NO
READ/WRITE/EDIT NIBBLES - ENABLES USER TO READ NIBBLES AND DUMP THE SCREEN TO THE PRINTER.	YES	NO
PARAMETER CONVERSION - PARAMETERS PUBLISHED FOR SIMILAR BACK UP SYSTEMS MAY BE USED WITH NIBBLES AWAY II.	YES	NO
DISK DRIVE SPEED CALIBRATION	YES	YES
TEST DISKETTE MEDIA RELIABILITY	YES	YES
DISKETTE DEGAUSSING OPTION	YES	YES
FREE BACK UP DISK	YES	NO

TERMS: WE ACCEPT M.C. OR VISA (INDICATE CARD # AND EXPIRATION DATE) PERSONAL CHECKS ALLOW 3 WEEKS TO CLEAR. PLEASE ENCLOSE $2.00 FOR POSTAGE AND HANDLING. LOUISIANA RESIDENTS - PLEASE ADD 6% SALES TAX. NO C.O.D.'S.

CALL (318) 942-9446
(318) 948-6305

COMPUTER HIDEOUT P. O. BOX 264
OPELOUSAS, LA 70570

LOCKSMITH IS COPYRIGHTED BY OMEGA MICROWARE.
APPLE IS A REGISTERED TM OF APPLE COMPUTER, INC.

the compleat guide to LOCKSMITH PARAMETERS

This document describes all user-changeable Locksmith parameters in depth. A partial list of these parameters was previously distributed to all version 4.0/4.1 owners. Also provided here is detailed Locksmith program logic information.

Note---this document is of a highly technical nature, and is intended primarily for the advanced user of Locksmith.

BACKGROUND

When Locksmith was first introduced in January 1981, it would copy almost all disks with no special instructions from the user. Only a few disks required parameter changes. Alas, those good old days are gone forever. Instead of providing the user with better backup policy, software vendors decided to escalate the battle by developing more complicated (and in some cases, bizarre) protection techniques. Because of the many different techniques now in use, it is likely that many disks will require some input from the user in the form of parameter changes.

Omega Microware currently maintains an extensive list of software, along with the Locksmith parameters used to copy each. Some of the entries on this list are user-supplied. Omega Microware welcomes information from users regarding how to back-up software not already on this list.

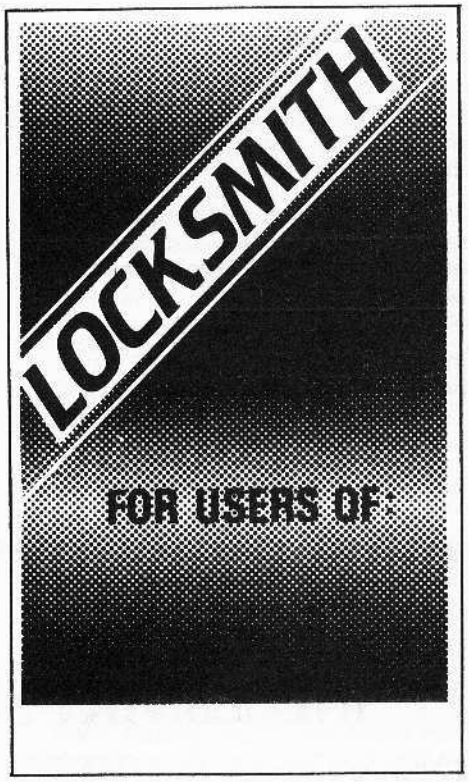
OVERVIEW

Locksmith copies disks by reading a track, performing analysis on the data, and writing the track to the copy disk. Reading and writing are fairly straightforward functions.

The analysis of the track data is by far the most difficult task, and must provide for flexibility. Many analysis routines (algorithms) are provided within Locksmith. Each algorithm performs a specific function relating to the analysis of track data. By changing parameters, the user may select, disable, or change the execution order of algorithms. Parameters may also be used to define values to be used by individual algorithms.

ALGORITHMS

The algorithms are numbered from 0 to $23 (all values are in hex). New algorithms may be added in future versions of Locksmith. During track analysis, algorithms are selected sequentially



from a table of algorithm numbers, located from PARM 4C-80. As algorithms are selected from this table during analysis, they are displayed on the screen as 2-digit hex numbers in inverse video.

Algorithm 00 indicates a null algorithm, which can be used to replace algorithm numbers in the table which the user wants to disable. An FF entry in this table indicates the end of the algorithms to perform.

Currently, the algorithm table contains four separate algorithm sequences, each one terminated by an FF entry. The starting point of the algorithm sequence to be used is defined by PARM 25. This parameter contains the index into the algorithm table to be used as the first algorithm of a sequence. For example, if PARM 25 = 00, the algorithm sequence would start at PARM 4C. If PARM 25 = 10, the algorithm sequence would start at PARM 5C. The section of algorithm table starting at PARM 71 is selected as an algorithm sequence start (instead of PARM 4C) when synchronized tracks are chosen.

Algorithms, in addition to performing their specialized function, can return a flag to indicate success or failure. It is possible to indicate that an algorithm is to be performed only if the previous algorithm failed. This may be done by setting the high-order bit of the algorithm number within the algorithm table. For example, an entry of A1 indicates that algorithm 21 is to be performed only if the previous algorithm failed.

Description of Algorithms

The following is a list of algorithm numbers and the parameters which affect them.

ALG 00 (This algorithm doesn't do much of anything)

ALG 01 (Consecutive nibbles to self-sync) changes normal nibbles to self-sync nibbles based on: finding (PARM 10) consecutive nibbles in the range (PARM 34) to (PARM 35), inclusive. For example, if PARM 10 = 0C, PARM 34 = FE, and PARM 35 = FF, then algorithm 01 would search for sequences of length 0C nibbles with values FE through FF, and set them to self-sync.

ALG 02 (Invalids to self-sync) sets invalid nibbles (those with 3 or more consecutive zero bits) to self-sync.

ALG 03 (Standardize self-sync) sets all self-sync to (PARM 33), which must have high-order bit clear.

ALG 04 (Loner self-sync to normal) set consecutive self-sync strings of less than or equal to (PARM 3C) to normal.

ALG 05 (Glitch remover) set consecutive normal nibbles of length less than or equal to (PARM 12) to self-sync.

ALG 06 (Set self-sync by marker pattern match) searches for pattern specified by (PARM 44-4B), and sets the previous (PARM 40) nibbles to self-sync. Values of 00 within the pattern are "don't care" and always match.

ALG 07 (Extend bit shifted self-sync) extends self-sync strings backwards, using the table at (PARM 86-A5). This table contains nibble value sequences frequently found to be self-sync.

ALG 08 (Reserved for future use)

ALG 09 (Trackstart after longest gap)

set trackstart to first normal after longest string of self-sync (gap).

ALG ØA (Minimum length self-sync) extend self-sync strings backwards to minimum length of (PARM 2C).

ALG ØB (Set self-sync by self-sync pattern match) set self-sync based on multiple-byte pattern match. Pattern is defined at (PARM 81-85) and is terminated with a ØØ value.

ALG ØC (Shorten all gaps) shorten all gaps (consecutive strings of self-sync) by (PARM 41) nibbles if the string length was greater than or equal to (PARM 16).

ALG ØD (2 of 3 gap merge) merges first and second gaps (by setting to self-sync, nibbles between them) if 3 gaps are found within (PARM 26) nibbles. (The gaps merged are usually the gap after a data field.)

ALG ØE (Trackstart after first self-sync) sets trackstart to first normal after the first string of self-sync.

ALG ØF (Shorten longest gaps) shorten the longest gap if longer than (PARM 2C) by (PARM XX) nibbles. Repeat this procedure (PARM YY) times. XX=27 (or 29 if synchronized) YY=28 (or 2A if synchronized)

ALG 1Ø (Reserved for future use)

ALG 11 (Set failure flag) same as algorithm ØØ, but sets the failure flag.

ALG 12 (Trackstart by marker pattern match) set trackstart to the first sequence to match pattern at (PARM 44-4B). (see ALG Ø6)

ALG 13 (Center of gaps to normal) leaving 8 self-sync at the start and at the end of a gap, set self-sync in the center of the gap to normal.

ALG 14 (Bit-translate to self-sync) using the bit table at (PARM D9-E8), translate nibbles corresponding to a one-bit to self-sync. Bits in the table represent values for nibbles in the following order: 8Ø,81,82, . . . FC,FD,FE,FF

ALG 15 (Reserved for future use)

ALG 16 (Reserved for future use)

ALG 17 (Track-end and compare) this algorithm searches for a repeat of the track-start beginning at (PARM 1D) pages beyond the current track-start. A repeat of the track-start is determined by matching (PARM 1E) number of nibbles. If the track size is greater than (PARM 1B) pages, an error 2 status code will be issued.

Once a track-end is chosen, the first two track images are compared, nibble for nibble. If an unequal nibble compare occurs, a look-ahead of up to (PARM 13) nibbles is performed, looking for self-sync.

If self-sync is found, the compare failure is ignored. If no self-sync is found during this look-ahead, a counter is incremented for the compare-failure, and this count is checked against (PARM 14), which must not be exceeded, or an error 4 status code is issued immediately.

WHY DO UNNECESSARY SURGERY ON YOUR APPLE?

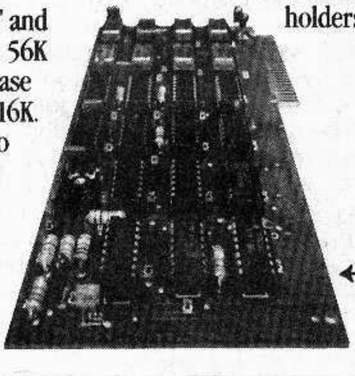
Sooner or later, you're going to need a 16K memory-expansion for your Apple. When you do, we suggest you buy it on the card that doesn't require poking about on the motherboard — nor removing a RAM chip, installing a strap, etc.

The Ramex 16 RAM Board just plugs in. It's simple, reliable, and does its own memory refresh, with no additional connections.

Run Pascal, Fortran, FP, INT and other alternate languages, 56K CPM with a Z80 Softcard, increase usable memory for Visicalc by 16K. The possibilities are endless. Do it with the finest, closed-track engraved, epoxy sealed, 16K board available — the Ramex 16. And do it without unnecessary surgery on your Apple.

In spite of its quality, the Ramex 16 costs less than most other expansion boards — just $139.95. And it comes with a *one year* limited warranty, instead of the usual 90 days.

Get the Ramex 16 from your local dealer, or order direct. Visa and Mastercard holders call toll-free, 1-800-835-2246.



OMEGA MICROWARE, INC.
222 SO. RIVERSIDE PLAZA
CHICAGO, IL 60606
312-648-1944

Apple and Applesoft are registered trademarks of Apple Computer, Inc. Pascal is a registered trademark of the Regents of the U. of C., San Diego. Visicalc is a registered trademark of Personal Software. CPM is a registered trademark of Digital Research, Inc. Z80 is a registered trademark of Zilos, Inc. Softcard is a registered trademark of Microsoft.

The 3rd track image is then used as a tie-breaker to determine which of the 1st or 2nd track images is correct. The exact position in the 3rd track image is found by first finding the approximate location in the 3rd image (by using track length), backing up (PARM 11) nibbles, and pattern-matching (PARM 32) number of nibbles, while searching through the next (PARM 31) number of nibbles. The 1st image is corrected by the tie-breaker nibble. This algorithm returns a success/fail flag.

ALG 18-1F (PARM modifier) these algorithms are used to dynamically modify PARMs. The table at (PARM B6-D8) consists of several sequences of PARM modifier entries. Each PARM modifier entry consists of a pair of bytes. The 1st byte defines the PARM number, and the 2nd byte defines the new PARM value. The end of a sequence is indicated by a 00 entry for PARM number, and a new sequence begins with the next byte. Algorithm 18 invokes the 1st sequence of parameter modifier entries, algorithm 19 invokes the 2nd sequence, etc. Using these algorithms, parameters may be automatically changed and restored during analysis. The defaults for these algorithms are currently set as follows:

ALG 18 Sets 13-sector PARMs.

ALG 19 Sets 16-sector PARMs.

ALG 1A Sets misc. PARMs.

ALG 1B Sets nibble-counting PARMs.

ALG 20 (Go to Nibble Buffer address) This algorithm is used in conjunction with the Nibble Editor. It prompts the user for an address to go to, and the Nibble Editor cursor is immediately placed at that location. (See: Invoking Algorithms from the Nibble Editor)

ALG 21 (Set error code 1) issues an error 1 status code. It is usually placed in the algorithm table with the high-order bit set, to cause it to execute only when the previous algorithm fails.

ALG 22 (Backup trackstart to front of gap) moves the trackstart pointer backwards to the beginning of the preceding gap.

ALG 23 (Set trackstart to longest normal) sets trackstart to the 1st nibble of the longest sequence of normal nibbles.

Printer Control PARMs

(Parm 2D) specifies the printer slot, and (PARM 2E) is set to 00 if Locksmith is not to generate >CR< at the end of a line, or left at 01 if >CR<'s are to be generated.

Maximum Error Count PARMs

(Parm 01),(PARM 02), and (PARM 04) are used to specify the number of errors allowed for error codes 1,2, and 4 in automatic error retry mode. If increments of 1/2 tracks are used, (PARM 09),(PARM 0A), and (PARM 0C) are used instead.

Nibble-Counting PARMs

There are 3 parameters which are used when nibble-count preservation is desired. Setting (PARM 36) to 01 turns on nibble-counting. The nibble-count tolerance value, (PARM 37), specifies how close to the original disk the copy must be. When nibble-counting, the track-end pointer is moved up by (PARM E9) pages before writing.

Parms Used for Synchronizing

(Parm 22) specifies the track*2 to sync to. This is normally 00, but may be set to any track. (Parm 1F) is the length of the nibble sequence to sync with, and (PARM A6-B5) contain the pattern to match when attempting to sync on the sync-track. Values of 00 within the pattern are "don't care" and always match. (PARM 23) and (PARM 24) are values which can be used to adjust the accuracy of the sync-track routine. They are normally equal, and can be adjusted by increasing the value of one with respect to the other.

Note: Nibble count adjustment from software (using '>' and '<') will not work if a type-ahead buffer is installed on the Apple. To perform nibble count preservation, either remove the type-ahead buffer, or adjust the disk speed pot directly.

When the Nibble-Editor is entered from the main menu, the track is read into the buffer with no Locksmith analysis whatsoever. No nibbles will be indicated as self-sync (inverse), as this is determined when analysis occurs during a disk copy operation. Also, CTRL-B and CTRL-E will only place the cursor at the beginning and end of the buffer, because track-start and track-end are also determined by analysis during disk copy.

To enter the Nibble-Editor after analysis by Locksmith, perform a copy operation, specifying manual error retry. Open the door of the copy disk drive to cause a verify error (error code 8) and enter the nibble editor by selecting:

5. NIBBLE-EDIT CURRENT TRACK DATA.

In this way a track may be nibble-edited after Locksmith has performed analysis on it.

Parms Used to Control Writing

(Parm 20) contains the lead-in self-sync nibble value. (Parm 2F-30) (default is $1A00) number of these lead-in self sync nibbles are written before track data is written, with the exception of synchronized track writing, which is preceded by (PARM 23) lead-in self-sync nibbles. The number of framing bits (1 or 2) is contained in (PARM 21). This places the proper number of trailing zero-bits after self sync. (Parm 2B) contains the number of the algorithm to be used to shorten the track after an over-write is detected by verify readback failure.

Other PARMs

(Parm 38) is the number of nibbles to test during verify readback. (Parm 39), if set non-zero, shows the hi-res screen during analysis, to provide a graphical representation of analysis. (Parm 3A) is used during disk certify. It specifies the maximum size of the track-end glitch. (Parm 3B), when set to 01, causes the Nibble-Editor to be entered for every track, before analysis.

Debug Parameter

(PARM 00) is a special parameter intended for use during Locksmith debugging. When this PARM is set to 11, certain debugging options are enabled. They are:

1. Inspector entry is allowed even with no resident RWTS.
2. Nibble-Editor is entered without prompting the user for track to read. This allows the previous track to be examined.
3. Invoking algorithms from the Nibble-Editor.

Invoking Algorithms from Nibble-Editor

With debug PARM set (PARM 00=11), the Nibble-Editor is sensitive to two additional commands. These are control-S and control-A. Control-S invokes Locksmith track-analysis for the track currently in the Nibble Buffer. Control-A first allows the user to change parameters by entering the parameter modifier, and after the user has indicated the end of parameter changes with a >CR<, it prompts the user for algorithm number.

The user-entered algorithm number is executed immediately, and control is returned to the Nibble-Editor. In this way, the user can dynamically test the effects of specific Locksmith algorithm sequences when attempting to copy unknown disks. Algorithm 00 can be specified if no processing is to be done. Algorithm 20 is very useful within the Nibble Editor to rapidly go to a specific address within the Nibble Buffer.

List of LS PARAMETERS

Maintaining an extensive and detailed list of disks which Locksmith will copy, as well as any parameter changes or patches which the user must make to copy certain disks is a gargantuan task.

Omega Microware is working hard to provide users with necessary PARM changes as they determine them. They openly solicit user input regarding Locksmith parameters which work for certain disks.

Uncopyable?

If the recommended parameter changes are made and the disk still won't copy, follow these instructions:

1. Make sure the disk drive speed is correct.

2. Retry the copy, reversing the order of the disk drives used.

3. If possible, attempt the copy on a different Apple or set of drives to make sure that the problem is not that particular set of disk drives.

4. If all fails, write Omega Microware about the problem including all pertinent information about what attempts were made to copy the disk.

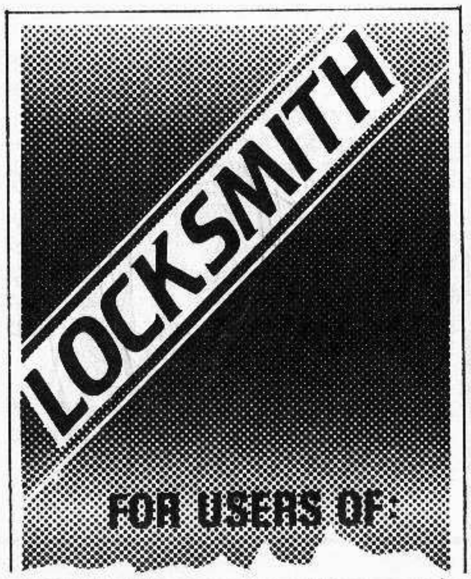
Counting Every Nibble...

Some protected disks use a technique known as nibble counting. This technique is based on the fact that all Apple disk drives run at slightly different speeds, and even the speed of a specific disk drive varies slightly over time. Disks which are protected by this method count the nibbles on a given track and record this unique number somewhere else on the disk. When the disk is booted by the user, the nibble count on the track in question is checked against the correct value. Simply copying the track will almost always write a different number of nibbles due to disk drive speed variation.

Locksmith 4.0 has a provision for nibble counting, and will preserve nibble counts on any track requested. To request nibble count preservation, set PARM 36 to 01. (This is automatically done for the user when PARM 4C is set to 1B.) After the track is written to the copy disk, the nibbles are counted and compared to the original count to be preserved. The difference is shown as a four-digit hex number preceded by '>' or '<' to indicate to the user which way to manually adjust the count. The count may be adjusted in one of two ways. Either the disk speed adjustment pot can be turned in the direction indicated by the '>' or '<' arrows (see user manual regarding disk speed adjustment), or a limited amount of adjustment can be done from software without actually adjusting the disk speed.

To adjust the nibble count from software, press either '>' or '<' as indicated, and wait until the speaker begins beeping. The speaker will beep rapidly once for each nibble that the track is being shortened or lengthened. Then press the return key (or any key other than '>' and '<') and allow the nibble count routine to test the track again. When the nibble count is within the tolerance value specified by PARM 37 (normally 00), the track will be considered copied correctly. This technique may seem cumbersome, but it is the only way in which a track may be copied while preserving the nibble count.

```
SOURCE FILE TY (10)TCA640>LS.PARAMETERS)
SYMBOL   MEANING
------   -------
T      = Tracks to copy.
S      = Use syncronized copy.
...    = PARMs to be changed before
         copying tracks indicated.
BY     = Shows track increment (By 1 if
         not otherwise indicated).
**     = User supplied. Not verified as to
         accuracy.
( )    = Name of patch to apply before
         copying. Patches are for version
         4.0 only.
(V4.1) = PARMs for version 4.1 only.
```



```
  NOTE: When following instructions
below, do each step in the order listed.
Change parms before copying tracks
indicated to the left of the periods.
Previously entered PARMs remain as
changed unless changes are indicated in
subsequent lines.
  If a particular program is not listed,
try tracks 0-22 normal. Programs that
only require T0-T22 normal are not
listed due to space constraints.


A2-FS1 (New as of 11-1-81)
   0
   1.5-21 BY 1.5......44=DB 45=AB 46=BF
                      40=20 4E=00 54=12
   7-8
   9.5

AIRSIM - 1                      (NOTE 2)
S  0-2..........A8=B5 78=00 79=12 47=FF
S  9-22
S  3-8..........79=0E (ERR ON TRK 8 OK)

AKALABETH ✓
S  0-18.........44=DD

ALIEN TYPHOON ** ✓
S  0-E

ALIEN TYPHOON ** ✓
   0-0
   1-5..........46=AD
   6-E..........44=DE

APPLE-OIDS **
   0-22
   3.5-3.5
   21.5-21.5

APPLE PANIC ✓
   0-D

APPLE PANIC ** ✓
   0-0
   1-C..........44=DD

APPLE PRINT USING **
S  0-23

APPLE WORLD
   0-23

APPLEWRITER (APPLE ///)
S  0-22

AUTOBAHN ✓
   0-0
   4-6..........74=00
   9.5-C.5

BAG OF TRICKS ** ✓
   0
   1-14.........40=10 44=D6 53=00

BATTLE OF SHILOH **
      Same as TORPEDO FIRE
```

```
BEER RUN  ** ✓
   Ø-Ø..........18=2Ø 19=ØØ 46=96 4D=ØØ
                4E=ØØ 52=ØØ 53=ØØ 54=12
                57=ØØ 4Ø=2Ø
S  1.5-D.5

BEER RUN  ** ✓
   Ø-Ø
S  1.5-D.5......72=ØØ 73=ØØ 77=ØØ 78=ØØ
                79=12 7C=ØØ 4Ø=2Ø 19=ØØ
                44=DD 45=AD 46=DA

BPI BUSINESS ACCOUNTING SYSTEM ✓
(All 4 disks - Revised 1Ø/26/81)
   Ø-22.........19=ØØ 21=Ø2 58=19 59=Ø6
                5A=1A 5B=FF BD=44 BE=E6
                BF=45 CØ=FF C1=4Ø C2=Ø1
                C4=44 C5=D5 C6=45 C7=AA
                C8=4Ø C9=Ø4 CA=ØØ

CARTELS AND CUTTHROATS **
   Same as TORPEDO FIRE

CASTLE OF DARKNESS **
S  Ø-Ø..........46=96
S  1-23.........44=AB 45=AB 79=12

CASTLE WOLFENSTEIN ✓
S  Ø-22

COMPUTER AIR COMBAT
   Ø-22.........25=19 65=ØØ 6B=ØØ

CONTEXT CONNECTION
   Same as DESK TOP PLAN II ✓

COPTS & ROBBERS ✓
   Ø-Ø..........(PARMS same as GORGON)
S  1.5-F.5......(PARMS same as GORGON)

CRANSTON MANOR  ** ✓
   Ø-22
   18.......... 53=ØØ 44=D5 45=FE 4C=1B
                57=ØØ

CROSSFIRE ** ✓                 (NOTE 3)
   Ø-22
   1............4C=1B              <NC3Ø>
                4C=1B 57=ØØ E9=Ø2 (V4.1)

CROSSWORD MAGIC ** ✓
S  Ø-22.........46=96 75=ØØ 76=ØØ 77=ØØ
                78=ØØ 79=12 4B=AA

CRUNCH, CRUMBLE & CHOMP  ** ✓
S  Ø-22

CYBERSTRIKE ✓
   Ø
S  4-B..........46=F5 79=12
   11-1C........46=B5

DARK FOREST  ** ✓
S  Ø-22

DARK FOREST  ** ✓
   Ø............18=2Ø 19=ØØ 46=96 4D=ØØ
                4E=ØØ 52=ØØ 53=ØØ 54=12
                57=ØØ 4Ø=2Ø
S  2-22.........72=ØØ 73=ØØ 77=ØØ 78=ØØ
                79=12 7C=ØØ 4Ø=2Ø 19=ØØ
                44=D5 45=AA 46=AF
```

```
DATADEX
S  Ø-2..........79=12 46=96
S  3.5
S  5-22

DATA FACTORY 5.Ø             (NOTE 2,4)
S  Ø-23.........46=96 79=12 71=19 75=ØØ
                76=ØØ 77=ØØ 78=ØØ
DATA PLAN **
S  Ø-22

DATA REPORTER
   Ø-22         4D=ØØ 46=96 54=12

DB MASTER AND UTILITIES DISK   (NOTE 2)
S  Ø-5
   6.5-22.5

DESK TOP PLAN II
   Ø-22.........19=ØØ 21=Ø2 58=19 59=Ø6
                5A=1A 5B=FF BD=44 BE=EB
                BF=45 CØ=FD C1=4Ø C2=Ø1
                C4=44 C5=D5 C6=45 C7=AA
                C8=4Ø C9=Ø6 CA=ØØ

DISK ORGANIZER II  ** ✓        (NOTE 3)
   Ø
S  2-4
S  A-B
   1............4C=1B              <NC3Ø>
                4C=1B 57=ØØ E9=Ø4 (V4.1)

DISK RECOVERY ** ✓
   Ø
S  2-4
S  A-B

EPOCH (REVISED AS OF 1Ø-26) ✓
   Ø............18=2Ø 19=ØØ 46=96 4D=ØØ
                4E=ØØ 52=ØØ 53=ØØ 54=12
                57=ØØ 4Ø=2Ø
S  1.5-F.5......72=ØØ 73=ØØ 77=ØØ 78=ØØ
                79=12 7C=ØØ 4Ø=2Ø 19=ØØ
                44=D5 45=AA 46=DA

ESCAPE FROM ARCTURUS ✓
   Ø-22.........4D=ØØ

EXECUTIVE SECRETARY
   Ø-22.........46=96 54=12

EXPEDITER                      (NOTE 3)
   Ø-22
   3 & 1F.......4C=1B              <NC3Ø>
                4C=1B 57=ØØ E9=Ø2 (V4.1)

FALCONS **  (New as of 12/21/81) ✓
   Ø
   1.5-4.5 BY 1.5...18=2Ø 34=AA 44=DF
                    45=AD 46=FE
     (IF THE ABOVE PARMS DON'T WORK TRY:
     18=4Ø 19=ØØ 34=AA 4Ø=4Ø 44=DF 45=AD
     46=ØØ 4E=ØØ 4D=ØØ 52=ØØ 53=ØØ)
   5.5
   7-A BY 1
   8.5-E.5 BY 1.5
   1Ø-12 BY 1
   13.5-14.5 BY 1
   16-19 BY 1.5
   1A-1B.5 BY 1.5

FALCONS ** ✓
S  Ø
S  1.5-19.5
```

```
FINANCIAL CONTROLLER (ALL 5 DISKS) **
S  Ø-22

FIREBIRD ✓
   Ø............18=2Ø 19=ØØ 46=96 4D=ØØ
                4E=ØØ 52=ØØ 53=ØØ 54=12
                57=ØØ 4Ø=2Ø
S  1.5-B.5 BY 1...... 72=ØØ 73=ØØ 77=ØØ
                78=ØØ 79=12 7C=ØØ 4Ø=2Ø
                19=ØØ 44=DD 45=AD 46=DA

GAMMA GOBLINS **
     Same as SNEAKERS

GENETIC DRIFT ✓
   Ø............18=5Ø 19=ØØ 4Ø=2Ø 46=96
                4D=ØØ 4E=ØØ 52=ØØ 53=ØØ
                54=12 57=ØØ
   1-3..........44=BB 45=D5 46=BB
   4.5-6 BY 1.5
   7.5-B.5 BY 1
   D............44=D4 45=D5 46=BB
   E.5-12.5.....44=AD 45=B5 46=DE

GOBLER                         (NOTE 5)
   Ø-22.........4E=ØØ ✓
   3............4C=1B D2=ØØ 45=BD 4E=Ø1
                34=FF 54=12 52=ØØ

GORGON (NEW AS OF 11-1-81) ✓
   Ø............18=2Ø 19=ØØ 46=96 4D=ØØ
                4E=ØØ 52=ØØ 53=ØØ 54=12
                57=ØØ 4Ø=2Ø
S  1.5-E.5......72=ØØ 73=ØØ 77=ØØ 78=ØØ
                79=12 7C=ØØ 4Ø=2Ø 19=ØØ
                44=DD 45=AD 46=DA

HAYDEN APPLESOFT COMPILER
S  Ø-22.........46=996 71=19 79=12
                (Errors on 1Ø-1E OK)

HAYDRON **
     Same as GORGON

HIRES CRIBBAGE AND HIRES FOOTBALL ** ✓
S  Ø-5
   6-22

HIRES GOLF **
   Ø-22.........4E=ØØ 46=B5 54=12

INVOICE FACTORY (2 DISKS)
   Ø-22.........46=96 54=12

JAWBREAKER ** ✓
   Ø-22
   3............34=FF 44=DF 45=EF 46=F7
                5Ø=ØØ 51=ØØ 52=ØØ 53=ØØ
                54=12

LETTER PERFECT
   Ø-22.........44=ØØ 45=D5 46=AA

MAGIC WINDOW
   Ø-22.........4F=ØB

MASTER TYPE ✓
   Ø-2
   3-1A.........44=D4
   1C-22

MICRO BASEBALL **
   Ø-4
S  5-22
```

```
MICRO COURIER
   Ø-22
   1F...........81=97 82=EB 4Ø=Ø8 16=Ø8
                41=FF 19=ØØ 58=ØB 59=FF

MICRO TELEGRAM
     Same as MICRO COURIER
MICROWAVE ✓
   Ø-22
   11...........4C=1B 57=ØØ E9=Ø2

MISSLE DEFENSE ✓
S  Ø-22

MULTI DISK CATALOG III ** ✓
S  Ø-22

NIGHT MISSION **                    (NOTE 2)
   Ø
   1-15.........44=BB 45=AB 46=DF 4Ø=2Ø
                4E=ØØ 54=12

OLYMPIC DECATHALON ✓
S  Ø-22.........46=B5 A8=ØØ 71=18 79=12

OO-TOPOS ✓
   Ø-22.........21=Ø2

ORBITRON (Same as GORGON) ✓

OPERATION APOCALYPSE
   Ø-22.........25=19 65=ØØ 6B=ØØ

OUTPOST (Same as SNEAKERS) ✓

PALACE IN THUNDERLAND
   Ø-22.........25=19

PEGASUS II ✓                        (NOTE 3)
   Ø-22
   3............4C=1B                 <NC3Ø>
                4C=1B 57=ØØ E9=Ø2 (V4.1)

PFS **                              (NOTE 2)
   Same as PFS REPORTS (Track Ø error
   may occur)

PFS REPORTS
   1-13
   Ø............4Ø=Ø8 41=FF 16=Ø8 19=ØØ
                58=ØB 59=FF 54=12 12=Ø2
                44=93 45=F3 46=FC 47=FF
                81=93 82=F3 83=FC 84=FF

PHANTOMS FIVE ✓
   Ø
   2-1C.........44=DD

POOL 1.5 ✓
S  Ø-15.........46=B5 79=12
S  1E-21

PRESIDENT ELECT (Revised as of 1Ø/23/81)
   Ø-22.........25=19 6B=ØØ

PUCKMAN ✓
   Ø............54=12
   1-D..........54=Ø9

PULSAR II ✓
S  Ø
S  1C.5-1D.5
S  2-C..........44=DD
S  13-19
S  1A.5-1B.5

QUICK LOADER  **
   Ø
S  2-11

RASTER BLASTER (Old version) ✓
   Ø
S  5-11 BY 4....44=AD 45=DE 53=ØØ
S  6-12  BY 4
S  7.5-F.5 BY 4
S  1.5-3.5 BY 2

RASTER BLASTER (New version) ✓
   Ø............46=96 54=12
S  5-11 BY 4....44=AD 45=DE 46=ØØ 72=ØØ
                73=ØØ 75=ØØ 78=ØØ 79=12
S  6-12 BY 4
S  7.5-F.5 BY 4
S  1.5-3.5 BY 2

RINGS OF SATURN **
S  Ø-2 SYNC
   3-22
S  5 SYNC
S  9 SYNC

SABATOGE ** ✓                       (NOTE 3)
   Ø-22
   3............4C=1B                 <NC3Ø>
                4C=1B 57=ØØ E9=Ø2 (V4.1)

SARGON II ** ✓
   Ø-1A.........19=ØØ 54=12

SHATTERED ALLIANCE
   Ø-22.........25=19

SHATTERED ALLIANCE (New)
   Ø............4C=18 47=FF 53=ØB 54=12
   1-22.........44=D4 46=B7

SINGA SHAPE MANAGER **
S  Ø-22

SNAKEBITE ** Same as GORGON

SNEAKERS ✓
   Ø............18=2Ø 19=ØØ 46=96 4D=ØØ
                4E=ØØ 52=ØØ 53=ØØ 54=12
                57=ØØ 4Ø=2Ø
S  1.5-D.5 BY 1.5.....72=ØØ 73=ØØ 77=ØØ
                78=ØØ 79=12 7C=ØØ 4Ø=2Ø
                19=ØØ 44=DD 45=AD 46=DA

SNOGGLE ** ✓
   Ø-9
        OR
   Ø-F
S  1Ø.5-11.5

SOFTPORN ADVENTURE ✓                (NOTE 3)
   Ø-22
   3............4C=1B                 <NC3Ø>
                4C=1B 57=ØØ E9=Ø2 (V4.1)

SOUTHERN COMMAND **
   Ø-22.........25=19 6B=ØØ 34=D5 35=AB

SPACE EGGS ✓
   Ø
   2-6
   11-13
   14-1A........44=DD

SPACE QUARKS ✓
   Ø............18=5Ø 19=ØØ 4Ø=2Ø 46=96
                4D=ØØ 4E=ØØ 52=ØØ 53=ØØ
                54=12 57=ØØ
   1-2..........44=AB 45=D4 46=AB
   3.5-5.5
   7
   9............44=FE 45=DD 46=AF
   A.5-B.5......44=AA 45=DE 46=BB
   D-15

SPACE WARRIOR ✓
   Ø............18=5Ø 19=ØØ 4Ø=2Ø 46=96
                4E=ØØ 52=ØØ 53=ØØ 54=12
                57=ØØ
   2.5-3.5......44=DF 45=AD 46=DE
   5-8 BY 3
   6.5
   A-1Ø BY 3

STAR CRUISER ** ✓
S  Ø-3 BY 3
S  5-B BY 1
S  11-12
S  4............44=AA 45=DD 46=BB

STAR MINES **
   Ø
   1-2..........46=AD
   4-A

STAR RAIDERS ** ✓
   Ø-5          (Track 5 error may occur)

STAR THIEF ✓                        (NOTE 3)
   Ø-13    (Track E-13 errors may occur)
   22...........4C=1B                 <NC3Ø>
                4C=1B 57=ØØ E9=Ø2 (V4.1)

SUPER APPLE BASIC **
   Ø-22
   3............(Use EXTENDED RETRY)

SUPERSCRIBE II
   Ø-22
   3............45=ØØ 5Ø=ØØ

SUPERSCRIBE II **
   Same as PEGASUS II

TAX PREPARER
   Ø-22.........46=96 54=12 4C=19
THEIF ✓
   Ø-22.........83=FF 4F=ØB 53=ØØ
S  4-5..........38=Ø2 1E=Ø2 19=ØØ 12=Ø1
                7C=ØØ

THRESHOLD                           (NOTE 3)
   Ø-22
   1-23 BY 22...4C=1B                 <NC3Ø>
                4C=1B 57=ØØ E9=Ø2 (V4.1)

TIGERS IN THE SNOW **
     Same as PRESIDENT ELECT

TIME ZONE (Disk #1)
S  Ø-4
   5-22

TIME ZONE (All other disks)
   Ø-22

TINY TROL
   Ø-22
   3.5-5 BY 1.5
```

```
TORPEDO FIRE
   Ø
   1-22.........4F=ØB

TWERPS **
   Ø............Same as GORGON
S  1.5-E.5......Same as GORGON
   1A...........4C=1B 57=ØØ E9=Ø2

ULTIMA
   Ø-22.........1E=ØB

ULYSIS **                        (NOTE 3)
   Ø-22
   3............4C=1B                <NC3Ø>
                4C=1B 57=ØØ E9=Ø2 (V4.1)

VISICALC (DOS 3.3 Version)
   Ø-22  (Error on track 1 is OK)

VISICALC (APPLE ///)
S  Ø-22

VISIDEX (Change as of 11/18/81)
   Ø-22.........4Ø=Ø4 16=Ø8 41=FF 19=ØØ
                58=ØB 59=FF 81=AA 82=EB
                83=FD 21=Ø2 46=96 54=12

VISIFILE
   Same as DESK TOP PLAN II except use
   PARM CØ=EC

VISISCHEDULE **
   Ø-22.........4Ø=Ø4 16=Ø8 41=FF 19=ØØ
                58=ØB 59=FF 81=AA 82=EB
                83=EC

VISITERM
   Ø-22
   6............4Ø=Ø8 16=Ø8 41=FF 19=ØØ
                58=ØB 59=FF 81=AA 82=EB
                83=FC

VISITREND/VISIPLOT
   Ø-22
   7............4Ø=Ø8 16=Ø8 41=FF 19=ØØ
                81=DE 82=AA 58=ØB 59=FF

WARP FACTOR **
    Same as TORPEDO FIRE

WIZARDRY (MASTER) ?              (NOTE 1,2)
   Ø-9        THEN TAK Ø →1T
   F-22          COPY IT
S  A-E..........36=1

WIZARDRY ** (SCENARIO)           (NOTE 1,2)
   Ø............36=Ø1 21=Ø2 46=96
S  1-22.........36=ØØ

ZORK (Old versions) ✓            (NOTE 3)
   Ø-22.........1E=ØB
   3............4C=1B                <NC3Ø>
                4C=1B 57=ØØ E9=Ø2 (V4.1)

ZORK I and ZORK II (New versions) ✓
   Ø-22.........46=96 4Ø=14
```

NOTE 1 Uses nibble count which requires drive speed adjustment with either the < > signs or by removing the drive cover and adjusting actual speed.

NOTE 2 Put a write-protect tab on the copy disk before using it.

NOTE 3 Patches listed below should be entered before starting copy routines unless otherwise indicated. These patches work only for version 4.Ø. They are built into version 4.1. Entering them into 4.1 will result in an 'INCORRECT PATCH' error. When using patches, enter them as shown with no spaces.

```
NAME   ENTER AS LISTED BELOW
----   -------------------------------
MULT - MULTNQNQG/RP
       This patch, when applied first,
       allows multiple patches to be
       applied.
LSGP - LSGPNJMFM/G//LA/LG/M/ZNGEKCPSWQ
       Use as directed
NC3Ø - NC3ØWKDXQQSJ/EQQSJ/EY/
       Use as directed (When PARM 4C is
       changed to 1B)
SI9K - SI9KWRTY//AWRTU/NAWRTQ//
       Allows using Silentype or
       Trendcom printers
```

NOTE 4 To Boot disk follow these steps:
1) Boot disk.
2) When computer starts to beep open drive door and press reset.
3) When computer prints "BREAK IN 5", close drive door and press reset.
4) use program.

NOTE 5 In this set of PARMs changing 4C to 1B will not invoke nibble count preservation since PARM D2 is set to ØØ.

SPECIAL NOTE: (VERSION 4.Ø ONLY) Some Apple disk drives have exhibited problems with the head seek algorithm. To determine if your Apple is affected by this problem, perform the following test:
1) Attempt to copy your Locksmith disk, track 9 only (no SYNC).
2) The copy may fail, but should indicate a track start (in the second line of the 3 line inverse block) of - D5 AD FF.
3) If this is not the case, your drives exhibit the problem, and you should apply patch LSGP after booting Locksmith.
If your drives exhibit this problem, send your 4.Ø version in for a revision to version 4.1 that solves this problem.

This parameter information is a reprint of several files maintained on the "Source". If you have any additional information, send it to:
Hardcore Computing
Copy Notes
P.O. Box 44549
Tacoma, WA 98444

INPUT

continued from page 44

software. He is entitled to a fair return (in cash) for his investment (in hours). I also believe that he is entitled to all the traffic will bear. The Yankee peddler is (thank GOD) still alive and well. Competition will manage the cost of software without any more help from you or me (or any government agency). A good many people have become quite wealthy because of the personal computer, but some can never be satisfied. The law of diminishing returns never applied (they think) to them. The marketplace will control (eventually) the price of software. The copying of overly expensive software may become a factor in that control. I am not encouraging such actions, I only note the probability. The price of software needs to be balanced against the cost. Reasonably priced software, lightly protected, seldom will be copied. If it costs you as much or more to copy it (and repro the manual) than it costs to buy it, few will copy it. There is yet another reason why original software will sell.

We are a nation of collectors. We collect everything from rocks to diamonds and all in between. Every collector wants an original, not a copy. Eventually it will be a status item among collectors to have a particular program on the original disk. Many of us realize that and often purchase software (that we already have copies of) for that reason alone. Who can find an original copy of Dr. Memory or Apple DOS 3.Ø? Remember, nobody wanted an Edsel when they were new, but now they bring a handsome price from collectors.

So please excuse a writer's runoff at the pen, keep up the good work, and if you find time to do it some day, how about an in-depth explanation of the various protection schemes, as they look when you do a sector inspection such as with Nibbles Away. The explanation that accompanies Nibbles is utterly lacking in depth.

Yours,
Russ McCaffrey Jr.
N. Torrance, CA

How To MAKE PARAMETER CHANGES

This file was lasted updated--< May 21, 1982
The parameters below can be accessed from "The Source" by typing:

TY (12)TCD328 < NIBBLESAWAY

from the command mode.

Listed below are the parameters to change in order to back-up certain pieces of software which require more than the default values given with Nibbles Away][. If a number is listed within the "less than" (>) and "greater than" (<) signs, it corresponds to the number of the Auto-Load file which will perform the listed function. To use the Auto-Load files, see Chapter 6 in the Nibbles Away][manual.

To back-up a program, first find its name in the list of parameters. Directly across from the file name is the Auto-Load file to use, if one exists. Remember that Auto-Load files are within the "less than" and "greater than" signs.

Directly below the name is a list of the tracks to copy and what parameter changes to make. If the letter "S" appears to the far left of the track number, set the "SYNC" mode before copying these tracks. If the word "BY" is used, set the increment to this value, otherwise use the default increment of 1. Parameters which are assigned values can be accessed under the Control Parameter Modifier. The parameters "ADDR" and "INS" should be entered under "ADDRESS MARK" and "INSERT MARK", respectively, in the Backup Modifier.

When the word SECTMOD appears below, it means that a sector should be changed using the Track/Sector Editor (TSE). Place the destination disk into drive 1, then perform the changes listed. The command format is:

SECTMOD [. F=NN,C=XX,S=YY,T=ZZ]
CHANGE ADDRESS A1 FROM A2 TO A3

The meaning of NN,XX,YY,ZZ and A1,A2,A3 are explained below:

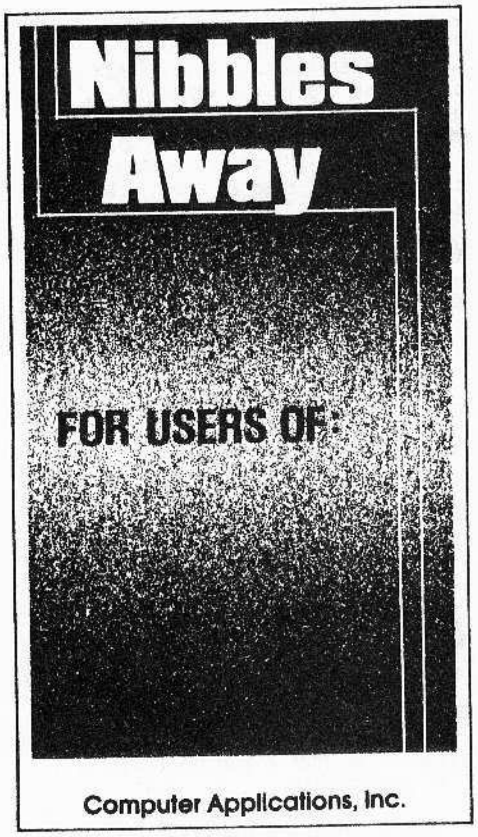
NN—This will be either 13 or 16, and represents the disk format to be used. This should be set by selecting the "O" option in the TSE, then pressing 'F' until the proper format is shown in inverse.

XX—This will be either on or off, and should be set using the Checksum option on the Options page, as above ('C' to toggle).

YY—This is the sector to be read.

ZZ—This is the track to be read. (See Nibbles Away][manual for details on how to set these)

After setting these options, use the "R" option to read the given sector into the



buffer. Then change the information in the sector, following the convention listed below:

A1---This is the location to be changed in the buffer.

A2—This is the old value.

A3—This is the new value.

If multiple changes are listed, they should be performed in sequence. After making changes to a sector, it should be written back to the disk with the "W" option.

Note: Parameters from Nibbles Away I may be used in Nibbles Away][. They must be entered using the name of the desired parameter listed in the Nibbles Away I manual. Nibbles Away I parameters may not be entered under the global modifier in Nibbles Away][.

The following example shows a file that incorporates many changes. Step-by-step directions will be given on how to copy this program. Line numbers have been added to each line for reference. These numbers do not appear in the parameter list.

```
1) EXAMPLE FILE (XXX)
2) Ø-5..............ADDR=D5 AA 96
3)       SECTMOD [F=16,C=OFF,T=ØØ,S=Ø3]
4)       CHANGE ADDRESS 42 FROM 38 TO 18
5)       OVERIDE STANDARDIZER
6) S 6-9 BY 1.5.....ADDR=DD AD DA
8)    11-22 BY 2.... INS=AD FB E6 FF E6
9)                   SYNC SIZ=ØA
1Ø)                  DATA MAX=25
```

Line 1 contains the name of the program ('EXAMPLE FILE'), and the abbreviation of the company that markets the program ('XXX') in parentheses. The name of the company can be found in the table of abbreviations.

Line 2 gives the tracks to copy and what parameter changes to make before copying the tracks. First set the ADDR parameter to D5 AA 96. These changes are made from the Backup menu. To get to the Backup menu first enter the Modify Parameter menu ('M" from the main menu), then press "B."

Line 3 shows that some special changes must be made to a certain sector (track ØØ, sector 3). Enter the TSE and use the "O" command to select the following options:

F. . . Set up for 16 sector.
C. . . Turn the CHECKSUM FLAG off.

Track ØØ, sector 3, needs to be read. Type "T" then ØØ, and "S" then 3. Use the "R" command to read the proper track/sector.

Line 4) After the sector is in memory, change address 42 from 38 (what it originally was) to 18. Write the track back to the disk using the "W" command.

Line 5) Enter the Backup menu. From this menu answer yes ('Y') to the question "OVERIDE STANDARDIZER."

Line 6) Set the synchronized copy mode when copying tracks 6-9. The "S" means make a synchronized copy. Also set the increment to 1.5.
Before copying tracks 6-9, set the ADDR to DD AD DA.

Line 8) Copy tracks 11-22 with an increment of 2. Set INS to AD FB E6 FF E6. The changes to INS are made from the Backup menu.

Line 9) Set the SYNC SIZe to ØA. To change SYNC SIZes enter the Modify Parameter menu, and from this menu enter the Control menu ('C'). Use the arrow keys to move the cursor to the SYNC SIZe (2nd column, third one down) and press the space bar to change this value, in our case to ØA.

Line 1Ø) The DATA MAX value is changed in the same manner as the SYNC SIZe.

The file is now copied (yea!).

NOTE: The following company names and software titles are copyrighted and/or trademarked. An abbreviation of the company which reserves these rights follows each software title. A list of the abbreviations and the name of the company each represents may be found at the end of the parameter lists.

```
Program Name (Company)  (Auto-load file>
Tracks to Copy   Parameters to change
-----------------------------------------
ACCOUNTING SYSTEM (BPI)                <15>
   Ø-22..........ADDR=D5 AA 96
    11-11.........INS=AD FB E6 FF E6
                  SYNC SIZ=ØA

APPLE PANIC (BS) ✓
   Ø-D

APPLE WORLD (USA) ✓
 Ø- 23

APPLE WRITER /// (APC)
S  Ø-22

AUTOBAHN (SRS) ✓                        <6>
S Ø-Ø
S 4-6
S 9.5-C.5

A2-FS-1 (SBL) ✓                        <13>
  Ø-Ø
  1.5-21 BY 1.5..ADDR=DB AB BF
                 REDUCED ERROR CHECK
  7-8............REDUCED ERROR CHECK
  9.5-9.5....... REDUCED ERROR CHECK

A2-PB1 PINBALL (SBL)
  Ø-Ø............ADDR=D5 AA 96
                 DATA MAX=25
  1-15...........ADDR=DB AB BF

BEER RUN (SRS) ✓
S Ø-Ø............ADDR=DD AD DA
                 DATA MAX=25
S 1.5-13.5
  NOTE: Errors will begin to occur some-
where between track C.5 and track 13.5,
depending on the particular disk.  This
is normal.

BORG (SRS) ✓
S Ø-Ø............ADDR=DD AD DA
S 1.5-B.5
S D-2Ø

CANNONBALL BLITZ (OLS) ✓
  Ø-22...........ADDR=D5 AA 96
         SECTMOD [F=16,C=ON,T=17,S=ØE]
         CHANGE ADDRESS CD FROM 49 TO 6Ø

CASINO 21 (DM)
   Ø-22..........ADDR=D5 AA 96
         SECTMOD [F=16,C=OFF,S=Ø3,T=ØØ]
         CHANGE ADDRESS 63 FROM 38 TO 18

CEILING ZERO (TKS)
  Ø-2............ADDR=D5 AA B5
  3-11...........ADDR=D6 AA B5
             INS=DE AA EB F9, SYNC SIZ=ØA

COPTS & ROBBERS (SRS) Same as Beer Run ✓

COUNTY FAIR (DM) ✓
   Ø-22..........ADDR=D5 AA B5
```

```
CRANSTON MANOR (OLS) ✓
  Ø-22..........ERASE DEST TRACKS

DARK FOREST (SRS) ✓
S Ø-Ø...........ADDR=DD AD DA
S 1-22..........ADDR=D5 AA A5
  (Errors on 6-8 and last few tracks OK)

DB MASTER (OLD) (STW)                 <9>
  Ø-5...........ADDR=D5 AA 96
  6.5-22.5

DB MASTER (NEW) (STW)                <19>
S Ø-5...........ADDR=D5 AA 96
  6.5-22.5

DEADLINE (IC)                  <DOS 3.3>
   Ø-22.........ADDR=D5 AA 96

DESKTOP PLAN II (VCP)                <16>
 Ø-22...........ADDR=D5 AA 96
                INS=AA EB FD
                SYNC SIZ=ØA, FIX AMNT=Ø4

DUNG BEETLES (DS) ✓
   Ø-Ø...........ADDR=D5 AA B5
   1-1...........ADDR=F5 F6 F7
   4-22
         SECTMOD [F=13,C=ON,T=ØØ,S=Ø1]
         CHANGE ADDRESS 6D FROM Ø1 TO 7B
         CHANGE ADDRESS 6E FROM 61 TO 69

ELIMINATOR (AI) ✓
   Ø-21.........ADDR=D5 AA 96
         SECTMOD [F=16,C=OFF,T=Ø3,S=ØD]
         CHANGE ADDRESS 2E FROM 2Ø TO EA
         CHANGE ADDRESS 2F FROM 3Ø TO EA
         CHANGE ADDRESS 3Ø FROM 72 TO EA

EPOCH (SRS) ✓Same as Beer Run ✓

ESCAPE (SBL)                   <DOS 3.3>
  Ø-22......ADDR=D5 AA 96

ESCAPE FROM ARCTURUS (SNS) ✓
  Ø-22...........ADDR=D5 AA 96
                 OVERIDE STANDARDIZER
                 OVERIDE NIBBLE FILTER

EXECUTIVE SECRETARY (PBS)
  Ø-22...........ADDR=D5 AA 96
   DOS 3.3

EXPEDITER II (OLS)                    <2>
  Ø-22...........ADDR=D5 AA 96
                 ERASE DEST TRACKS

FIREBIRD (GS) ✓                       <7>
S  Ø-Ø...........ADDR=DD AD DA
S  1.5-B.5

GAMMA GOBLINS (SRS)
S Ø-Ø............ADDR=DD AD DA
S 1.5-B.5
S D-D............ADDR=FF FF FF D5 AA EE
                 DATA MAX=3Ø

GENETIC DRIFT (BS) ✓
   Ø-Ø...........ADDR=D5 AA B5
   1-3...........ADDR=BB D5 BB
   4.5-6 BY 1.5
   7.5-B.5
   D-D...........ADDR=D4 D5 BB
   E.5-12.5......ADDR=AD B5 DE
```

```
GOBBLER (OLS) ✓                       <1>
  Ø-22...........ADDR=D5 AA B5
                 ERASE DEST TRACKS

GOLD RUSH (SS)                  <DOS 3.3>
  Ø-22...........ADDR=D5 AA 96

GORGON (SRS) ✓
S Ø-Ø............ADDR=DD AD DA
                 DATA MAX=25
S 1.5-C.5
S E.5-E.5
S D.5-D.5........ADDR=D5 AA B5

GUARDIAN (CTS)
   Ø-1...........ADDR=D5 AA B5
    2-11.........ADDR=D6 AA B5
           INS=DF AA EB F7, SYNC SIZ=ØA

HADRON (SRS) Same as Beer Run

HIRES ADV #1 (OLS) ✓            <DOS 3.2>
  Ø-22...........ADDR=D5 AA B5

HIRES ADV #2 (OLS) ✓            <DOS 3.2>
  Ø-22...........ADDR=D5 AA B5

HIRES CRIBBAGE (OLS) ✓               <2Ø>
S Ø-22...........ADDR=D5 AA B5

INTERNATIONAL GRAND PRIX (RBS) ✓
  Ø-C............ADDR=FF FF FF AA

INVOICE FACTORY (ML)            <DOS 3.3>
  Ø-22...........ADDR=D5 AA 96

JABBERTALKY (MT)                <DOS 3.3>
  Ø-22...........ADDR=D5 AA 96

JAW BREAKER (OLS) ✓                   <1>
  Ø-22...........ADDR=D5 AA B5
                 ERASE DEST TRACKS

LETTER PERFECT (LJK)            <DOS 3.2>
   Ø-22..........ADDR=D5 AA B5

MASTER TYPE (LNS) ✓
   Ø-2...........ADDR=D5 AA B5
   3-22..........ADDR=D4 AA B5
                    (ERROR ON $1B OK)
         SECTMOD [F=13,C=OFF,S=Ø3,T=ØØ]
         CHANGE ADDRESS 63 FROM 38 TO 18
         SECTMOD [F=13,C=OFF,S=ØA,T=Ø2]
         CHANGE ADDRESS 2E FROM 23 TO 2E

MICROWAVE (CC) ✓
   Ø-22..........ADDR=D5 AA 96
         SECTMOD [F=16,C=ON,T=Ø2,S=Ø1]
         CHANGE ADDRESS DA FROM A9 TO AD
         CHANGE ADDRESS DB FROM 6Ø TO Ø3
         CHANGE ADDRESS DC FROM 8D TO 81
         CHANGE ADDRESS DD FROM 7E TO 6Ø

MISSILE DEFENSE (OLS) ✓              <2Ø>
S Ø-22...........ADDR=D5 AA B5

MOUSKATTACK (OLS) ✓
  Ø-22...........ADDR=D5 AA 96
         SECTMOD [F=16,C=ON,T=18,S=Ø3]
         CHANGE ADDRESS B1 FROM 49 TO 6Ø

NEUTRONS (L1Ø)                        <1>
  Ø-22...........ADDR=D5 AA 96
```

```
ORBITRON (SRS) ✓
S Ø-Ø............ADDR=DD AD DA
                 DATA MAX=25
S 1.5-E.5
  F.5-F.5........ADDR=FF B5 D5 AA

OUTPOST (SRS) ✓
S Ø-Ø............ADDR=DD AD DA
S 1.5-9.5
  B.5-B.5........ADDR=D5 AA AD
                 DATA MAX=25

PADDLE GRAPHICS (OLS)                  <2Ø>
  Ø-23...........ADDR=D5 AA B5
S 2..............ADDR=D5 AA B5

PEEPING TOM (ML)
  Ø-Ø............ADDR=D5 AA B5
  1-1............ADDR=F5 AB BE
  4-22
        SECTMOD [F=13,C=ON,T=ØØ,S=Ø1]
        CHANGE ADDRESS 6D FROM Ø1 TO 7B
        CHANGE ADDRESS 6E FROM 6Ø TO 68

PEGASUS II (OLS) ✓                     <1>
  Ø-22...........ADDR=D5 AA B5
                 ERASE DEST TRACKS

PERSONAL FINANCE MANAGER (SDS) <DOS 3.3>
  Ø-22...........ADDR=D5 AA 96

PHOTAR (STP)                     <DOS 3.3>
  Ø-22...........ADDR=D5 AA 96

POOL 1.5 (IDS) ✓
  Ø-15..........ADDR=D5 AA B5
  1E-21
        SECTMOD[F=13,C=OFF,T=ØB,S=Ø7]
        CHANGE ADDRESS 6A FROM 8D TO 6Ø
        SECTMOD[F=13,C=OFF,T=ØØ,S=Ø3]
        CHANGE ADDRESS 63 FROM 38 TO 18

PULSAR II (SRS) ✓                      <18>
  Ø-C
  13-19
  1A.5-1D.5

RASTER BLASTER (BC) ✓
S  Ø-Ø..........ADDR=D5 AA 96
                DATA MIN=18, DATA MAX=4Ø
S  5-11 BY 4.....ADDR=AD DE, DATA MIN=13
S  6-12 BY 4
S  7.5-F.5 BY 4
S  1.5-3.5 BY 2

RICOCHET (MT) ✓                  <DOS 3.3>
  Ø-22...........ADDR=D5 AA 96

ROACH HOTEL (ML)
  Ø-Ø............ADDR=D5 AA B5
  1-1............ADDR=EE EA FE
  4-22
        SECTMOD [F=13,C=OFF,T=ØØ,S=Ø1]
        CHANGE ADDRESS 75 FROM Ø1 TO 7B
        CHANGE ADDRESS 76 FROM 61 TO 69

SNACK ATTACK (DM) ✓
  Ø-22..........ADDR=D5 AA B5
        SECTMOD [F=13,C=OFF,S=Ø3,T=ØØ]
        CHANGE ADDRESS 63 FROM 38 TO 18

SNAKE BYTE (SRS) Same as Beer Run
```

```
SNEAKERS (SRS) ✓
S Ø-Ø............ADDR=DD AD DA
S 1.5-C.5
S D.5-D.5........ADDR=D5 AA B5 ✓

SOFTPORN ADVENTURE 3.2 (OLS) ✓         <1>
  Ø-22...........ADDR=D5 AA B5
                 ERASE DEST TRACKS
SOFTPORN ADVENTURE 3.3 (OLS) ✓         <2>
  Ø-22...........ADDR=D5 AA 96
                 ERASE DEST TRACKS

SPACE QUARKS (BS) ✓
  Ø-Ø............ADDR=D5 AA B5
  1-2............ADDR=FF DF DE
                 DATA MAX=25
  3.5-5.5
  7-9 BY 2
  A.5-B.5
  D-15

SPACE WARRIOR (BS) ✓
  Ø-Ø............ADDR=D5 AA B5
                 DATA MAX=3Ø
  2.5-3.5.......ADDR=DF AD DE
  5-8 BY 3
  6.5-6.5
  A-1Ø BY 3

STAR BLASTER (PDS)
  Ø-Ø............ADDR=D5 AA 96
  7-2Ø BY 1.5....ADDR=DF AD DE

STAR DANCE (USA) ✓               <DOS 3.2>
  Ø-22...........ADDR=D5 AA B5

SUICIDE (PDS) ✓
  Ø-Ø............ADDR=D5 AA B5
  11.5-22 BY 1.5.ADDR=DF AD DE

SWASHBUCKLER (DM) ✓
  Ø-22...........ADDR=D5 AA 96

TAX PREPARER (HS)                <DOS 3.3>
  Ø-22...........ADDR=D5 AA 96

THRESHOLD (OLS)                        <1>
  Ø-22...........ADDR=D5 AA B5
                 ERASE DEST TRACKS

TIME ZONE  V1.Ø  (OLS)
  DISKS A-L
  Ø-22...........ADDR=D5 AA 96
                 OVERIDE STANDARDIZER
  THEN DISK A
           SECTMOD [F=16,C=ON,T=Ø3,S=Ø5]
           CHANGE ADDRESS 5B FROM 4C TO 6Ø
           SECTMOD [F=16,C=ON,T=Ø3,S=Ø3]
           CHANGE ADDRESS AB FROM A9 TO 6Ø

TIME ZONE  V1.1  (OLS)
  DISKS A-L
  Ø-22...........ADDR=D5 AA 96
                 OVERIDE STANDARDIZER
  THEN DISK A
           SECTMOD [F=16,C=ON,T=Ø3,S=ØØ]
           CHANGE ADDRESS D9 FROM FC TO ØØ
           CHANGE ADDRESS ******* Ø8 TO 13

TORPEDO FIRE (SSM)
  Ø-22.......... ADDR=D4 AA B7

TUNNEL TERROR (MS)
  Ø-Ø............ADDR=D5 AA B5
  1-12...........ADDR=D6 AA B5
             INS=DF AA D7 EB, SYNC SIZ=ØA
```

```
TWERPS (SRS)
S Ø-Ø............ADDR=DD AD DA
S 1.5-E.5
  1A-1A

ULYSSES & GOLDEN FLEECE (OLS)          <2>
  Ø-22...........ADDR=D5 AA 96
                 ERASE DEST TRACKS

VISICALC /// (APC)
S  Ø-22

VISICALC 3.3 (VCP)
  Ø-Ø............ADDR=D5 AA 96
  2-22...........ADDR=D5 AA B5
                 (ERRORS TOWARD END OK)

VISIDEX (VCP)
  Ø-22...........ADDR=D5 AA 96
                 INS=DE AA EB FD
                 SYNC SIZ=ØA, FIX AMNT=Ø4

VISIFACTORY (ML)
  Ø-22...........ADDR=D5 AA 96
          SECTMOD [F=16,C=OFF,T=ØØ,S=Ø3]
          CHANGE ADDRESS 42 FROM 38 TO 18
          SECTMOD [F=16,C=OFF,T=Ø1,S=ØØ]
          CHANGE ADDRESS 84 FROM 4C TO AD
          CHANGE ADDRESS 85 FROM 8E TO E9
          CHANGE ADDRESS 86 FROM AE TO B7

VISIFILE (VCP)
  Ø-22...........ADDR=D5 AA 96
                 INS=DE AA EB
                 SYNC SIZ=ØA, FIX AMNT=Ø4

VISISCHEDULE (VCP)
  Ø-22...........ADDR=D5 AA 96
                 INS=DE AA EB
                 SYNC SIZ=ØA, FIX AMNT=Ø4

VISITERM (VCP) ✓
  Ø-22...........ADDR=D5 AA 96
                 INS=DE AA EB FC
                 SYNC SIZ=ØA, FIX AMNT=Ø4

VISITREND/VISIPLOT (VCP)
  Ø-22...........ADDR=D5 AA 96
                 INS=DE AA EB
                 SYNC SIZ=ØA, FIX AMNT=Ø4

WORD HANDLER II (SVS)
  Ø-Ø............ADDR=D5 AA 96
  11-22
  1-C............ADDR=FF DF DE

ZERO GRAVITY PINBALL (AGC)       <DOS 3.2>
  Ø-22..........ADDR=D5 AA B5

#######################################
```

Table of Abbreviations of Publishers

AC	APPLE COMPUTER
AGC	AVANTE GARDE CREATIONS
AI	ADVENTURE INTERNATIONAL
BC	BUDGCO
BPI	BPI
BS	BRODERBUND SOFTWARE
CC	CAVALIER COMPUTER
CTS	CONTINENTAL SOFTWARE
DM	DATA MOST
DS	DATA SOFT

GS	GEBELLI SOFTWARE
HS	HOWARDSOFT
IC	INFOCOM
IDS	INNOVATIVE DESIGN SOFTWARE
LJK	LJK ENTERPRISES
L10	LEVEL 10 SOFTWARE
LNS	LIGNTNING SOFTWARE
MS	MAGNA SOFT
ML	MICRO LAB
MT	MIND TOYS
OLS	ONLINE SYSTEMS
PBS	PERSONAL BUSINESS SYSTEMS
PDS	PICADILLY SOFTWARE
PST	PROFESSIONAL SOFTWARE TECHNOLOGY
RBS	RIVERBANK SOFTWARE
SRS	SIRIUS SOFTWARE
SS	SENTIENT SOFTWARE
STP	SOFTAPE
SVS	SILICON VALLEY SOFTWARE
TKS	TURNKEY SOFTWARE
USA	USA SOFTWARE
VCP	VISICORP

This parameter information is a reprint of several files maintained on the "Source". If you have any additional information, send it to:
Hardcore Computing
Copy Notes
P.O. Box 44549
Tacoma, WA 98444

INTERVIEW

continued from page 13

MARKKULA: When you look at it from an industry basis, you have to look at volume, not just the software pirate industry which is very cliquish. And you have to question how many copies are being made, and are legitimate potential customers turning to the copiers in preference to buying the program. And if they are, what percentage of them are . . . and why are they? Do they feel that the price is too high for what the manufacturers are offering? Do they feel that they are being ripped off? Do they prefer working with the people who offer the copies? Do they find it too difficult to find the original product? Or do they learn of the product through the copying network and that's why they get it that way instead of seeing it at a computer store where they can buy it legitimately? There's just a million questions that needs to be asked.

What people have to realize is that we have to put some work into the solution. It's not just going to be presented and then everything is going to be fine. We're a responsible industy and we have to work at it.

HARDCORE: And right now you feel that you are doing your part in uncovering the solutions?

MARKKULA: We're working at it. I hope that we'll be successful, that we'll be the guys who will present a workable solution or help to create a workable solution. I think that it would be a feather in our cap and one thing that we could then say about Apple is that we TRY to do things that are right for the industry.

HARDCORE: Is there anything that you would like to add, especially regarding the quotes used by major computer magazines?

MARKKULA: Only that it was taken out of context. It is true. We are against software protection. We don't think that in the long term it is the best thing for the industry or the customer. On the other hand, it is not possible for us to propose a fully acceptable solution at this time. So, in light of that, we're going to work on it. And we'd like to have anyone else who wants to help assist us in trying to put together an organization that will come up with a solution.

WHY YOU NEED THE INSPECTOR.

If you're serious about programming, you need to set all your utilities together in one place—*inside* your Apple. The Inspector comes on an Eprom that simply plugs into the D8 socket, or on a disk ready to merge with Integer Basic for automatic loading on boot. Either way, it stays at your fingertips, ready to call without disturbing your current program.

The Inspector puts you in total control of both memory and disks. You can search forward and backwards, edit, read nibbles, map disk space, dump the screen to a printer, examine every secret of your Apple. Use The Inspector to repair blown disks, undelete files, input "illegal" commands, read and alter files, locate strings in memory or on disk. The uses are endless. The manual, alone, is an education. And it's always *there* when you need it.

You need the most powerful disk and memory utility available for your Apple. You need the Inspector.

See your local dealer, or order direct for just $49.95. Mastercard and Visa holders order toll-free, 1-800-835-2246.

OMEGA MICROWARE, INC.
222 SO. RIVERSIDE PLAZA
CHICAGO, IL 60606
312-648-1944

Apple is a registered trademark of Apple Computer, Inc.

The following is a reprint of a document describing the procedure for backing up disks which cannot be copied with the standard Nibbles Away][parameters. This information can be found on "The Source." (a Modem-based data communications network.) Access the file from command by entering: **<TY (12)TCD328<NAII.**

This information has been edited for publication.

HOW TO MAKE BACKUPS: A STEP BY STEP GUIDE

There are three basic steps to back-up a diskette:

1. Locate the tracks which contain data.

2. Find the address marker for the sectors there.

3. Determine if any additional protection is used. (This is the hard one!)

Track/Bit Editor

For most of the procedures below, a basic working knowledge of the Track/Bit Editor (TBE) is required. For those who are not familiar with the TBE, an overall description and some examples are given below.

The examples are easier to understand if they are performed while reading the instructions, so boot Nibbles Away][and try them out to get a better understanding of what is going on.

Enter the TBE by selecting option "T" from the main menu. A large section of numbers will appear on the screen, with two dashed lines at the top. The information between these lines is the status information. It shows such things as cursor position and track number. It is also the location where various prompts appear for certain functions. The numbers at the bottom are separated into two sections. On the left are the starting memory addresses for each line to the right.

Move the cursor around using I,J,K or M, and watch the ADDR indicator in the status line. It will show exactly what memory address the value under the cursor represents.

The arrow keys change the area of memory that can be seen. They shift the view 256 bytes forward or backward at a time. The only really important thing to know for this discussion is how to use the arrow keys to move the viewing 'window' around in memory.

The ";" (unshifted "+") and the "-" keys increment and decrement the track number in the status line.

Pressing "R" will cause drive one to read the data from the track indicated in the status line into memory. The bytes on the screen will change, since different data has been read. Pressing the "R" key multiple times will result in different data being displayed. This is because Nibbles Away][starts reading at whatever point happens to be under the Read/Write head when the drive is turned on. The data is not actually different, it is just not loaded at the same memory location as it was previously.

Step 1: Locate the Tracks with Data

To begin, the track pointer should be set to track $00. Pressing "R" will read the track and show it on the screen. The arrow keys should be used to move the viewing 'window' to start at $2000.

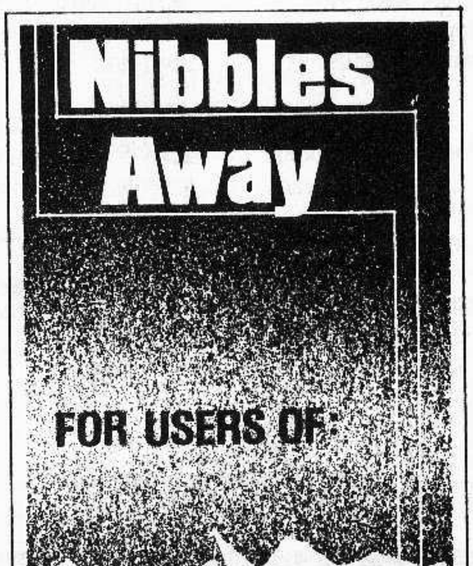
Now move forward and try to determine if this track contains valid data. Actually, track $00 must contain some data in order for the disk to boot, but we will be using this procedure on other tracks which do not necessarily contain data.

Gaps

The main thing which will identify a track as containing data is the presence of gaps. Gaps are sections of the same byte repeated several times. Normally they are made up of $FF's and are 6-20 bytes in length. To see what these look like, insert the system master disk and read track $00 as described above.

Moving through the buffer with the arrow keys will reveal a large variety of values. Spaced among these should be sections of 6-20 FF's in a row, depending on the exact disk. Normally DOS 3.2 disks have larger gaps than DOS 3.3 disks. There should be many gaps, spaced so that one is seen about every other time the arrow keys are used to move forward or backward.

Note: A second, smaller (2-5 $FF's) gap may be seen following a large gap, with a small section of data in-between. This is called the secondary gap. When referring to a gap here, the allusion will be to the primary gap, not the secondary one.



Full/Half Tracks

Now try looking at other tracks on the disk. First look only at the full tracks (no ".5" on the end). All of them will be similar to track $00 in the appearance of the gaps. Try this several times to become comfortable with locating gaps on a given track.

Now read a half-track (".5" on the end). Scan memory to locate some of the gaps. Since System Master disks do not use half-tracks, the data which is seen is really 'cross-talk'. In other words, data was written on the full track, but the magnetic pattern spread out a bit, so some data is seen here. The tell-tale sign of this phenomenon is that the gaps will not all be the same. That is, they may contain one or more values which are inconsistent. This reveals that there is some data on the track, but that it is not valid data. Take a look at other half-tracks, until half-tracks and full tracks can be discerned by examining the gaps.

Blank Tracks

The next item to be able to identify is a blank track. To do this, insert a blank (non-initialized) disk into drive 1. Read any track on this disk and scan through the memory addresses. There will be no gaps, and many of the bytes will end in "0" (IE. $A0, $B0, $E0), which are not legal disk bytes. This means that the controller can find no valid data on the track. Some disks have portions of tracks which are not used, so always be sure to examine at least 24 screens full of information to make sure that there is no data at any point on the track.

The next tool for finding data is the fact that valid data must be at least 1 track apart. In other words, if data is located on track 3.5, track 4 cannot have data and the next place where data can be is track 4.5. This is very helpful for finding tracks with data.

Note: If data is located on a given track, it is a good idea to look at the tracks one half-track to either side, to make sure that they look less valid than the track selected as the real one.

Now that valid data can be recognized, begin at track $00 and step towards track $22, checking each track to see if it appears to have data on it. Most disks have a pattern to the position of the data, and if that pattern can be figured out it may be possible to just check a few tracks to make sure, and then go on to step 2. Otherwise the data must be located one track at a time.

Most disks use the standard tracks (1,2,3, . . . ,22), but there are some which use half-tracks and some which use track $23 (which cannot be read on all drives since Apple drives were not designed to go out that far).

When all tracks which contain some type of data are located, go on to step 2.

Step 2: Find the Address Markers

Now tell Nibbles Away][how to read the information on the tracks which have been found to contain valid data. This is done by going back to each of these tracks with the TBE and finding the address mark for each one. The address mark will be the first 3 bytes following the gap. To see this in operation, take a look at a track from the System Master disk. After each gap either 'D5 AA 96' for a DOS 3.3 Master disk, or 'D5 AA B5' for a DOS 3.2 disk will be seen. These values should be noted alongside of each track number which contains data. Many times there will be only one, or maybe two patterns for all tracks.

After this, these tracks can be copied. This is done by exiting the TBE (use "Q') and then selecting "M" for the modifiers menu. Then select "B" for Backup Modifier. When asked "USE ADDRESS MARK', answer "Y" and then type in the address mark that was noted for the range of tracks to be copied. Simply press >return< to the rest of the questions and then return to the main menu.

Select "N" to enter Nibbles Away][, and answer "Y" to the question "CHANGE DEFAULT OPTIONS." Use the >RETURN< key to move to the "START TRACK" prompt, and then enter the first track to be copied. Press >return< and then type in the last track to be copied with the current address marker setting.

If the tracks in the specified range are not spaced at 1-track intervals, enter the interval at the "TRACK INCREMENT" prompt. Press >return< for the following questions, and begin the copy after inserting the disks (when prompted). After returning to the main menu, repeat the above procedure for each range of tracks which contains a different address marker.

Now comes the moment of truth! Try to Boot the copy disk (if the original had a write-protect tab, the copy should too!). If the copy boots, then all went successfully.

Step 3: Determine Additional Protection

If the back-up did not work properly, there are a few things to look for.

1 . . . Did all of the tracks which should have copied do so? This can be seen while the copy takes place as a "Y" or an "N" under that track status location. If some did not, then the address marker was probably not determined properly. If this is the case, go back to the TBE and try those tracks again.

2 . . . If everything seemed to go well, but the copy refuses to work (it might help to try the procedure again, maybe with the source and destination drives reversed, to make sure it was not a power glitch or other such occurrence which messed things up), the next step is to try the procedure with the "SYNCHRONIZED COPY" option selected. Disks which use this method often make violent head movements during their boot procedure. This can be a clue to this type of protection.

Additional Information:

On some DOS 3.3 diskettes, the gaps between the sectors are reduced in size. In some cases they can be as small as 4 or 5 bytes. When Nibbles Away][finds the beginning of a section of data, it normally adds 8 bytes of sync just before the data. This will normally put sync bytes into the gap before the data, where it should be. However, if a disk has very small gaps, then the added sync can overwrite the end of the previous sector. The parameter FIX AMNT controls the number of sync bytes which are added, so this value can be reduced to prevent any data from being overwritten. The value that Nibbles Away][uses for the sync which it puts in is contained in the parameter FIX VALU. Normally this is a $7F, but it can be set to any desired value.

It should be noted that Nibbles Away][regards any data byte which has its high bit cleared to be a sync byte. So the $7F which is normally in this parameter means that a sync $FF is to be added. If the "OVERIDE STANDARDIZER" option is selected, then Nibbles Away][will not add any bytes, it will simply convert the data which is present before a sector into sync, without changing its value. This technique can also be used for disks whose gaps are very small.

Long Tracks

Another item to watch for is disks whose tracks appear to be very long. Some disk protection schemes put garbage on a portion of the track. When this garbage is read back, more bytes are read in than were written out. This causes the track to be longer than normal, and in some cases it becomes so long that the default parameters for Nibbles Away][cannot find the data properly.

DATA MIN/MAX

The parameters DATA MIN and DATA MAX control the minumum and maximum track lengths (in increments of 256 bytes) which Nibbles Away][will accomodate. The normal value of DATA MAX is $1D, but this can be set to a higher value, such as $25, if a track appears to be very long. Even though the track may read a large number of bytes, many of these will be removed by the Nibble Filter, since they are garbage bytes. This will assure that the amount of data written back will not be too large to fit on the destination track.

When Nibbles Away][finds a sector of data, it looks ahead to find a second occurrence of the same pattern. This insures that the sector has been read and located correctly. On many disks, there is a primary section of data, called the address field, and the the actual data field follows. In-between these is a small gap, which often contains random information. This means that Nibbles Away][should only match the number of bytes which are found in the address field, since the bytes in the gap may not read as the same value every time.

FIND MAX

The parameter FIND MAX controls the number of bytes which are checked during this procedure. The default value of $0C works in most cases, but some disks use a smaller address field which may require this parameter to be set to a smaller value. However, if this parameter is set too low, then Nibbles Away][may identify the match for a section of data whose first few bytes are the same, but which differ later on. Therefore one should exercise caution when lowering this value.

CHR$(4): HYPER D.O.S.

continued from page 27

```
9C91- C8                    INY              POINT TO SECTOR
9C92- 11 FB                 ORA (PTR),Y      BOTH TRACK AND SECTOR Ø?
9C94- FØ 3E                 BEQ SKIP2        YES, GOTO PARTIAL READ ROUTINE
9C96- 2Ø E9 9C              JSR READ         NO, SO READ IN THE ENTIRE SECTOR
                  *---------------------------------
                  * SEE IF ANOTHER T/S LIST MUST BE READ IN FROM THE DISK
                  *---------------------------------
9C99- A4 FF                 LDY TEMP         GET T/S COUNTER
9C9B- DØ E1                 BNE LOAD.IT      IF NOT ZERO THEN EXIT
9C9D- C8                    INY
9C9E- B1 FB                 LDA (PTR),Y      GET TRACK OF T/S LIST
9CAØ- DØ Ø3                 BNE NOT.DONE     IF IT ISN'T ZERO THEN READ IT IN
9CA2- 4C EA A2              JMP CLOSE        OTHERWISE WE ARE DONE
                  *---------------------------------
                  * READ ANOTHER T/S LIST
                  *---------------------------------
9CA5- 8D EC B7    NOT.DONE  STA RW.TRK       SAVE TRACK NUMBER
9CA8- C8                    INY
9CA9- B1 FB                 LDA (PTR),Y      GET SECTOR NUMBER
9CAB- 8D ED B7              STA RW.SCT       SAVE IT
9CAE- AD FØ B7              LDA RW.BUF       SAVE CURRENT BUFFER ADDRESS
9CB1- 48                    PHA              BY PUSHING THE ADDRESS
9CB2- AD F1 B7              LDA RW.BUF+1     ONTO THE STACK
9CB5- 48                    PHA              AND
                  *---------------------------------
9CB6- A5 FB                 LDA PTR          RESET BUFFER TO NORMAL
9CB8- 8D FØ B7              STA RW.BUF       TO PREPARE FOR READING THE
9CBB- A5 FC                 LDA PTR+1        NEW T/S LIST
9CBD- 8D F1 B7              STA RW.BUF+1     THEN
```

continued on facing page

MENU

continued from page 35

575: If the first character of A$ is an asterisk (*), change the LOCK option to UNLOCK and set the LOCK flag (L = 1).

58Ø: Use the second character of A$ to set the file type flag. If it is binary, change the options from LOAD to BLOAD, RUN to BRUN, and set the Binary flag (B = 1).

585: If the file type is Text, change RUN option to EXEC and set the Text flag (B = 2).

59Ø-61Ø: Print all options available. Do not print the LOAD option if the file is Text.

615-63Ø: Prepare for future errors and redefine A$ as just the file name (the status is removed).

65Ø: GET choice.

2) LOCK FILE (66Ø-7ØØ)

This routine will LOCK a file only if the user confirms the action and the LOCK flag is set.

66Ø: Check for the "K" command (LOCK). If not "K", then go to routine 4 (UNLOCK).

continued on facing page

REPLAY

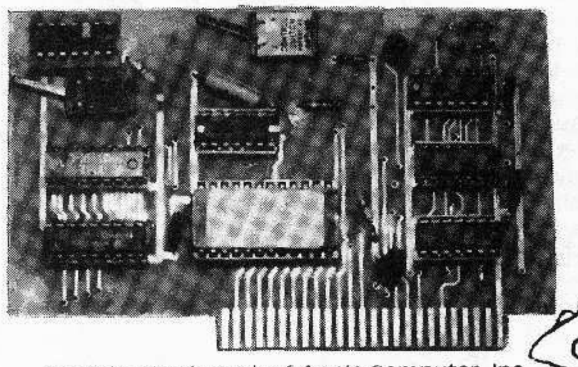
QUICK DEPENDABLE BACKUP COPIES FOR THE APPLE II

Replay does not copy a disk, rather it copies a program executing in memory. The term total load describes a program that fully loads into memory to execute.

The original disk formatting is not important. If the program will boot, it is possible to make a backup copy that executes in 15 seconds.

The backup copy is in a different format. With the card comes an editor utility that operates under DOS 3.3. This allows editing of the copy disk. A second utility will pack the copy disk and put it into an executable DOS 3.3 binary file.

- COPIES TOTAL LOAD PROGRAMS
- DISK FORMATTING IRRELEVANT
- COPY IN JUST 15 SECONDS
- PROGRAM RESTART IN 10
- TRANSPARENCY SWITCH



Apple is a trademark of Apple Computer, Inc.

COST	CONFIGURATION
$129.95	Fully Assembled Card W/Utility Disk
$ 89.95	Kit (Wire Wrap)
$ 59.95	EPROM Plus Schematic

Add 5% sales tax in TEXAS plus $5.00 shipping and handling

MASTER CARD/VISA ACCEPTED

Mail to:
Texas Ranch and Shoreline Systems
P.O. Box 20
5712 Manor Rd.
Austin, TX 78723

67Ø: Otherwise, confirm action (GOSUB 3ØØØ).

68Ø: If the LOCK request is not confirmed, go back to the Mini-Menu.

69Ø,7ØØ: If confirmation is given, LOCK the file and change file name NA$(X) to include an asterisk (*). Return to the main menu.

3) CENTER PRINTING

Lines 69Ø-699 are similar in appearance to routines in each of the other sections, so it is numbered "3" and will be referred to each time the same function is performed.

69Ø: Center the text vertically.

692, 694: Center the text horizontally.

699: Print the operation being performed (in this case the operation is LOCK) and perform that action. This is where major revisions will be found (compare lines 699, 745, 86Ø, 979, and 11ØØ).

4) UNLOCK FILE (71Ø-76Ø)

This routine will UNLOCK a file only if the request is confirmed and the LOCK flag is not set.

71Ø: Check for the "U" command (UNLOCK). If the LOCK flag is set or the "U" command was not entered, go to routine 5 (RUN/BRUN).

72Ø: If so and the LOCK flag is not set, confirm the action (GOSUB 3ØØØ).

74Ø: Let us know what's going on (see routine 3), then return to main menu.

5) RUN/BRUN FILE (765-8ØØ)

765: If the "R" command (RUN/BRUN) was not entered, continue to routine 7 (LOAD/BLOAD).

77Ø: Set the DOS command to BRUN (C$ = "BRUN").

8ØØ: If the binary flag is zero, set the DOS command to RUN (C$ = "RUN") and go to routine 6 (DOS CONTROL).

6) DOS CONTROL (81Ø-89Ø)

This routine has two entry points:
A. OPTIONAL ADDRESS (line 81Ø)
B. DEFAULT ADDRESS (line 83Ø)

Line 81Ø will ask where to put the file (for BRUN or BLOAD). If the >RETURN<key is pressed, the binary program will default to its normal location. If a different location is desired, then enter that location (in hex or decimal). A hex location MUST be preceded by a dollar sign ($).

Enter at line 82Ø to bypass the optional BLOAD/BRUN address.

To enter either routine, C$ must equal the DOS command that is associated with the desired action and A$ must be equal to the file name . To enter the second routine, L$ and B$ must both be cleared.

82Ø: Control the default address for Binary files.

83Ø: Get confirmation of the action to be performed (GOSUB 3ØØØ).

84Ø: If confirmation is not given, go back to the Mini-Menu.

85Ø-88Ø: If confirmation is given, print the action (see routine 3) and call DOS to do it.

continued on next page

```
                  *-------------------------------
9CCØ- 2Ø E9 9C            JSR READ     READ IN THE NEW TRACK/SECTOR
                  *-------------------------------
9CC3- 68                  PLA          RECOVER THE CURRENT BUFFER ADDR
9CC4- 8D F1 B7            STA RW.BUF+1 BY PULLING IT OFF
9CC7- 68                  PLA          THE STACK
9CC8- 8D FØ B7            STA RW.BUF
                  *-------------------------------
9CCB- BØ 2A               BCS EX.ERR   CARRY IS SET IF AN ERROR OCCURED
9CCD- AØ ØC               LDY #$ØC     SET YREG TO POINT TO FIRST T/S
9CCF- 84 FF               STY TEMP     SAVE IT
9CD1- B8                  CLV          SET UP FOR BRANCH ALWAYS
9CD2- 5Ø AA               BVC LOAD.IT  ALWAYS TAKEN
                  *-------------------------------
9CD4- AD 6Ø AA    SKIP2   LDA F.LEN    GET NUMBER OF BYTES IN LAST SECTOR
9CD7- 18                  CLC          SET UP FOR ADDITION
9CD8- 6D F9 B7            ADC OFFSET
9CDB- 8D 3D BE            STA $BE3D    SET UP FOR PARTIAL DECODE
9CDE- 2Ø E9 9C            JSR READ     READ IN PARTIAL SECTOR
9CE1- A9 ØØ               LDA #$ØØ     RESET NIBBLIZING ROUTINE
9CE3- 8D 3D BE            STA $BE3D
9CE6- 4C EA A2            JMP CLOSE    CLOSE FILE
                  *-------------------------------
                  * READ IN A TRACK AND SECTOR, CARRY SET IF ERROR OCCURS
                  *-------------------------------
9CE9- A9 ØØ       READ    LDA #$ØØ     SET VOLUME NUMBER TO ZERO
9CEB- 8D E8 B7            STA RWTS.VOL SO WE CAN READ FROM ANY DISK.
9CEE- 2Ø E3 Ø3            JSR IOB.LOC  GET IOB LOCATION (IN DOS)
9CF1- 2Ø B5 B7            JSR RWTS     CALL DOS TO READ IN THE T/S
9CF4- BØ Ø1               BCS EX.ERR   IF NO ERROR THEN RETURN
9CF6- 6Ø                  RTS
                  *-------------------------------
9CF7- 68          EX.ERR  PLA          AN ERROR HAS OCCURED
9CF8- 68                  PLA          SO RESET THE STACK
9CF9- A2 Ø8               LDX #$Ø8     AND PRINT THE ERROR
9CFB- 4C Ø2 A7            JMP PRT.ERR  PRINT "I/O ERROR"
                  *-------------------------------

                 A471- END.LD     9C54- LOOP1     9CE9- READ      9CD4- SKIP2
SYMBOL TABLE     9CF7- EX.ERR     B7FA- MOV.NUM   B7FØ- RW.BUF    ØØ67- START
                 9C16- EXIT       9CA5- NOT.DONE  B7ED- RW.SCT    ØØFF- TEMP
AA72- B.ADDR     AA6Ø- F.LEN      B7F9- OFFSET    B7EC- RW.TRK    B5C9- TS.BUF
9C1F- BLOAD      Ø3E3- IOB.LOC    A7Ø2- PRT.ERR   B7B5- RWTS
A2EA- CLOSE      9CØ1- LOAD       ØØFB- PTR       B7EB- RWTS.VOL
B5CB- DA.BUF     9C7E- LOAD.IT    ØØFD- PTR2      9C41- SCT.LD
```

DOUBLE your DISKETTES

The only reasons your Apple][cannot use the back side of your diskette are:

1. There is no notch.
2. The diskette manufacturer did not test the back side, or worse, put the flawed front to the back.

A nibbling tool will solve problem number 1.

DISK PREP will solve problem number 2.

DISK PREP formats and tests your disk. Sectors with flaws are left so that they cannot be used. Your disk is left ready to boot, complete with a flaw report program saved on it.

$25.00

sympathetic software
9531 Telhan Drive
Huntington Beach, CA 92646

California residents add
$1.50 sales tax

Dealer inquiries invited

continued from page 71

890: Return to the main menu.

7) LOAD/BLOAD FILE (900-930)
900: Check for the "L" command (LOAD or BLOAD). If not "L", go to routine 8 (DELETE).
910: Otherwise, set the default to BLOAD.
920: If the binary flag is clear (B is not equal to 1), change to LOAD (C$ = "LOAD") and enter routine 6 (DOS Control) at the second entry point (line 820).
930: If the binary flag is set, enter routine 6 at the first entry point (line 810).

8) DELETE FILE (940-980)
940: Check for the "D" command (DELETE). If not "D", go to routine 9 (RENAME).
950: Otherwise, check the LOCK flag to see if the file is locked. If so, issue a warning (THIS FILE IS LOCKED).
960: Confirm action (GOSUB 3000).
965: If confirmation is not received, then return to the Mini-Menu.
970: If confirmation is received, center the text (see routine 3) and UNLOCK the file (in case it was locked), then DELETE it.
980: Re-start the program.

9) RENAME FILE (990-1120)
990: Check for the "C" command (RENAME). If not "C", go to routine 10 (UNLOCK/LOCK ALL).
1010: Otherwise, check the LOCK flag to see if the file is locked. If it is, print a warning message.
1030: Confirm action (GOSUB 3000).
1050: If confirmation is not received return to the Mini-Menu.
1070: Otherwise, INPUT a new file name.
1075: If the file name is greater than thirty characters print an error message and go back to the Mini-Menu.
1080: If the new name is nothing (RETURN was pressed), then return to the Mini-Menu.
1090-1100: Center the text (see routine 3) and do the required action.
1110: LOCK the file if the old file name was locked (check the LOCK flag).
1115: Redefine the old file name, NA$(X), to be the first seven characters (status) of the old file name plus the new name.
1120: Return to the main menu.

10) LOCK/UNLOCK ALL (1160-1270)
This routine will allow us to UNLOCK or LOCK ALL of the files on the disk.
1160: If "A" was not selected, go to routine 11 (EXIT?).
1170: Otherwise, get the choice (LOCK or UNLOCK).
1180: If the "L" or "U" key is not pressed, go back to the Mini-Menu.
1190, 1200: Depending upon whether "U" or "L" was pressed, set the DOS command to LOCK or UNLOCK.
1210: Center the text (see routine 3).
1220-1260: Run through a loop performing the DOS action (LOCK or UNLOCK) and modifying each file name, NA$(X), to contain an asterisk (locked) or a blank (unlocked) depending on the action taken.
1230: When a file consisting of "='s is found, exit to the main menu.

11) EXIT MINI-MENU (1280)
1280: If the "X", CTRL "X", RETURN or ESC key is pressed, go back to the main menu. Otherwise go to routine 12 (EXEC).
12) EXEC TEXT FILE (1282-1300)
1282: If the "E" command is not selected or if the Text flag isn't set (B is NOT equal to 2), then go to the Mini-Menu.
1284: Otherwise set DOS command to EXEC (C$ = 'EXEC'), and clear L$ and B$. Go to the second entry point in routine 6 (line 830).

- ERROR CONTROL -
Control of error messages is done with ONERR GOTO's. Depending upon where the ERRor occurred, these are the responses:

A. ERR IN NEW NAME. PLEASE TRY AGAIN
1310-1340: The new name of a file (change command) is illegal. LOCK the file and print message. Return to the Mini-Menu.

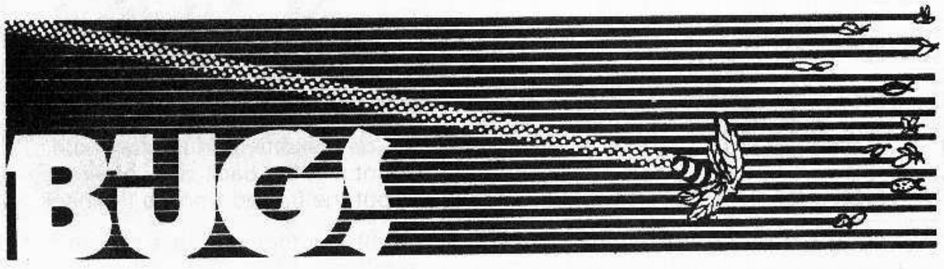
B. !!! -UNABLE TO READ DIRECTORY !!!
1350-1360: DOS was unable to read the disk, probably an I/O ERROR. Print message and exit the program.

C. ERR IN LOAD ADDRESS
1370: A bad LOADing address (used in BLOAD/BRUN command). Print the message to the screen and go to line 1340 to get a keypress. Re-enter the Mini-Menu.

D. STRANGE ERR. I WILL RE-READ THE CATALOG
1380: A strange error was encountered, so re-read the catalog. Go back to the hard entry point (line 55).

- PRINT HELP (3010-3080) -
3010-3080: Print the instructions for the help mode (invoked by pressing the ESC key when in the main menu).

- CONFIRMATION OF ACTION (3000)
3000: Asks for Y or N in order to confirm an important DOS action.



For those who have the magazine version of TEXT INVADERS, please make this change to one of the lines and it will stop crashing after you wipe out the first row of invaders.
Line 970 of TEXT INVADERS should be:

```
970 INVADER$(I) = LEFT$ (INVADER$ (I),
    RANX) + TEMP$ + MID$ (INVADER$ (A),
    RANX +2)
```

For those with DiskView 1.1, there is a bug in the "print routine". Please change line 690 to read:

```
690 PR = KY%: RETURN
```

Hardcore Computing is splitting into two magazines. See the back cover for more information...

CORRECTIONS FOR ISSUE #3

NOTE: This is a reprint of Volume 1, Issue 1 of Hardcore Computing. The following are corrections to that issue.

Page 1 - Table of Contents - VISICALC: "Job Costing" appears on page 51 instead of page 41. "HGR: Huey's Hi-Res Corner" appears on page 43 instead of page 42.

Page 24 - "Relief Map-Editor" - first column, step 3.

```
Change: 1ØØ4Ø AA = BB : BB = BMEM% + 399 : = AA TO BB STEP 2Ø : GOSUB
        11ØØØ : NEXT IN
    To: 1ØØ4Ø AA = BB : BB = BMEM% + 399 : TEST = MAX : GAP = -1 FOR IN =
        AA TO BB STEP 2Ø : GOSUB 11ØØØ : NEXT IN.
```

Page 33 - "Menu" - The machine language listing in the first column is correct, but to make line Ø identical to the sample line Ø change the two FF's in line 828 to B2 84 by typing:

```
82C:B2 84
```

When setting the "end of program" counter, make sure you type a space between the 3A and Ø8.

Page 4Ø - "Zephyr Wars" - Insert line 2ØØ:

```
2ØØ ROT=Ø:X% = XUFO%(Z) * 7:Y% = YUFO%(Z): IF Y% = Ø THEN RETURN
```

Page 46 - "Vector Graphics" - Line 21Ø should read:

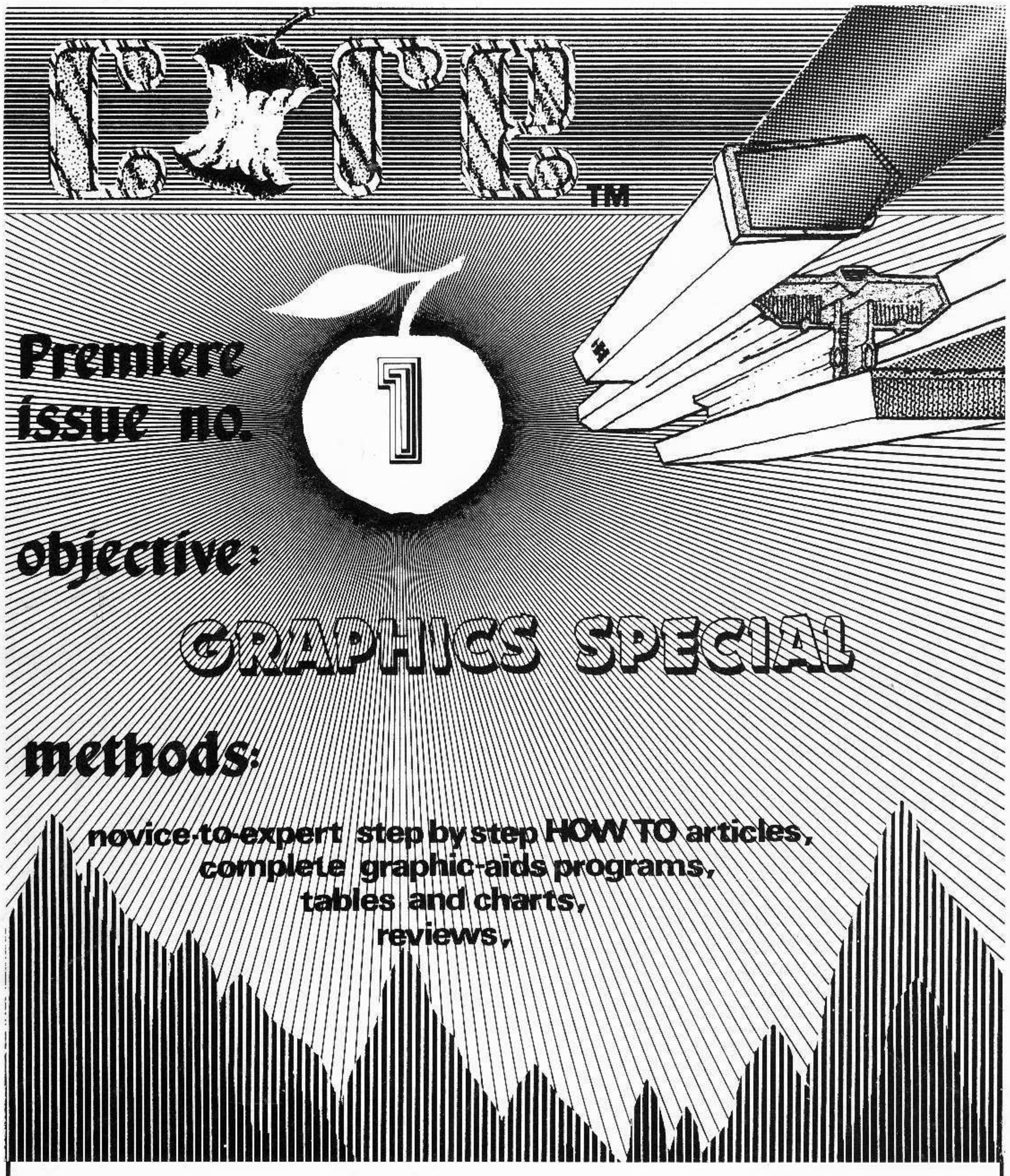
```
21Ø IF A<Ø THEN 25Ø
```

Page 72 - "Bugs" - This section should read: For those of you who have the magazine version of TEXT INVADERS, make this change to line 97Ø and it will stop crashing after you wipe out the first row of invaders:

```
97Ø INVADERS$(1) = LEFT$ (INVADERS$(1), RAN%) + TEMPS$ + MID$
    (INVADERS$ (AA), RAN% +2)
```

Also, for those with DiskView 1.1, there is a bug in line 69Ø of the "print routine." Please change line 69Ø to read:

```
69Ø PT = KY%: RETURN
```



Premiere issue no. 1

objective:

GRAPHICS SPECIAL

methods:

novice-to-expert step by step HOW TO articles,
complete graphic-aids programs,
tables and charts,
reviews,

Hardcore Computing has become two magazines: CORE and HARDCORE COMPUTIST. CORE will be a quarterly magazine, and each issue will focus on a different topic. HARDCORE COMPUTIST will be published the other eight months, and will be devoted to articles about unlocking copy-protected discs and other advanced subjects. Four issues of CORE and eight issues of HARDCORE COMPUTIST will be included in a single $2Ø subscription. Use the coupon on page 3 to subscribe to both magazines.