ISSUE 66 1989

For The Serious User Of Personal Computers

COMPUTIST

PO Box 110846-T Tacoma, WA 98411

Canada & Mexico $7.00 **U.S. $3.75**

Table of Contents

Editorial **3**

Most Wanted Softkeys... **37**

Features:

Displaying Hi-Res and Double Hi-Res pictures from BASIC **11**
Two slide-show programs that work under DOS or ProDOS.

The Product Monitor **13**

A Treatise on Deprotection **14**
An indepth look at how to go about deprotecting software.

Laser 128 - The Dream Machine **21**
Looking at the pluses of owning a Laser 128.

Super COPYA 1.1 .... **22**
Automate all those hard to remember patches.

An even BETTER Bootable Thexder **23**
Run Thexder under GS/OS.

APT Scanner **24**
A disk search utility to help you make A.P.T.'s.

ProDOS EOR Disk Scanner **30**

New Routines for Super IOB **35**

Softkeys:

4th & Inches (IIgs) 6
ACT Preparation 12
Addition Logician 8
Adventure Double Feature Vol II 9
Algebra 1 11
Algebra 2 11
Algebra Disks 1-6 12
Alien Rain 21
Binomial Multiplication Equations 12
Broadsides v2.0 23
Calender Crafter (IIgs) 5
Calendar Crafter v1.1 29
California Games (IIgs) 5
Chessmaster 2100 v1.01 29
Computer Drill and Instruction: Mathematics 'Addition A' 26
Crossword Magic 4.0 29
Decimals Disk 1 & 2 12
Demon Derby 21
Discovery Lab 8
Dive Bomber 26,30
Elementary Volume 1 8
Equations II 12
Factoring Algebraic Expressions 12
Files on the Apple 8
F.M.C. Program 8
Fractions 26
Friendly Computer 8
Game Show (The) 7
Geometry Disk 1-5 12
Gradebook III 14
Graphing Linear Functions 12
Intermediate Algebra 26
Into the Eagle's Nest 23
Introductory Algebra 26
Kid Niki - Radical Ninja 26
Label Utility 20
Last Ninja GS (The) 6,19
LOGO Robot 9
Magic Spells 35
Mastery Arithemetic Games 11
Math Shop 19
MECC 1988-89 Copy System 20
MECC Copy Program 7
Microzine #25 25
Microzine #27 12
Microzine #28 12
Microzine #29 12
Microzine Jr #3 Disk 1 & 2 12
Millikens Pre-writing Series Branching-Brainstorming-Nutshelling 19
Mindshadow 8
Mini Putt 12
Monkey Business 8
Moptown 35
Mystery Double Feature Vol 1 9
Paint with Words & Word Art Show 26
Poster 9
Reading Skills 2 7
Right of Way 8
Sargon III 11
School Magic 8
Science Volume 1 8
Science Volume 2 8
Science Volume 3 8
Science Volume 4 8
Scrabble 31
Scruples 34
Serve & Volley (IIgs) 5
Simultaneous Linear Equations 12
Snoggle 20
Space Subtraction 8
Spanish for Mastery Software 7
Special Needs Volume 2 8
Stickybear Alphabet (IIgs) 6
Subtraction Puzzles 8
Times of Lore 25
Word Herd: Look-Alikes 8
Word Herd: Sound-Alikes 8
Word Wizards 8
Zoyon Patrol 9

A.P.T.s:

Buck Rogers 25
Kid Niki - Radical Ninja 25
Lode Runner 28
Marble Madness 24
Montezuma's Revenge 25
Pirates! 10
Roadwar 2000 22
Saracen 25

Playing Tips:

Maniac Mansion 23
Marble Madness 24
Wasteland 20
Zany Golf 27

Notes:

Viruses (just say No) 7
Faulty joystick problems? 9
Publish It! fonts 9
Softswitch and the BBR 10
Fiber Optic LAN 10
LISP Compiler (Help!) 10
IIgs Plus rumors 27

Bugs:

The Bard's Dressing Room III 7

IBM Softkeys:

Managing Your Money 38
Print Shop 38
Reader Rabbit 38
Xenocopy Plus v1.09 38

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

COMPUTIST
PO Box 110846-T
Tacoma, WA 98411

Readers Data EXchange

New COMPUTIST readers using Apple IIs are advised to read this page carefully to avoid frustration when attempting to follow a softkey or entering the programs printed in this issue.

What is a softkey, anyway?

Softkey is a term which we coined to describe a procedure that removes, or at least circumvents, any copy-protection on a particular disk. Once a softkey procedure has been performed, the resulting backup copy can usually be copied by the normal copy programs (for example: COPYA, on the DOS 3.3 System Master disk).

Commands and control keys

Commands which a reader is required to perform are set apart by being in boldface and on a separate line. The **RETURN** key must be pressed at the end of every such command unless otherwise specified. Control characters are preceeded by a small control key. An example of both is:

6 ⊖P

Press **6**. Next, place one finger on the ⊖ key and then press **P**. Don't forget to press the **RETURN** key.

Other special combination keypresses include **⊖RESET** or ɗ ⊖ **RESET**. In the former, press and hold down the ⊖ key then press the **RESET** key. In the latter, press and hold down both ⊖ and ɗ then press **RESET**.

Software recommendations

The Starter Kit contains most of the programs that you need to "Get started". In addition, we recommend that you aquire the following:

- Applesoft program editor such as "Global Program Line Editor (GPLE)".
- Assembler such as the "S-C Assembler" from S-C software or "Merlin/Big Mac".
- Bit-copy program such as "Copy II Plus", "Locksmith" or "Essential Data Duplicator".
- Word-processor (such as AppleWorks).
- "COPYA", "FID" and "MUFFIN" from the DOS 3.3 System Master disk are also useful.

Super IOB and Controllers

This powerful deprotection utility (in the COMPUTIST Starter Kit) and its various Controllers are used in many softkeys. (It is also on each **Super IOB Collection** disk.)

Reset into the Monitor

Softkeys occasionally require the user to stop the execution of a copy-protected program and directly enter the Apple's system monitor. Check the following list to see what hardware you will need to obtain this ability.

Apple II+, //e, compatibles: 1) Place an Integer BASIC ROM card in one of the Apple slots. 2) Use a non-maskable interrupt (NMI) card such as *Replay* or *Wildcard*.

Apple II+, compatibles: 1) Install an F8 ROM with a modified reset-vector on the computer's motherboard as detailed in the "Modified ROM's" article (COMPUTIST #6 or Book Of Softkeys III) or the "Dual ROM's" article (COMPUTIST #19).

Apple //e, //c: Install a modified CD ROM on the computer's motherboard. Cutting Edge Ent. (Box 43234 Ren Cen Station-HC; Detroit, MI 48243) sells a hardware device that will give you this important ability but it will void an Apple //c warranty.

Apple //gs: If you have the 2.x ROM, there is a hidden classic desk accessory (CDA) that allows you to enter the monitor. In order to install the new CDA, you should enter the monitor before running any protected programs (CALL -151) and press "#" **RETURN**. This will turn on two hidden CDAs, *Memory Peeker* and *Visit Monitor*. Thereafter press ɗ ⊖ **ESC** to go to the Desk Accessories menu. Select "Visit Monitor" and there you are. Use ⊖ **Y** to exit.

Recommended literature

- *Apple II Reference Manual (or IIe, IIc, etc.)*
- *DOS 3.3 or ProDOS manual*
- *Beneath Apple DOS & Beneath Apple ProDOS, by Don Worth and Pieter Lechner, from Quality Software*

Typing Applesoft programs

BASIC programs are printed in a format that is designed to minimize errors for readers who key in these programs. If you type:

```
10HOME:REMCLEAR SCREEN
```

The LIST will look like:

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

...because Applesoft inserts spaces into a program listing before and after every command word or mathematical operator. These spaces don't pose a problem except when they are inside of quotes or after a DATA command. There are two types of spaces: those that have to be keyed and those that don't. Spaces that must be typed appear in COMPUTIST as delta characters (^). All other spaces are there for easier reading.

NOTE: If you want your checksums to match, only type spaces within quotes or after DATA statements if they are shown as delta (^) charactors. **SAVE** the program at periodic intervals using the name given in the article.

Typing Hexdumps

Machine language programs are printed in COMPUTIST as hexdumps, sometimes also as source code. Hexdumps are the shortest and easiest format to type in. You must first enter the monitor:

CALL -151

Key in the hexdump exactly as it appears in the magazine, ignoring the four-digit checksum ($ and four digits) at the end of each line. When finished, return to BASIC with:

3D0G

BSAVE the program with the filename, address and length parameters given in the article.

Typing Source Code

The source code is printed to help explain a program's operation. To enter it, you need an "Assembler". Most of the source code is in *S-C Assembler* format. If you use a different assembler, you will have to translate portions of the source code into something your assembler will understand.

Computing checksums

Checksums are 4-digit hexadecimal numbers which tell if you typed a program correctly and help you locate any errors. There are two types of checksums: one created by the *CHECKBIN* program (for machine language programs) and the other created by the *CHECKSOFT* program (for BASIC programs). Both are on the "Starter Kit".

If your checksums do not match the published checksums then the line where the first checksum differs is incorrect.

CHECKSOFT instructions: Install Checksoft (BRUN CHECKSOFT) then LOAD your program. Press **&** to get the checksums. Correct the program line where the checksums differ.

CHECKBIN instructions: Enter the monitor (CALL -151), install Checkbin at some out of the way place (BRUN CHECKBIN, A$6000), and then LOAD your program. Get the checksums by typing the Starting address, a period and the Ending address of the file followed by a ⊖ **Y**.

SSSS.EEEE ⊖ Y

Correct the lines where the checksums differ.

Writing to the RDEX editor

- RDEX stands for: Reader's Data EXchange. We print what you write. When you send in articles, softkeys, APTs, etc., you are submitting them for FREE publication in this magazine. **RDEX does NOT purchase submissions nor do we verify data submitted by readers.** If you discover any errors, please let us know.
- Remember that your letters or parts of them may be used in RDEX even if not addressed to the RDEX editor. Correspondence that gets published may be edited for clarity, grammar and space requirements.
- Because of the great number of letters we receive and the ephemeral and unpredictable appearance of our part-time staff, any response to your queries will appear only in RDEX, so it would be more appropriate for you to present technical questions to the readers and ask for their responses which will then be placed in the Apple-RDEX.
- Whenever possible, **send everything on 5¼" disk.** Use whatever text editor you like, but tell us which one. Put a label on the disk with your name (or pseudonym) and address (if you want to receive mail). Don't reformat any programs. Send Applesoft programs as normal Applesoft files and machine language programs as normal binary files. We have programs to convert them to the proper format for printing. If you are sending source code files, send them as normal text files. We will return your disks, whenever possible, with the current library disk copied onto it.
- Don't send hardcopy (printout) unless it is about a bug or other printing error. *(If you are writing about your subscription or sending an order, do send your letter on paper. Karen keeps the hardcopy and forwards the disks.)*
- If you use a pen name and want to receive mail, we need to have your address. Our readers privacy is important, so we will not print your address unless you specifically say too.
- When writing to request help, be sure to include ALL relevent information. The more information you include, the easier it is to find a solution.
- When writing to one of the RDEX authors. Write your letter and seal it in an envelope. Put your return address, the authors name (as it appears in RDEX) and the correct postage on the envelope. Put this envelope into another and send it to RDEX. We will put the correct address on your letter and mail it.

COMPUTIST

Editor: Charles R. Haight
Circulation: Karen Fitzpatrick
Advertising, call: (206) 474-5750
Publisher: SoftKey Publishing

- Address all advertising inquiries to: COMPUTIST; Advertising Department; PO Box 110816; Tacoma, WA 98411
- Mail all RDEX letters to:

COMPUTIST
Apple-RDEX or IBM-RDEX
PO Box 110846-K
Tacoma, WA 98411

- COMPUTIST does NOT purchase editorial material. The entire editorial content consists of information submitted to COMPUTIST for publication in the shared interests of all COMPUTISTs.
- Unsolicited material (manuscripts, letters to the editor, softkeys, A.P.T.s, playing tips, questions, etc.) are assumed to be submitted as letters-to-the-RDEX-editor for publication with all and exclusive rights belonging to COMPUTIST.

SUBSCRIPTIONS: Rates (for 8 issues):

U.S.—$24	U.S. 1st Class—$34
Canada/Mexico.—$34	Other Foreign—$54

- Send subscription inquiries to: COMPUTIST; Subscription Department; PO Box 110846-T; Tacoma, WA 98411
- Domestic Dealer rates: Call (206) 474-5750 for more information.
- **Change Of Address:** Please allow 4 weeks for change of address to take effect. On postal form 3576 supply your new address and your most recent address label.
- Issues missed due to non-receipt of change of address may be acquired at the regular back issue rate.
- We are not responsible for missing issues 90 days after mailing date. If you do not receive an issue at the usual time each month, please call or write.

You have a LEGAL RIGHT to an unlocked backup copy of your commercial software

Our editorial policy is that we do NOT condone software piracy, but we do believe that users are entitled to backup commercial disks they have purchased.

In addition to the security of a backup disk, the removal of copy-protection gives the user the option of modifying programs to meet his or her needs.

Furthermore, the copyright laws guarantee your right to such a DEPROTECTED backup copy:

..."It is not an infringement for the owner of a copy of a computer program to make or authorize the making of another copy or adaptation of that computer program provided:

1) that such a new copy or adaptation is created as an essential step in the utilization of the computer program in conjunction with a machine and that it is used in no other manner, or

2) that such new copy or adaptation is for archival purposes only and that all archival copies are destroyed in the event that continued possession of the computer program should cease to be rightful.

Any exact copies prepared in accordance with the provisions of this section may be leased, sold, or otherwise transferred, along with the copy from which such copies were prepared, only as part of the lease, sale, or other transfer of all rights in the program. Adaptations so prepared may be transferred only with the authorization of the copyright owner."

United States Code title 17, §117

Charles R. Haight

I've been banging my head against a wall for a long time, trying to keep COMPUTIST going. Some of you have been with us for an equally long time and are already aware of what's going on. Some of you are new to COMPUTIST and don't know what I'm talking about. Some of you don't believe me.

To those few who don't believe, I say this; Call any large printer in your area, ask them for a quote on a 5000 copy, 8½" by 11", 48 page, self-cover, on 40 lb book stock, with one color besides black on one signature, saddle stitched and delivered. Then call your nearest USPS bulk mailing center and ask how much a bulk mailing permit is and what the current cost of mailing 4000 copies of a 4 oz magazine is. Then call a local typesetter and ask what the cost of typesetting 48 pages of text (point size 9, line space 10) plus various sized heads using a 2600 lines/inch phototypesetter will cost. After that, you will have some idea of what we are up against for every issue we send out to you. And that doesn't include rent, telephone, water, electric, insurance, equipment maintenance, daily postage, UPS service, bulk mailing services, foreign mail fowarding services, state and local business operating taxes, business property (computers) taxes, etc, etc, etc... Sometimes the list seems endless.

The volunteers who come here hate the donkey work but love the magazine. If it weren't for the support of the volunteers and many of our hardcore readers, I would have given up long ago.

Whew! Thanks for letting me get that out. Now, let's put that behind us, get out and kick some rears, and put this rag back on track.

I've got a lot of information for you. Please take the time to read this, it's important. We are at a crossroads here and the next few months are going to see some dramatic changes, for better or for worse. Read the rest of this editorial and then, sit down and write a long letter to me. Tell me what you like, tell me what you hate, praise me or curse me, but sit down and write. It's important that you let me know what **you** think.

Why we're in trouble

If you divide COMPUTIST's average monthly operating costs by the number of subscribers and multiply by 12 issues, the cost of a one year subscription should be around $48. As printing and other costs went up, the cost of a subscription should have followed. But we got so much negative feedback and outright cancellations when we first went to $32/year that we have been reluctant to ask for another increase.

For the longest time, the magazine has been subsidized by back issue and disk sales. (That's why those prices haven't come down.) The average new subscriber spends about $60 dollars on back issues and library disks. Part of that is added to the $32 subscription fee to get $48. The rest goes to supplement ordinary renewals from our regular readers. This has worked for some time, but it is a false security and a five month long drop in back issue sales pointed out how fragile our house really was. So we are looking at all possible options very closely. I think you should all take a close look at the following options and perhaps someone out there has some other ideas that could help.

Option 1 - raise the rates

Change the subscription rates to $48. Ugh! Yuck! Forget it! Aside from a few of us, who could afford this, most of us would be hard pressed to come up with the extra bucks. We would, reluctantly, let our subscription lapse. Besides, I don't want to answer the nasty letters, full of rude suggestions, that I would surely receive. I've already gone thru that once.

Option 2 - Increase our (paid) readership

We figure that an increase to 8000 readers will allow us to meet all bills using the subscription money alone. More readers means a larger press run and that will save money. Printers dislike small press runs. They have to charge more and can't make any money. The reason is the setup charges. Here's a rough idea of how it goes.

1. Prepare the paste-up pages so negatives can be shot. Also strip in the color overlays.
2. Shoot the negatives and check them for errors, spots etc.
3. Make the printing plates from the negatives.
4. Mount the plates on the presses.
5. Load the paper rolls and ink. (5 stations are used for a tabloid, 3 stations for the magazine.)
6. Start the press and adjust the alignment so the pages are correctly positioned when the sheet is folded. Also clean any smudges and adjust the ink feed.

Note: While this is going on, the press is running at reduced speed but it is still wasteing paper and ink. Everything that comes out of the press is thrown away as scrap. The initial setup plus running alignment takes about 2 hours.

7. Everything is correct, start counting the copies. At this point the press can be brought up to speed. 4000 copies takes about 12 minutes.
8. Shut down the presses.
9. Unload and store the 5 rolls of paper. A single roll can print 3 issues so these 5 rolls will be used again for the next 2 issues.
10. Clean the presses so they will be ready for the next job. Cleaning takes a long time, also.

As you can see, only a small amount of the total time is spent actually printing our 4000 copies. So, doubling the number of copies would not double the cost. Only the paper and ink charges would increase. The setup charges would be the same.

Increasing our readership is obviously the best solution to our problem.

Option 2a - The Readers Wish List

One idea, suggested by a reader, is to give software prizes for finding new subscribers.

We would compile a list of software, using the wish lists that you send us. (You are sending your list, aren't you?) Each piece of software would have a point value, based on its wholesale cost. (IE. A game might retail for $49 but actually cost a dealer only $32. It's point value would be 32.)

For every new (paid) subscription that you bring in, you would accumulate 5 points. These points would then be used to get free software from the list. No, renewals don't count, remember, we're trying to find **new** readers.

We will send any subscriber, who is interested, one copy of our new info flyer, with your name and record number printed on it, next to the subscription form. You would make copies of the flyer and give them to other Apple owners. The information flyer tells about the magazine so all you really have to do is take a little time and pass them out. Are there any computer shows, computer swap meets, Apple clubs, etc., in your area? Well, here's your chance to help us and yourself.

Option 3 - Lower the rates

Are you kidding? Well, maybe. It was suggested that lowering the rates might inspire more subscriptions, which could, in the long run, make those rates feasible. (Has anyone calculated how long?)

One way to achieve lower rates is to print less often. As most of you have heard by now, we are "suggesting" a move to a 8 time per year publishing schedule. The primary reason for this is to lower the cost of a subscription. Other than a 3 fold increase in the number of subscribers, nothing else would allow us to lower the rates.

It seems pretty silly to reduce our publishing schedule when we are receiving more and more material from you, our readers. We should be increasing the number of pages/issue not reducing the number of issues. I need those extra pages, but many of you are annoyed enough at the $32 per year cost and would not be friendly towards an increase.

Going to 8 issues would allow us to reduce the subscription rate to $24/yr. Here's the current page breakdown for a normal 48 page issue:

1 Front Cover
2 Inside Front - Masthead
3 Subscription ad
4 Authors list and Editorial page
5 Table of Contents
6 Data Page
7-37 Apple RDEX (31 pages)
38-42 Back issue list (5 pages)
43 Shopper Ad
44-45 IBM RDEX (2 pages)
46 unClassified ads
47 Inside back - outside ads
48 Back cover - house ads

Counting the Authors list and Table of Contents, 33 out of 48 pages are used for the Apple RDEX. There are some things I can do to increase this number.

- We currently use a point size of 9 with a line space of 10 for our body text. Other magazines use point size 10 with a line space of 11. Our slightly smaller print already packs more information per page than the usual magazine (about 1 extra page for every 8 pages of text). But I can use the "Set Size" function on the typesetter to make the print just a hair thinner. This saves about 1/9th of a page or, you could say, adds 1 page for every 9 pages of text.
- The RDEX logo is cute but it eats up 1/2 inch at the top of each page. Recovering that space gives us another page for every 20 pages.
- We can use 3 columns for the BASIC program checksums, instead of 2, and go one point size smaller on the BASIC program listings.
- We can axe the back issue list and print it only once or twice a year. This would free more 5 pages.
- I can also move some pages around to allow "tear-outs". I know, most of you would hang anyone who tore a page out of one of your issues. But some of you do tear pages and, for them, this would make it a little less painful.

Here's what we end up with.

1 Front cover
2 Inside front - Masthead and RDEX info
3 Data page or ads Tear-out
4 Subscription ad Tear-out
5 Authors list and Editorial
6 Contents
7-42 Apple RDEX (36 pages)
43-44 IBM RDEX (2 pages)
45 Shopper ad Tear-out
46 unClassified ads Tear-out
47 Inside back - other ads
48 Back cover - other ads

You'll notice, the tear-out pages are all ads of one sort or another and match each other in the front and back of the issue.

Counting the Authors list and Table of Contents, there would be 38 pages for Apple RDEX. But wait, let's not forget the extra page we get from deleteing the RDEX logo and the 1 in 9 boost with type width compression. 9 goes into 38 about 4 times. This means 5 more pages or an effective total of 43 pages of text, compared to previous issues. Now let's see... The new page count, (43), times 8 issues/year is 344. Divide this by the old page count (33) and you get 10 plus a little. With the new subscription rates, you pay for 8 issues but get 10 issues worth of information. That makes me feel a little better, but I'll keep working on it to see if there isn't something else we can do.

Option 3a

We could use the 8 issue subscription to lower the renewal rates but continue to publish each month. This means that you would get your renewal notices every 8 months instead of every 12. The "all at once" out of pocket expense would be lower ($24) while the real subscription cost remained the same.

Option 4 - Tabloid & book

This issue is printed in the tabloid format. For less cost than the current 48 page issue, we can print a 40 page tabloid. The paper used is a high quality newsprint called "Electrabrite". It is smoother and whiter than newsprint but, since it is newsprint, it is less expensive than the book paper we use now and can be printed on a less expensive press. The text area of this tabloid is 10" by 13¼". Our regular page is about 7½" by 10". That makes a tabloid page almost twice the size of a regular page. A 40 page tabloid is like getting an 80 page issue. You would receive double the amount of information and I would have more room for all the material that you are sending now.

At the end of each year, we were thinking of compiling all the data and printing a book. A big book. Only those who needed a long time archive copy would need to buy the book.

Another option would be to print the "book" as loose

8½" by 11" sheets with 3-holes for a 3-ring binder.

I'd like to cast my vote for the tabloid format. I have a lot of information ready to be printed and more coming in every day. I really need those pages. Remember, our greatest strength is our ability to print your questions and letters quickly. But letters are coming in faster than I can print them using our regular format. Some material in my in file has been waiting almost 2 months to be printed. The softkey you need may be in there. Give me those extra pages and I'll double the information you get each issue and bring down the time to get letters printed too. I may be greedy, but I want those extra pages.

One other thing to consider. With the tabloid format, the per page cost is less so the advertising cost is less. Maybe we can pick up some new advertising with lower rates. Anything to help pay for the issues.

Option 5 - Frequent issues

I know it sounds crazy but I certainly have enough material to warrant it, and this is what everyone I talked with wants. How can we save by printing more often? Well, it's a trade-off. The 3 biggest bills are for printing, mailing and typesetting. Printing an issue every 2 weeks means that you would have to renew sooner. (Twice a year, double the printing would double the 1 year sub rate.) The subscription rates would pay (?) for the increased printing and mailing cost. Any savings would come from typesetting. The typesetting equipment is on a lease. We pay the same amount each month whether we use it or not. Using it more often would only increase the cost of film and chemicals. We calculated the lease, maintenance and materials for one year, divided by 12, and came up with $2300 per month. Only half of that would apply to each issue. This reduces the per issue cost.

It sounds like you can have your cake and eat it too but there are some other considerations.

1. Everyone would have to send all their letters to RDEX, on disk, even the short letters. Two weeks leaves very little time to edit and no time at all to type in letters. Don't forget, your disk is returned with the current library disk copied onto it. So you don't lose anything. And you can use any text editor or word processor that you own and send it on a DOS or ProDOS disk. (Use only a 5¼" disk at this time.).

2. We would have to start slow and gradually reduce the time between issues. That way, we would have time to work out any bugs. Also, the Tacoma area subscribers should give some serious thought to stopping by and helping out.

3. If you decide you like the idea of more frequent issues, you should also consider the tabloid idea. Frequent printing and the tabloid would go together well. (You like ice cream with your cake?)

The Nit'is Grit'is

The bottom line is still the same, we need to double the subscription rates or double the number of subscribers. The latter solution is the best. We, all of us, need to do our part to find more subscribers. If you have any ideas, please write and tell me. There's almost 4000 of you out there, and if we all think about it, we're bound to come up with some good ideas.

The free software giveaway (detailed in the flyer) is one step in that direction. We hope it will get people talking about us, and maybe they'll take a look at COMPUTIST too. If it works, then we can take more $money$ away from what we spend on commercial ads and use it to increase the amount of software given away in each issue, but only if it works.

A healthy magazine needs a strong subscriber base and like the saying goes, "There is strength in numbers".

Best BASIC program

While we're on the subject of free software, I've got an idea for another way to win. Many of our subscribers would like to see more BASIC program listings. I propose that we add another piece of software to the 4 pieces that are going to be given away each issue. This one would be for the best BASIC program in each issue (not counting the Super IOB controllers). Let me know how you feel about this, or how you feel about the whole idea of the free software giveaway. Is it a good idea or a bad one? Should there be catagories or should it just be the 3 or 4 best articles, period? How should the categories be set up? (Best BASIC program, best softkey, best machine language routine, most informative, etc.) If we do this right, we can use the free software program as incentive to improve the quality of the material in the issues. I need your feedback on this. Sit down and write me a letter, right now!

Help lines

Jack Nissel called the other day and suggested that we have telephone "help" lines. I think it's a great idea. Would any of you be interested in receiving calls from other subscribers needing help? If you are, write and let me know your number, the times (and your time zone) that you would be available and the subject(s) you feel you can help out with. Sometimes, all someone needs to get out of trouble is a few words from someone who's been there before.

Callers to these help lines should insure that they have all the details of their problem/questions ready before they call and, most of all, they should **respect** the times given for that help line. Don't forget the time zones.

The Beginners Notes

We need some volunteers to help compile a beginners notebook for new subscribers. This notebook should contain the best of the tutorial articles from past issues. This job would involve a lot of editing and rewriting. One article for each of the major protection schemes needs to be selected and updated. Also, a complete glossary of terms used in Computist and general topics such as DOS and ProDOS workings, Applesoft BASIC, machine code, Apple internals, etc. should be covered. (A complete set of issues would be very helpful.) We can load the text for any past article onto a disk for editing. If anyone is interested in working on this, please contact the RDEX editor. Any help would be greatly appreciated.

Bounties for the Most Wanted

A reader has suggested that instead of dropping program names from the most wanted list, we offer a bounty for stubborn programs. I think he has a good idea, if it's done right.

We will still drop programs from the most wanted list after 6 issues. This cleans out the deadwood and forces you to keep the list current by sending new wants for the list. But if you think a program has been on the list too long, then write us and nominate it for a bounty. If at least 1% of the subscribers nominate a program, then we will put a bounty on it. A certain number of points based on how many readers voted for it. (Remember, points are good for free software from the wish list.)

We won't tell you which programs have a bounty. Instead, we will tell the number of bounties in each Most Wanted list. That way readers won't be tempted to sit on a softkey and wait for a bounty.

What do you think? Is that a good idea? Should we do it?

Wish list

We are compiling a list of software for our free giveaway. Send us your wish list of software (hardware?). Be honest and don't get crazy. No requests for MAC II's or Cray computers please. Tell us what software you would really like to have but can't afford or just haven't got around to buying yet. You just might be the first winner in our giveaway.

Club news

The first response to the club flyer came by telephone on Friday (April 14). Bob Kesslick is the first official member of the Hardcore Computist Club. Thanks Bob. It's nice to know who our hardcore supporters are. As of today (May 9) there are 700 members in the club. Over 20% of those members sent more than $10. That's the good news.

The bad news is that we need a 63% response to get out of the hole we're in. Where are the rest of you? I remember an old saying "When you're -ss deep in alligators, you don't have time to worry about how deep the water is". Well, we've been beating off the alligators with a short stick. We need your help to get Computist out of the water and back on solid ground.

If you haven't joined the Computist Club, do so today. And if you know anyone who may be interested in our newsletter, talk to them today and convince them to subscribe.

Send us a postcard or call us at (206) 474-5750 and request a copy of the Computist Info Flyer. Remember Option 2a, you can get free software for finding new subscribers. If we all pull together, we can make this the best user supported magazine, anywhere.

RDEX Contributors

The	A.S.P.	38
Douglas	Bancroft	24
Matthew D.	Bancroft	24
Mike	Basford	38
B. Dudley	Brett	35
I.B.	Darn	20
Rick	Davis	20
Bob	Colbert	21
Richard	Cole	38
A.	Evans	10
Stephen A.	Garbaty	28
C.E. "Chuck"	Garrett	28
Phil	Goetz	25
Bette B.	Goode	19
Jim S.	Hart	25
A.L.	Head Jr	14
Jim	Heil	7
James A.	Hodge	29
Jeff	Hurlburt	13
Bob	Igo	24
Bill	Jetzer	11
Kay	Jun	12
Gary	Kowalski	21
Rich	Linville	9
Mike	Maginnis	19,25
Brent	Michalski	7
Jose A.	Montano	7
Bud	Myers	9
Jack R.	Nissel	7
Ann	Onymous	38
Tim	Rafert	23
Michael	Reese	14
Gary	Rohr	23
Irwin	Roth	38
Vince	Ruggiano	19
Scott M.	Simon	22
Edward	Teach	7
Brian A.	Troha	5
	Unk	38
Gary	Verbuch	10
Chris	Willis	22
	Zorro	23

Brian A. Troha

IIgs Softkey for...

Calender Crafter
MECC

■ Requirements

☐ Apple IIgs 512K
☐ 3½" disk copier
☐ 3½" disk editor

Calender Crafter (CC) from MECC is a great program for making personalized calenders. You can even add GS SHR (Super Hi-Res) pictures if you have 768K in your GS. The type of protection used is a bad block check on block $7. I scanned the disk for 22 A8 00 E1 22 and found it on block $604. After much checking I found the routine that makes it's calls to the block read routine. It is on block $603 and looks like this (on the disk):

```
10C:A9 00 00      LDA #0000       Load pass value
10F:8F F3 66 01   STA 0166F3      Store in the pass/fail flag
113:20 B5 08      JSR 08B5        Go check for the original
116:A2 04 CA      LDX #CA04
119:22 00 00 E1   JSL E10000
11D:6B            RTL             Return to caller
11E:22 A8 00 E1   JSL E100A8      ProDOS 16 interface (MLI)
122:21 00             0021        Call number (GET LAST DEVICE)
124:DD 09 00          0009DD      Parm table location in memory
127:00                00          End of MLI parm table
128:B0 46         BCS 170 (+46)   Had an error, goto hang
12A:20 4A 09      JSR 094A        Read the bad block $7
12D:90 05         BCC 134 (+05)   Carry clear = no bad block
12F:C9 27 00      CMP #0027       0027=I/O error from disk interface
132:F0 3B         BEQ 16F (+3B)   bad block, so it's an original
134:A9 01 00      LDA #0001       Start with first device on line
137:8D DD 09      STA 09DD        Store in parm table
13A:20 4A 09      JSR 094A        Check for a bad block $7
13D:B0 30         BCS 16F (+30)   If an error, then continue
13F:EE DD 09      INC 09DD        Try next device
142:C9 11 00      CMP #0011       Any devices left?
145:D0 F3         BNE 13A (-0D)   If yes, goto 13A
147:A2 04 CA      LDA #CA04
14A:22 00 00 E1   JSL E10000
14E:48            PHA
14F:F4 00 00      PEA 0000        Out of devices
152:F4 0A 09      PEA 090A        Set up for asking user
155:F4 00 00      PEA 0000        to 'Insert the Calender
158:F4 00 00      PEA 0000        Crafter master disk...'
15D:A2 15 17      LDX #1715
15E:22 00 00 E1   JSL E10000      Print it
162:68            PLA
163:C9 01 00      CMP #0001       0001 = continue
166:F0 CC         BEQ 134 (-34)   So try again
168:A9 FF FF      LDA #FFFF       forget it, let's quit
16B:8F F3 66 01   STA 0166F3      Mess up pass/fail flag
16F:60            RTS             Return to the sender
170:4C F6 0D      JMP 0DF6        Print some stuff & hang
```

After some checking around I found the JSR 08B5 is the call to the bad block check. However, I thought I would check the disk for all references to F3 66. I found two checks of $66F3 (for a zero) and one place where it incremented $66F3. I found that calls to the beginning of the disk check look like JSL 000A95 (22 95 0A 00). So I scanned the disk for that sequence and found four calls to it. After finding so many calls to the protection, I decided to just cancel the JSR 08B5 (a single byte edit) instead of going through and changing all references to $66F3 and 0A95 (almost 30 bytes). Well to wrap this one up, it works! If you don't find the 20 B5 08 on block $603, then search the disk for 8F F3 66 01 20 B5 08 and change the 20 to AD.

1. Make a copy of the disk (ignore any read errors).

2. Edit the copy.

Block	Byte(s)	From	To
$603	$114	20	AD

3. Write the block back to the copy.

Optional: upload the program to your Hard Disk.

IIgs Softkey for...

California Games
Epyx

■ Requirements

☐ Apple IIgs 512K
☐ 3½" disk copier
☐ 3½" disk editor

California Games has been released in a full GS format, with great GS SHR (Super High Resolution) screens. Like most GS programs from Epyx, they are using the old 3½" disk nibble counting routine. This routine is more like the one used on Street Sports Soccer, rather than Destroyer. On the boot, ProDOS 16 runs CAL.SYS16 which sets the prefix to volume/OBJECT, then runs the program CALIF.GAMES. It's the file CALIF.GAMES that contains the protection and a single call to that routine. The protection routine looks like this on block $FC:

```
40:20 C5 52       JSR 52C5
43:08             PHP
44:18             CLC
45:FB             XCE             Enter full GS mode
46:08             PHP
47:C2 30          REP #30         16-bit wide Accum.
49:AD C9 52       LDA 52C9
4C:48             PHA
4D:22 D2 52 00    JSL 0052D2
51:A8             TAY
52:68             PLA
53:90 03          BCC 58 (+03)
55:4C 93 52       JMP 5293        Goto the fail routine
58:E2 30          SEP #30         8-bit wide Accum.
5A:A2 20          LDX #20         Track $20
5C:A0 01          LDY #01         Side
5E:5A             PHY             Push side
5F:DA             PHX             Push track
60:F4 00 00       PEA 0000
63:F4 AA 52       PEA 52AA        Push disk/volume name
66:22 93 53 00    JSL 005393      Goto the nibble count
6A:8E 9E 52       STX 529E        Store half or nibble count sum
6D:8C 9F 52       STY 529F        Store the other half of the sum
70:A8             TAY
71:68             PLA
72:68             PLA
73:68             PLA
74:68             PLA
75:68             PLA
76:68             PLA             Pull all extra values from the stack
77:90 03          BCC 7C (+03)    Carry clear = no read errors
79:4C 93 52       JMP 5293        Goto the fail routine
7C:A2 21          LDX #21         Track $21
7E:A0 01          LDY #01         Side
80:5A             PHY             Push side
81:DA             PHX             Push Track
82:F4 00 00       PEA 0000
85:F4 AA 52       PEA 52AA        Push disk/volume name
88:22 93 53 00    JSL 005393      Goto the nibble count routine
8C:8D D0 52       STA 52D0
8F:8E A4 52       STX 52A4        Store half of nibble count sum
92:8C A5 52       STY 52A5        Store the other half of the sum
95:A8             TAY
96:68             PLA
97:68             PLA
98:68             PLA
99:68             PLA
9A:68             PLA
9B:68             PLA             Pull all extra values from the stack
9C:90 03          BCC A1 (+03)    Carry clear = no read errors
9E:4C 93 53       JMP 5393        Goto the fail routine
A1:C2 30          REP #30         16-bit wide Accum.
A3:22 76 53 00    JSL 005376
A7:28             PLP
A8:FB             XCE             Back to IIe emulation mode
A9:28             PLP
AB:18             CLC
AC:FB             XCE             Now to full GS mode
AD:08             PHP
AE:C2 30          REP #30         16-bit wide Accum.
B0:A0 FF FF       LDY #FFFF
B3:AD 9E 52       LDA 529E        Load nibble count sum for track $20
B6:CD A0 52       CMP 52A0        Compare to low end bench mark
B9:90 1D          BCC D8 (+1D)    Too low, then goto fail routine
BB:AD 9E 52       LDA 529E        Reload sum for track $20
BE:CD A2 52       CMP 52A2        Compare to high end bench mark
C1:B0 15          BCS D8 (+15)    Too high, then goto fail routine
C3:AD A4 52       LDA 52A4        Load nibble count sum for track $21
C6:CD A6 52       CMP 52A6        Compare to low end bench mark
C9:90 0D          BCC D8 (+0D)    Too low, then goto fail routine
CB:AD A4 52       LDA 52A4        Reload sum for track $21
CE:CD A8 52       CMP 52A8        Compare to high end bench mark
D1:B0 05          BCS D8 (+05)    Too high, then goto fail routine
D3:28             PLP
D4:FB             XCE             Return to IIe emulation mode
D5:28             PLP             Made it this far, it's an original
D6:18             CLC             Set flag for a pass condition
D7:60             RTS             Return to sender
D8:C2 30          REP #30         16-bit wide Accum.
DA:5A             PHY
DB:22 76 53 00    JSL 005376
DF:7A             PLY
E0:4C CB 52       JMP 52CB        Jump to the fail routine
E3:00 00              0000        Nibble count sum storage (track $20)
E5:08 20              2008        Track $20 low end bench mark
E7:02 21              2102        Track $20 high end bench mark
E9:00 00              0000        Nibble count sum storage (track $21)
EB:B0 1D              1DB0        Track $21 low end bench mark
ED:78 1E              1E78        Track $21 high end bench mark
EF:43414C47414D4553 CALGAMES      Disk/volume name in ASCII
```

The call to the protection looks like this on the first block of CALIF.GAMES (block $D2) and is called like this:

```
92:20 FB 51       JSR $51FB       Go do the protection routine
95:90 03          BCC 98 (+03)    Carry set = an original
```

The best way around this protection is to change the JSR (20) to LDA (AD) and the BCC (90) to BRA (80). The program is now completely deprotected. While I have NOT tried it, I think the new copy should be hard disk compatible.

1. Make a copy of the game disk.

2. Edit the copy.

Block	Byte(s)	From	To
$D2	$92	20	AD
	$95	90	80

3. Write the block back to the copy.

IIgs Softkey for...

Serve & Volley
Accolade

■ Requirements

☐ Apple IIgs 512K
☐ 3½" disk copier
☐ 3½" disk editor

Serve & Volley (SV) from Accolade is a tennis game/simulation along the same lines has Hardball and 4th & Inches. It would seem that Accolade has moved to the 3½" disk nibble count routine for it's copy protection. This is nice, as it's a little easier to find and bypass then the protection used on Hardball. Searching the 16 bit system file for "A2 20 A0 01" failed to find the protection routine. After checking all the different files on the disk I found a permanent initialization file (type $B7) called STARTIT in the volume/SYSTEM/SYSTEM.SETUP subdirectory. Scanning this file for the above string did reveal the protection, which looks like this (on the disk, block $DD):

```
0DF:E2 30         SEP #30         8-bit wide Accum.
0E1:A2 20         LDX #20         Track $20
0E3:A0 01         LDY #01
0E5:20 B4 00      JSR 00B4        Go to the protection routine
0E8:B0 0C         BCS F6 (+0C)    Carry set = copy/error
0EA:A2 21         LDX #21         Now for track $21
0EC:A0 01         LDY #01
0EE:20 B4 00      JSR 00B4        Go to the protection routine
0F1:B0 03         BCS F6 (+03)    Carry set = copy/error
0F3:A9 00         LDA #00         Passed, so load the pass value
0F5:60            RTS             Return to sender
0F6:A9 01         LDA #01         Failed, so load the fail value
0F8:60            RTS             And still return to sender
0F9:8E D3 00      STX 00D3        Store the Track number
```

```
0FC:8C D4 00      STY 00D4          Store the side number
0FF:5A            PHY               Push side onto stack
100:DA            PHX               Push track onto stack
101:F4 00 00      PEA 0000
104:F4 D6 00      PEA 00D6          Push disk name/volume
107:22 AC 01 00   JSL 0001AC        Go to the actual protection
10B:8D D5 00      STA 00D5          Store the result
10E:68            PLA
10F:68            PLA
110:68            PLA
111:68            PLA
112:68            PLA
113:68            PLA               Pull extra values from the stack
114:AD D5 00      LDA 00D5          Load the results
117:60            RTS               Return to caller
118:20                20            This is D3 (track) storage
119:01                01            This is D4 (side) storage
11A:00                00            This is D5 (result) storage
11B:484C532E534947 HLS.SIG          This is D6 (disk volume) storage
```

There are two calls to the routine which look like this (still on block $DD):

```
70:E2 30          SEP #30           8 bit wide Accum.
72:20 9A 00       JSR 009A          Do the protection
75:D0 02          BNE 79 (+02)      Branch on fail
77:80 17          BRA 90 (+17)      Passed, goto continue code
79:20 9A 00       JSR 009A          Try the protection one more time
7C:D0 02          BNE 80 (+02)      Branch on fail
7E:80 10          BRA 90 (+10)      Passed, goto continue code
80:38             SEC               Set up for full GS mode
81:FB             XCE               Switch to full GS mode
82:F4 00 00       PEA 0000
85:2B             PLD
86:A9 00          LDA #00
88:48             PHA
89:AB             PLB
8A:5C A6 FA 00    JMP 00FAA6        Mess up program!
```

The best way (I found) to bypass the protection routine is to change the LDX #20 (A2 20) at $E1 to BRA F3 (80 10) which is the pass section of the code. The resulting code will jump straight to the pass section of the code instead of actually going through and reading for the nibbles.

[1] Make a copy of the game disk.

[2] Edit the copy.

Block	Byte(s)	From	To	
$DD	$E1	A2 20	80	10

[3] Write the block back to the copy.

IIgs Softkey for...

The Last Ninja GS
Activision

■ **Requirements**

☐ Apple IIgs 512K
☐ 3½" disk copier
☐ 3½" disk editor

The Last Ninja (TLN) is an incredible game for the GS! Imagine a 3D version of Karateka with weapons and GS SHR screens and you can begin to imagine TLN. The game uses joystick (or keyboard) control and you can save the game at any point. These features will come in handy while trying to solve the arcade/adventure game.

The game disk has a bad block over block $63F (the last block of the disk) and checks it through a simple ProDOS 16 block read. The whole protection looks like this on block $CD:

```
16B:C2 30         REP #30           Start of protection
16D:9C EF C4      STZ C4EF
170:22 A8 00 E1   JSL E100A8        ProDOS 16 MLI call
174:20 00             0020          GET—DEVICE
176:FB C4 00          00C4FB        Parm table location
179:00                00            End of MLI parms
17A:90 25         BCC 1A1 (+25)     On right disk, continue
17C:E2 30         SEP #30           8 bit wide Accum.
17E:A9 01         LDA #01
180:8D EF C4      STA C4EF
183:A9 00         LDA #00
185:85 44         STA 44
187:22 49 00 00   JSL 000049
18B:A9 16         LDA #16
18D:22 DF 0D 00   JSL 000DDF
191:A9 19         LDA #19
193:22 DF 0D 00   JSL 000DDF
197:22 C6 01 00   JSL 0001C6        Ask for Master disk
19B:90 FA         BCC 197 (-06)
19D:C2 30         REP #30           16 bit wide Accum.
19F:80 CF         BRA 170 (-31)     Go get device number
1A1:AD FF C4      LDA C4FF          Right disk, check for bad block
1A4:8D F1 C4      STA C4F1
1A7:9C F9 C4      STZ C4F9
1AA:A9 3F 06      LDA #063F         Bad block number
1AD:8D F7 C4      STA C4F7          Store in parm table
1B0:22 A8 00 E1   JSL E100A8        ProDOS 16 MIL
1B4:22 00             0022          BLOCK—READ
1B6:F1 C4 00          00C4F1        Parm table location
1B9:00                00            End of MLI parms
1BA:90 C0         BCC 17C (-40)     No error, ask for master
1BC:C9 27 00      CMP #0027         $27 = I/O error, bad block
1BF:D0 BB         BNE 17C (-45)     Different error, ask for master
1C1:AD EF C4      LDA C4EF
1C4:F0 0C         BEQ 1D2 (+0C)     An original, start the game
1C6:E2 30         SEP #30           8 bit wide Accum.
1C8:A9 00         LDA #00
1CA:85 44         STA 44
1CC:22 49 00 00   JSL 000049
1D0:C2 30         REP #30           16 bit wide Accum.
1D2:E2 30         SEP #30           Start game/continue code
```

So, from looking at the above disassembly, I changed the REP #30 (C2 30) at 16B to BRA 1D2 (80 65) and had a kracked copy! Now you can upload the program to your hard drive and never pull out the original for a key disk check.

A quick tip: When kracking a disk with a bad block, many times searching for C9 27 will reveal part of the protection. The reason for this is when the smart port (built in GS disk controlling ROM) encounters a I/O error (bad block) it returns a 27 (or 0027 for 16 bit) in the accumulator.

[1] Copy the game disk.

[2] Edit the copy.

Block	Byte(s)	From	To	
$CD	$16B	C2 30	80	65

[3] Write the block back to the copy.

IIgs Softkey for...

Stickybear Alphabet
Optimum Resource

■ **Requirements**

☐ Apple IIgs 512K
☐ 3½" disk copier
☐ 3½" disk editor

Stickybear Alphabet (SA) is an educational program by Optimum Resource that has been released in a GS format. The program has very nice voice systhesis and good graphics. The program comes on two disks with the boot disk having an unformatted track. There is a simple check for the bad blocks from the file ABC.SYS16 on disk one. The protection routine looks like this on block $FE:

```
20:22 A8 00 E1    JSL E100A8        ProDOS 16 MLI
24:20 00              0020          GET—DEVICE call
26:01 19 00           001901        Parm table location
29:00                 00            end of MLI parms
2A:90 05          BCC 31 (+05)      On right disk, continue
2C:20 DC 18       JSR 18DC          Ask for original
2F:80 EF          BRA 20 (-11)      Go start over
31:AD 05 19       LDA 1905          Load device number
34:8D 0F 19       STA 190F          Store in block read parms
37:22 A8 00 E1    JSL E100A8        ProDOS 16 MLI
3B:22 00              0022          READ—BLOCK call
3D:0F 19 00           00190F        Parm table location
40:00                 00            End of MLI parms
41:B0 02          BCS 45 (+02)      Carry set = error/original
43:80 DB          BRA 20 (-25)      Start all over
45:60             RTS               Return to call
```

I checked that routine and it didn't seem to return any special value. Then after more code tracing, I found the only call to the above routine is in the form of 20 B6 18. Changing the JSR (20) to a LDA (AD) resulted in a deprotected copy.

[1] Make a copy of both disks, ignoring errors on disk one.

[2] Edit the copy of disk one.

Block	Byte(s)	From	To		
$F4	$14F	20 B6 18	AD	B6	18

[3] Write the block back to the copy

Softkey for...

4th & Inches
Accolade

■ **Requirements**

☐ Apple IIgs 512K
☐ 3½" disk copier
☐ 3½" disk editor

4th & Inches (FI) is a football game along the same lines as Hardball and Serve & Volley. You can control one team and play against the computer or a friend. The game has nice graphics and is alright. If you really like football, chances are you'll love FI.

It looks like Accolade has switched to the standard nibble counting routine for protection. The protection routine is located on block $C5 of the disk and contained in the file START in the volume/SYSTEM subdirectory. The whole protection routine is as follows:

```
97:C2 30          REP #30           16 bit wide Accum.
99:20 01 79       JSR 7901
9C:E2 30          SEP #30           8 bit wide Accum.
9E:AD 5D EB       LDA EB5D
A1:D0 2B          BNE CE (+2B)
A3:EE 5D EB       INC EB5D
A6:C2 30          REP #30           16 bit wide Accum.
A8:A2 01 02       LDX #0201         MMBootInit
AB:22 00 00 E1    JSL E10000        Tool locator call
AF:B0 6D          BCS 11E (+6D)     If error then bomb
B1:F4 00 00       PEA 0000
B4:A2 02 02       LDX #0202         MMStartUp
B7:22 00 00 E1    JSL E10000        Tool locator call
BB:B0 61          BCS 11E (+61)     If error then bomb
BD:68             PLA
BE:8D 5F EB       STA EB5F
C1:AD 5F EB       LDA EB5F
C4:68             PLA
C5:22 C2 E6 02    JSL 02E6C2
C9:68             PLA
CA:B0 52          BCS 11E (+52)
CC:E2 30          SEP #30           8 bit wide Accum.
CE:A2 20          LDX #20           Track $20
D0:A0 01          LDY #01           Side
D2:20 21 EB       JSR EB21          Jump to the nibble count
D5:8E 61 EB       STX EB61          Store half of result
D8:8C 62 EB       STY EB62          Store second half
DB:A2 21          LDX #21           Track $21
DD:A0 01          LDY #01           Side
DF:20 21 EB       JSR EB21          Jump to the nibble count
E2:8E 63 EB       STX EB63          Store half of result
E5:8C 64 EB       STY EB64          Store second half
E8:AD 61 EB       LDA EB61          Load Track $20 result
EB:C9 00          CMP #00           Check first half
ED:AD 62 EB       LDA EB62
F0:E9 20          SBC #20           Check second half
F2:90 2A          BCC 11E (+2A)     If error then bomb
F4:AD 63 EB       LDA EB63          Load track $21 result
F7:C9 00          CMP #00           Check first half
F9:AD 64 EB       LDA EB64
FC:E9 1F          SBC #1F           Check second half
FE:B0 1E          BCS 11E (+1E)     If error then bomb
00:82 2D 00       BRL 130 (+002D)   Branch to continue code
03:5A             PHY               Push side
04:DA             PHX               Push track
05:F4 02 00       PEA 0002
08:F4 56 EB       PEA EB56          Push disk name/volume
0B:22 83 E7 02    JSL 02E783        Count them up
0F:8D 5B EB       STA EB5B          Store error flag
12:68             PLA
13:68             PLA
14:68             PLA
15:68             PLA
```

```
16:68         PLA
17:68         PLA                    Pull all extra values
18:AD 5B EB   LDA EB5B               Load error flag
1B:B0 01      BCS 11E (+01)          If error then bomb
1D:60         RTS                    Return to caller
1E:C2 30      REP #30                16 bit wide Accum.
20:A2 00 00   LDX #0000
23:A9 DB DB   LDA #DBDB              Load 2 STP's
26:9D 00 00   STA 0000,X             Store them everywhere
29:EB         INX
2A:EB         INX
2B:D0 F9      BNE 26 (-07)
2D:4C 4B EB   JMP EB4B               -= BOMB =-
30:C2 30      SEP #30                8 bit wide Accum.
32:20 5A 78   JSR 785A
35:E2 30      REP #30                16 bit wide Accum.
37:60         RTS                    OKAY to run program!
38:54 4F 4E 59    TONY               Disk name/volume
```

Well that's most of the protection routine. After checking this code, I found the calls to it look like JSR EAB5 (20 B5 EA) on the disk. Searching for this sequence revealed two calls to the protection. After changing both JSR's (20) to LDA's (AD), I booted the disk and the program ran just fine.

[1] Make a copy of the game disk.

[2] Edit the copy disk.

Block	Byte(s)	From	To
$72	$86	20 B5 EA	AD B5 EA
$9A	$40	20 B5 EA	AD B5 EA

Jim Heil

Softkey for...

Spanish for Mastery Software

D.C. Heath and Company.

[1] Copy Master Disk using Super IOB changing the Address Epilog from DE AA to DE DE for reading.

[2] Edit the copy.

Trk	Sct	Byte(s)	From	To
$02	$03	$00	A0	4C
		$02	B9	9B

[3] Copy Both sides of the other disks and the back of the Master Disk with a normal copy routine.

These sector edits stop the program from looking at track $23 for sync info.

Edward Teach

First, for all of you who thought that I was attacking Jack Nissel in COMPUTIST #61, I wasn't. What I meant with the "38 softkeys" remark was that someone with that much ability should share their knowledge. And yes Jack, you are doing better.

Second, I was contacted recently about creating a virus for DOS 3.3 and ProDOS. An interesting question. I chose not to write back to the person, but instead will address my response here. If you think that a disk has a virus, there are a number of things that you can do. I usually boot the disk, then insert another disk in the drive, and catalog it. (I have a DOS that has had the "INIT" command removed). If that DOS is overwritten, then you have to wipe out the DOS from the original disk and place a new DOS on the disk. Boot it again, and catalog the second disk. If no changes occur then you MIGHT be safe. The virus could be located in a file. Next boot the original disk again and get to an Applesoft prompt. Type:

```
10 FOR X=1 TO 255: PRINT CHR$(4) "CLOSE":NEXT
```

Then run this program. If the virus is tagged to DOS commands, hopefully it will bomb on one of the "close" statements. If it still does not show signs of a virus, try Locksmith v6.0, the compare function. Make a copy of the disk and boot the copy. Compare the original with the booted copy. If any differences show up, (they should not) kill the disk. The one final thing that I try is to write protect the disk and boot it. If I end up in the monitor it could mean that the disk was trying to write during booting (decrementing a counter?) and bombed out. I also wipe the disk in this situation. So, in answer to the persons' question, could I write a KILLER DOS? Probably, but I see no reason. The COMPUTIST is here to help people, let's leave the virus to the IBM hackers who don't know what else to do with their time. (I do admit that I wanted to work in something about ONE BAD APPLE spoiling the whole bunch).

Softkey for...

MECC Copy Program

MECC

This is not your typical MECC disk. The MECC copy program will copy a normal disk, a MECC disk and also itself. It has a great RWTS, to be able to handle 3 different prologues. The header prolog was D5 AA FF, and the data prolog was D5 AA FE. The disk also had a nibble check. I have never seen a MECC disk with a nibble check. To deprotect this, we have to defeat the nibble check and put the disk back into normal format. Run SUPER IOB and enter the controller listed below, then sector edit the copy.

Trk	Sct	Byte(s)	From	To
$00	$07	$8B	FE	AD
	$08	$2A	FE	AD
		$98	FF	96
	$0B	$B2	FF	96
$02	$00	$41-42	B0 68	18 EA
		$48-49	B0 61	18 EA
		$4F-50	B0 A5	18 EA
		$AB-AD	20 8D 31	EA EA EA
	$01	$8A-8C	4C AB 30	EA EA EA
	$02	$23-25	4C AB 30	EA EA EA

Controller

```
1000 REM MECC COPY CONTROLLER - 12/01/88
1010 TK = 0:LT = 1:ST = 15:LS = 15:CD = WR:FAST = 1
1020 GOSUB 490: GOSUB 610
1030 GOSUB 490: GOSUB 610: IF PEEK (TRK) = LT THEN 1050
1040 TK = PEEK (TRK):ST = PEEK (SCT): GOTO 1020
1050 TK = 1:LT = 9:CD = WR:MB = 151: ONERR GOTO 550
1060 ST = 0:T1 = TK: GOSUB 490: RESTORE : GOSUB 190: GOSUB
    210: GOSUB 170
1070 GOSUB 430: GOSUB 100:ST = ST + 01: IF ST < 16 THEN
    1030
1080 IF BF THEN 1060
1090 ST = 0:TK = TK + 01: IF TK < LT THEN 1030
1100 GOSUB 230:TK = T1:ST = 0: GOSUB 490
1110 GOSUB 430: GOSUB 100:ST = ST + 01: IF ST < 16 THEN
    1070
1120 ST = 0:TK = TK + 01: IF BF = 0 AND TK < LT THEN 1070
1130 IF TK < LT THEN 1020
1140 HOME :A$ = "ALL^DONE" : GOSUB 450: END
5000 DATA 213,170,255,213,170,254,222,170,222,170
```

Checksums

1000 - $356B	1060 - $9F1B	1120 - $CFA8
1010 - $EA41	1070 - $D582	1130 - $D1F1
1020 - $3164	1080 - $EBED	1140 - $6B91
1030 - $5E3F	1090 - $8F02	5000 - $2CAF
1040 - $3A08	1100 - $5FD2	
1050 - $7B99	1110 - $234F	

Brent Michalski

I hope that this hack helps some people out. It is the first disk that I have successfully cracked and I thought that this one was very easy. I decided to break it because of the registration card that came with the program. I had paid $13 for the program and the card said I could have a backup for the low price of $10 (limit one). To me that is totally unacceptable.

Softkey for...

The Game Show

Advanced Ideas Inc.

■ **Requirements**

☐ 2 blank disks
☐ COPYA
☐ System Utilities (ProDOS 1.1.1 based)

[1] Tell DOS to ignore errors and COPYA side 1 of the disk.

POKE 47426,24
RUN COPYA

[2] Format the other blank disk with the volume name "GAMESHOW01GZMJ" using the ProDOS system utilities disk. When it asks whether you want ProDOS or DOS 3.3, choose ProDOS.

[3] Using the ProDOS system utilities disk again, choose "Copy Files" from the menu.

[4] Use the COPYAed disk as your original and the formatted disk as the blank and copy ALL of the files.

[5] Copy side 2 to a different disk, it isn't copy protected.

That is all I had to do, the game played fine for me and I was even able to make as many backups as I wanted using only COPYA. I don't really know what kind of protection was used, but it was sure easy to overcome once I quit messing around with the JSR $C600 on track $00 sector $0E. I guess it was just built into its own little version of ProDOS.

Note: this disk contains the dreaded ProDOS 1.1.1, so you might want to patch it using Gerald E. Myers patch from COMPUTIST #59, Page 29.

Could anyone please tell me where I can get a copy of "Beneath Apple DOS" through the mail. The selection of books on base over here in England is really depressing.

Thanks for being such a great magazine!

Order the book direct! See page 28, 1st column, 2nd paragraph. .... RDEXed

Jose A. Montano

After a rest and re-examination of my program, "The Bard's Dressing Room III" (TDR3), I discovered four bugs that should be corrected. With the exception of one, these bugs do not seriously affect the operation of the program, and two don't affect the operation at all. They are "cosmetic" bugs.

1. A very embarrassing, though cosmetic, bug in the "Edit Spell Level" routine causes the question "Which Item?" to appear over the border instead of within the border. To fix it all that is required is to replace "VTAB 21" with "VTAB 22" at the beginning of line 4530.

2. In the "List Available Items" routine a very nasty little bug will bomb the program if you try to F)orward the listing past the last screen of available items. To exterminate this one you must change "FI < 230" to "FI < 225" in line 3140.

3. The "Drop Item" routine will drop the last item on the "Carried Items" list if you hit <RETURN> alone when it asks "Which Item". If you like it that way, leave it. If not, you must change line 2920 to read:

```
2920 PRINT "WHICH ITEM: ";: INPUT CW$:CW= VAL (CW$): IF
    CW < 0 OR CW > 12 OR CW$ = "" THEN 2950
```

4. The last is a cosmetic bug in the "List Available Items" routine. The third screen of items (beginning with item #91) shows the second and third columns with a "zig-zag" effect. Unfortunately this is the hardest one to fix because it requires the most typing. Line 3080 must be retyped to read:

```
3080 PRINT SPC( X < 100);SPC( X < 10);X") "I$(X); TAB(
    27); SPC(( X + 15) < 100);X + 15") "I$(X + 15); TAB(
    54); SPC(( X + 30) < 100);X + 30") "I$(X + 30)
```

I'm really sorry for the inconvenience, but I hope you will use the program and let me know through COMPUTIST about any more bugs you might find. I am also interested in enhancements to modify different character attributes. Either make the changes yourself and tell us about it, or let me know what you would like and I'll make the changes and send them to COMPUTIST.

Jack R. Nissel

Softkey for...

Reading Skills 2

American Educational Computer

This title can be deprotected by using Super IOB with the Swap Controller. Use the RWTS of the protected disk to read the original disk then use a normal RWTS to write the information back to your blank disk.

■ **Requirements**

☐ DOS 3.3
☐ A blank disk
☐ Super IOB v1.5
☐ A way to reset into the monitor

[1] Boot DOS 3.3, insert the blank disk, and initialize it to create a slave disk.

INIT HELLO
DELETE HELLO

[2] Boot your original disk and at the Applesoft prompt, reset into the monitor.

[3] Move the RWTS to a safe place, where it won't be destroyed when you boot your slave disk, by entering:

1900<B800.BFFFM

[4] Boot the slave disk.

C600G

[5] After the disk boots and the Applesoft prompt appears, insert your Super IOB disk and save the RWTS to it.

BSAVE RWTS.READING COMP SKILLS 2, A$1900, L$800

[6] Install the controller into Super IOB, run it and copy your original disk to your blank disk. Answer "NO" when asked if you want to INITialize the blank disk.

Controller

```
1ØØØ REM READING COMPREHENSION SKILLS 2
1Ø1Ø TK = 3:ST = Ø:LT = 35:CD = WR
1Ø2Ø T1 = TK: GOSUB 49Ø: GOSUB 36Ø: ONERR GOTO 55Ø
1Ø3Ø GOSUB 43Ø: GOSUB 1ØØ:ST = ST + 1: IF ST < DOS THEN
   1Ø3Ø
1Ø4Ø IF BF THEN 1Ø6Ø
1Ø5Ø ST = Ø:TK = TK + 1: IF TK < LT THEN 1Ø3Ø
1Ø6Ø GOSUB 49Ø:TK = T1:ST = Ø: GOSUB 36Ø
1Ø7Ø GOSUB 43Ø: GOSUB 1ØØ:ST = ST + 1: IF ST < DOS THEN
   1Ø7Ø
1Ø8Ø ST = Ø:TK = TK + 1: IF BF = Ø AND TK < LT THEN 1Ø7Ø
1Ø9Ø IF TK < LT THEN 1Ø2Ø
11ØØ HOME : PRINT "COPY^DONE" : END
1ØØ1Ø PRINT CHR$ (4) "BLOAD^RWTS.READING^COMP^SKILLS^
   2,^A$19ØØ"
```

Checksums

1ØØØ - $356B	1Ø4Ø - $6342	1Ø8Ø - $6CA2
1Ø1Ø - $3565	1Ø5Ø - $ABA3	1Ø9Ø - $9DCA
1Ø2Ø - $617Ø	1Ø6Ø - $2ØCØ	11ØØ - $9A4D
1Ø3Ø - $7771	1Ø7Ø - $28C5	1ØØ1Ø - $9CD3

Softkey for...

School Magic
McCarthy-McCormick:, Inc.

This title can be deprotected by using Super IOB with the Swap Controller.

■ **Requirements**

☐ DOS 3.3
☐ A blank disk
☐ Super IOB v1.5
☐ A way to reset into the monitor

[1] Boot DOS 3.3. Insert the blank disk.

INIT HELLO
DELETE HELLO

[2] Boot your original disk and at the Applesoft prompt reset into the monitor.

[3] Move the RWTS to a safe place, where it won't be destroyed when you boot your slave disk, by entering:

1900<B800.BFFFM

[4] Insert the newly INIT'ed disk and boot it.

C600G

[5] After the disk boots and the Applesoft prompt appears, insert your Super IOB disk and save the RWTS to it.

BSAVE RWTS.SCHOOL MAGIC,A$1900,L$800

[6] Install the controller into Super IOB, run it, and copy your original disk to your blank disk. Answer "NO" when asked if you want to INITialize the blank disk.

Controller

```
1ØØØ REM SCHOOL MAGIC CONTROLLER
1Ø1Ø TK = 3:ST = Ø:LT = 35:CD = WR
1Ø2Ø T1 = TK: GOSUB 49Ø: GOSUB 36Ø: ONERR GOTO 55Ø
1Ø3Ø GOSUB 43Ø: GOSUB 1ØØ:ST = ST + 1: IF ST < DOS THEN
   1Ø3Ø
1Ø4Ø IF BF THEN 1Ø6Ø
1Ø5Ø ST = Ø:TK = TK + 1: IF TK < LT THEN 1Ø3Ø
1Ø6Ø GOSUB 49Ø:TK = T1:ST = Ø: GOSUB 36Ø
1Ø7Ø GOSUB 43Ø: GOSUB 1ØØ:ST = ST + 1: IF ST < DOS THEN
   1Ø7Ø
1Ø8Ø ST = Ø:TK = TK + 1: IF BF = Ø AND TK < LT THEN 1Ø7Ø
1Ø9Ø IF TK < LT THEN 1Ø2Ø
11ØØ HOME : PRINT "YOUR^SOFTKEY^IS^DONE" : END
1ØØ1Ø PRINT CHR$ (4) "BLOAD^RWTS.SCHOOL^MAGIC,^A$19ØØ"
```

Checksums

1ØØØ - $356B	1Ø4Ø - $6342	1Ø8Ø - $6CA2
1Ø1Ø - $3565	1Ø5Ø - $ABA3	1Ø9Ø - $9DCA
1Ø2Ø - $617Ø	1Ø6Ø - $2ØCØ	11ØØ - $C54C
1Ø3Ø - $7771	1Ø7Ø - $28C5	1ØØ1Ø - $DØ82

Addendum to the Softkey for...

Addition Logician
Discovery Lab
Elementary Volume 1
Files on the Apple
Friendly Computer
Right of Way
Science Volume 1
Science Volume 2
Science Volume 3
Science Volume 4
Space Subtraction
Special Needs Volume 2
Subtraction Puzzles
Word Herd: Look-Alikes
Word Herd: Sound-Alikes
Word Wizards
MECC

The MECC controller in COMPUTIST #65, page 19, will also copy these titles.

Softkey for...

Monkey Business
Learning Technologies

■ **Requirements**

☐ A blank disk
☐ A sector editor
☐ A fast copy program

I used Jerry Suchar's softkey for Shutterbug in COMPUTIST #56, page 10, for this title. For an explanation, read his article.

[1] Fast copy your original to your blank disk.

[2] Make the following sector edits to your copy.

Trk	Sct	Byte(s)	From	To
$2Ø	$Ø5	$8A-8B	DØ D8	6Ø EA

[3] Write the sector back to the disk.

Softkey for...

F.M.C. Program
Continental Software

■ **Requirements**

☐ A blank disk
☐ COPYA from your DOS 3.3 system disk

The only thing we have to do, to deprotect this title, is to make a copy with COPYA, after telling it to ignore checksums and the altered epilogues.

[1] Boot your DOS 3.3 system disk.

[2] Tell DOS to ignore checksum and epilog errors and use COPYA to copy the disk.

POKE 47426,24
RUN COPYA

Softkey for...

Mindshadow
Activision

■ **Requirements**

☐ 2 blank disks
☐ A sector editor
☐ Super IOB v1.5

The softkey, by Wayne Williams, in COMPUTIST #47 did not completely deprotect my version. I searched through the sectors on track $00 to see if any additional sector edits should be done and I found an additional byte that needed to be changed for the address prologues. Here is what to do.

[1] Install the controller shown below into Super IOB, run it, and copy both sides of your original disk to your blank disks.

[2] Make the following sector edits to the copy you made.

Trk	Sct	Byte(s)	From	To
$ØØ	$Ø7	$2F-3E	??	all to AA
		$3F-4E	??	all to 96
$ØØ	$ØA	$C5-D4	??	all to AA
		$D5-E4	??	all to 96
	$Ø1	$FB-FD	D9 2F BA	C9 DE EA
	$Ø2	$Ø6-Ø8	D9 3F BA	C9 AA EA
	$ØC	$5B-5D	D9 C5 F3	C9 DE EA
		$66-68	D9 D5 F3	C9 AA EA

[3] Write each sector back to the disk before going to the next edit.

Controller

```
1ØØØ REM MINDSHADOW CONTROLLER
1Ø1Ø TK = Ø:LT = 35:ST = 15:LS = 15:CD = WR:FAST = 1
1Ø2Ø GOSUB 49Ø: POKE 475Ø7,Ø: POKE 47517,Ø:MB = 55: IF
   TK > Ø THEN GOSUB 19Ø
1Ø3Ø GOSUB 61Ø:MB = MB + 16:TK = TK + 1: IF TK = 17 OR TK
   = 33 THEN RESTORE
1Ø4Ø IF MB < 152 THEN GØSUB19Ø: GOTO 1Ø3Ø
1Ø5Ø GOSUB 23Ø: POKE 475Ø7,174: POKE 47517,164:TK = TK
   - 7:MB = 151
1Ø6Ø GOSUB 49Ø: GOSUB 61Ø: IF PEEK (TRK) = LT THEN 1Ø8Ø
1Ø7Ø TK = PEEK (TRK):ST = PEEK (SCT): GOTO 1Ø2Ø
1Ø8Ø HOME : PRINT "NOW^ON^TO^THE^SECTOR^EDITS" : END
5ØØØ REM NO CHANGE ON TRACK Ø
5Ø1Ø DATA 213,151,238
5Ø2Ø DATA 213,154,239
5Ø3Ø DATA 213,155,242
5Ø4Ø DATA 213,157,243
5Ø5Ø DATA 213,158,244
5Ø6Ø DATA 213,159,255
5Ø7Ø DATA 213,166,247
5Ø8Ø DATA 213,237,15Ø
5Ø9Ø DATA 213,238,166
51ØØ DATA 213,239,17Ø
511Ø DATA 213,242,213
512Ø DATA 213,243,223
513Ø DATA 213,244,234
514Ø DATA 213,245,174
515Ø DATA 213,246,254
```

```
5160 DATA 213,150,237
```

Checksums

1000 - $356B	5000 - $1463	5090 - $E2A4
1010 - $2544	5010 - $AB61	5100 - $496E
1020 - $F402	5020 - $4739	5110 - $14FA
1030 - $D637	5030 - $15E9	5120 - $0B7E
1040 - $2181	5040 - $CFDE	5130 - $6FEB
1050 - $3DED	5050 - $82EB	5140 - $3CF7
1060 - $97B9	5060 - $8CE7	5150 - $6D2F
1070 - $742A	5070 - $1799	5160 - $0E5D
1080 - $5BE5	5080 - $3710	

Addendum to the Softkey for...

Zoyon Patrol
MECC

■ **Requirements**

- ☐ 2 blank disks
- ☐ COPYA from your DOS 3.3 system master
- ☐ A sector editor

The softkey in COMPUTIST #53 by Dudley Brett was not complete, 4 additional sector edits have to be made. Here is the complete softkey.

[1] Boot your DOS 3.3 system master.

CALL-151 *enter the monitor*
B8F3:00 *ignore 2nd data header byte*
B8FE:00 *ignores 3rd data header byte*
B925:18 60 *ignores data trailer*
3D0G *back to applesoft*
RUN COPYA

[2] Copy both sides of the original to your blank disks.

[3] Make the following sector edits to side 1 of your copy.

Trk	Sct	Byte(s)	From	To
$00	$0B	$99	AD	AA
$00	$0B	$A3	AA	AD
$00	$0C	$82	AD	AA
$00	$0C	$87	AA	AD
$02	$01	$51-52	AD AA	AA AD
		(if you find that these 2 sectors were AA-AD originally,ignore this step).		
$02	$01	$57-58	AD AA	AA AD

[4] Write each sector back to the disk before going to the next one.

Softkey for...

Adventure Double Feature Vol II
Mystery Double Feature Vol I
Scholastic

■ **Requirements**

- ☐ 2 blank disks for each title
- ☐ Copy II Plus
- ☐ FID from your DOS 3.3 system master

[1] Boot a DOS 3.3 disk. Insert a blank disk in the drive.

INIT HELLO
DELETE HELLO

[2] Use FID to copy all of the files from both sides of the original to a blank disk.

POKE 47426,24 *disable error checking*
BRUN FID

[3] Boot the Copy II Plus sector editor. When you are in the sector editor, press P for Patch Read/Write Routines, select DOS 3.3 PATCHED, press **Return**, then **Esc**. Read track $00, sector $00 from side 2 of the *original* disk.

[4] Press P for Patch Read/Write Routines, select DOS 3.3, press **Return** then **Esc**. Put in side 2 of your **copy** and write the sector back to the disk.

[5] We must remove the remaining part of the protection. Boot your DOS 3.3 system master, then insert side 1 of your copy.

LOAD HELLO
0
SAVE HELLO
LOAD HELLO2
0
SAVE HELLO2
LOAD HELLO3
0
SAVE HELLO3

Note: On one of my copies there was only HELLO and HELLO2, so if you get a "FILE NOT FOUND" message skip that file.

You can put Diversi-DOS on side one if you want to speed up your loading. Your disk is now in normal format except that the screen that says "Scholastic Wizware" when you first boot your original will not appear on your copies. To fix this see the end of the next softkey.

Softkey for...

LOGO Robot
Poster
Scholastic

■ **Requirements**

- ☐ A blank disks for each title
- ☐ Copy II Plus
- ☐ FID from your DOS 3.3 system master

[1] Boot your DOS 3.3 system master. Insert a blank disk in the drive.

INIT HELLO
DELETE HELLO

[2] Use FID to copy all of the files from your original to your blank disk.

POKE 47426,24 *disable error checking*
BRUN FID

[3] Now remove the remaining part of the protection. Boot your DOS 3.3 system master, then put in side 1 of your copy.

LOAD HELLO
0
SAVE HELLO
LOAD HELLO2
0
SAVE HELLO2
LOAD HELLO3
0
SAVE HELLO3

That's it, your disk is now in normal format except that the screen that says "Scholastic Wizware" when you first boot your original will not appear on your copies.

You can put Diversi-DOS on if you want to speed up your loading.

Getting the Title page

Here's how to get the 4 Scholastic titles shown above to load the title page that says "Scholastic Wizware".

Check all of your deprotected Scholastic disks, including the 4 titles shown here, for a binary file called PIC.WIZWARE, (I found it on one of these titles plus on another Scholastic disk). If you have this file, copy it onto Logo Robot, Poster and side 1 of Adventure Double Feature and Mystery Double Feature. Boot your DOS 3.3 system master, then insert the appropriate title in your drive.

LOAD HELLO
6 HOME
7 PRINT CHR$(4) "BLOAD PIC.WIZWARE, A$2000"
8 POKE -16302,0: POKE -16297,0: POKE -16304,0
9 POKE 16384,0: POKE 103,1: POKE 104,64
SAVE HELLO

If you cannot find the PIC.WIZWARE file, then it gets trickier.

[1] Boot your DOS 3.3 system master.

CALL-151 *gets you into the monitor*
800:00 *clears memory at 800*
801<800.95FFM *clears memory from 800 through 95FF*

[2] Put your original disk in the drive.

C600G *boots slot 6*

[3] As soon as the "Scholastic Wizware" page comes up, break into the monitor and search to see where the picture is loaded. It probably starts at $2000. Once you find the start of the file, continue looking through memory until you see where the file ends. If all of the file is in a location in memory where it will not be trashed by a reboot, put in a normal DOS 3.3 disk with a deleted HELLO program and boot the disk.

C600G

[4] Then BSAVE the file to disk with the starting address and the file length.

BSAVE PIC.WIZWARE, A$????, L$????

[5] Now do what I said to do if you found a disk with the file already on it.

[6] If the file is in a location in memory that will be trashed by a reboot then you will have to move it to a safe place in memory, boot the DOS disk, get into the monitor, move the file back to it's original place in memory, get back to Applesoft and then BSAVE it as shown above.

Am I doing better, Ed?

Rich Linville

I use Apple IIgs's in the classroom and have found two hardware problems:

Holding down the Option key while turning on the computer shows a menu that allows you to enter the Control Panel. If a faulty joystick is plugged into the computer, the screen will lock up at the Control Panel menu because one of the fire buttons on the joystick is the same as the Option key. Once you figure out that the joystick if your problem, simply unplug it and restart.

When the IIgs doesn't respond to typing, unplug the cable from the keyboard and plug it in where the mouse was connected. If you are now able to type, the fault could be inside the keyboard. Flexing at this connection causes the soldering to crack. Have someone who knows how to solder electronic equipment remove the 3 screws on the bottom of the keyboard and resolder the cracked connections on the green board inside.

(?) I would like to learn how to deprotect the latest two-sided "Oregon Trail" from MECC and put it onto a 3 1/2" disk. Also, could someone include a routine for this program that will save unfinished games on the disk, because there isn't always enough time in class to finish the adventure. Many of my students would be grateful.

(?) Is there any way to fix "The Seven Cities of Gold" by Electronic Arts so that you can move by the keyboard arrows instead of just the joystick?

(?) Even if I can only put one DOS 3.3 program from a 5¼" onto a 3½" disk for the IIgs, what are the best ways to accomplish it, if possible?

Bud Myers

To Warwick Phillips: The Print Shop saves images as 4-sector binary files in DOS 3.3 format. Publish It! will load only standard Apple hi-res screens from a ProDOS disk. To use Print Shop graphics, for instance, you need to collect eight or nine of them into a single screen image. Clipcapture, Print Shop Lover's Utility Disk, and others will allow you to do this. Next, use the ProDOS Utility disk or Copy II Plus to convert the files to ProDOS. Then load them into Publish It! and print them.

One further complication: Publish It! insists that its screen files be 34 sectors in length; some programs save them in 33 sectors. An article by Roberta Schwartz and Michael Callery, "Speaking of Graphics", in the December issue of A+ magazine, explains how to convert one to the other.

Florida PC Library, PO Box 2878, Leesburg, FL 32748, has five disks of Publish It! fonts at extremely reasonable prices (I forget exactly how much, but all five are less than a single font disk from Turning Point). They are Disks H60 through H64.

To all: The MECC copying system, which comes with a site license from MECC, is an excellent way to copy ALL their disks as well as any other unprotected disks. It makes use of all auxiliary memory in your Apple and where possible loads the entire disk contents into memory, thus allowing you to use both drives as destination drives, and speeding up the copy process considerably.

If anyone has been able to use Print Magic graphics with Publish It!, please share the method:

A. Evans

How about a softkey for Frogger & Pirates! that doesn't require an NMI card?

A.P.T. for...

Pirates!
MicroProse

■ **Requirements**

☐ ProDOS
☐ Pirates! Save disk

This APT is in response to Jason Cobb's request in COMPUTIST #62 and also to anyone tired of sailing against the wind from town to town. The following locations were discovered after BLOADing any one of the four SAVE.GAMEs (A$800, L$700). There is a four step process to use the data.

1. Boot into ProDOS.
2. Load one of the four SAVE.GAMEs.

BLOAD SAVE.GAMEx *(x = 1-4)*

3. Edit appropriate locations.
4. Save the SAVE.GAMEx file.

BSAVE SAVE.GAMEx, A$800, L$700

Town Info

There are 36 towns used in each game, but the names of towns will vary based on the scenario being played (certain year or famous expedition). The alphabetic list of towns starts at $900, each town uses 24 bytes of information:

Location

$900-917	=	town 1
$918-92F	=	town 2
etc.	=	etc.
$C48-C5F	=	town 36

Byte #

1-3	?
4	Ruling country: 0 = spanish, 1 = english, 2 = french, 3 = dutch
5	# of forts: (0 - 15)
6	# of soldiers (divided by 10)
7	# of citizens (divided by 100, minus 1)
8	gold (divided by 1000)
9-12	?
13-24	name of town

Player Info

I could not determine which bytes contained player age, health, and starting skills (fencing, navigation, gunnery, wit & charm, medicine), but I did recognize the following locations:

```
$E02-E03    personal gold (divided by 10)*
$E05        acres of land (divided by 50)
$E06        # of rescued relatives
$E07        mood of crew (the lower the value, the
            better)
$E0D        spanish status
$E0E        english status
$E0F        french status
$E10        dutch status
            1 - ensign, 2 - captain, 3 - major, 4 -
            colonel, 5 - admiral, 6 - baron, 7 -
            count, 8 - marquis, 9 - duke
$E14        spouse status
            0 - single, 1 - shrewish & pestersome, 2 -
            friendly & attractive, 3 - lovely &
            cheerful, 4 - exciting & beautiful
$E19        present town (1..36) minus 1
$E1A        year?
$E1D        in pursuit of: 0 - Vasquez, 1 - Delgado, 2
            - Alvarado, 3 - Mendoza
$E23        # of crew
$E25        # of cannon
$E27-E28    expedition gold (divided by 10)*
$E29        tons of food
$E2A        tons of goods
$E2B        tons of sugar or tobacco
$E2F        # of ships (1..8)
$E34        ship 1
$E35        ship 2
...         ...
$E3B        ship 8
            0 - pinnace, 1 - sloop, 2 - barque, 3 -
            cargo fluyt, 4 - merchantman, 5 -
            frigate, 6 - war galleon, 7 - galleon
            (damaged ships are ship # times 16)
$E50-E58    your character name
```

* value = Hi byte x 256 + Lo byte.

Gary Verbuch

Now that I finally have a little time, I'd like to make a few comments and suggestions that could be kicked around by the editor and readers.

After paging thru about six issues of Computist, there are always quite a few questions concerning various subjects that probably take months to be answered, if ever. One of the most important services a magazine can offer is answering technical questions, whether they be hardware or software. If a few of the current subscribers would agree to offer their name as a reference for one or more specific subjects, questions asked could be directed to an individual by the editor. Quite often, I am asked questions about the Apple, where the answers are hard to come by (hardware or software), especially where the IIgs is concerned. For example, here are a few questions I was asked during the past week from Computist subscribers.

Since the length of the answers were relatively long, I did not include the schematics and example code. If there are any questions concerning any of the below topics covered contact me through Computist

Q. Why does Softswitch use the BBR (Battery Backed Ram) and what is the BBR used for.

Note: I have heard that Softswitch no longer uses the BBR for protection. . . . . RDEXed

A. First, I'd like to mention that this question was sparked by Brian Troha's article about Softswitch in COMPUTIST #61 and while I'm on the subject, I have to admit I am surprised Roger Wagner would allow a protection of this nature to be used on one of his products. I agree with Brian regarding tampering with an area of memory that critical. The reason Softswitch uses the BBR is strictly for disk protection. The byte changed in the BBR is not required by the application to run properly, only to verify the disk. Apple Inc. explicitly warns against using this block of memory for anything other than what it was originally designed for. Disk protection was definitely not an exception.

To further describe the BBR, the buffer (256 bytes) is physically part of the RTC chip (Real Time Clock) and is reserved for the GS Control Panel, ProDOS 16 and Appletalk. For obvious reasons, any change to this block can cause serious if not fatal system errors.

Because the BBR is contained on the clock chip it cannot be addressed in a normal fashion. Data must be read or written serially to or from the clock one byte at a time. Over one half of the BBR is reserved by the system.

At bootup, the GS checks the BBR's checksum. I spent quite some time figuring out the protocol at this point because there was no mention in any of the GS reference manuals that the BBR is read into bank $E1/ ($2C0-$3BF) when verifying the checksum. The Tools to address the BBR are in the Miscellaneous ToolSet.

Q. I've configured a LAN (Local Area Network) using Fiber Optics but the data integrity at the receiver end is inconsistent.

A. After studying the schematic, the problem appears to lie with the power supply being common to the receiver and transmitter. One characteristic of the transmitter is it will generate glitches on the power bus (Vcc) during switching periods. This will inevitably cause false signals at the receiver end. To avoid the false signals, a separate power supply is necessary. Fortunately, the receiver requires only .5 amps. which can easily be supplied with a 9 volt battery and a few discrete components.

If a reader would like a schematic of the above mentioned LAN, I would be glad to forward it. The protocol can easily be configured as a standard RS-232C instead of a 20 ma. current loop. The Fiber Optic transmitter and receiver uses digital encoding and is TTL compatible. The LAN can also be configured without Fiber Optic transmission and achieve transmission rates of up to 9600 Baud at distances up to 1500 feet.

Q. Can you have more than 15 voices when using the Ensoniq chip?

A. The answer to this question is yes, but your programming experience will determine whether you want to attack the problem. There are thirty-two separate oscillators available on the DOC (Digital Oscillator Chip). Each oscillator is capable of generating a separate waveform. The problem lies with the GS firmware. The Sound Tools pair the oscillators together resulting in 15 voices. Why not 16 voices if there are 32 oscillators?. Good question. The Sound Tools utilize the last pair of oscillators to generate time slice interrupts. The only options available to the programmer at this point are to develop your own code to interact with the DOC through the GLU (General Logic Unit) to utilize all 32 oscillators. Easier said than done. If your experience is limited with this type of device, then the Sound Tools are probably your best option. The majority of the housekeeping required to interact with the DOC is dealt with by the GS firmware.

The register responsible for determining how many oscillators are active at any given time is the Oscillator Enable register ($E1), which is normally addressed through the Sound Tools. Simply multiply the number of oscillators you want active (0-31) by 2 and place the result in the Enable register.

Quite often the questions brought to my attention are concerning the Ensoniq Chip in the IIgs. I've noticed there is never any mention of this device in Computist and it's actually a hackers dream. In many ways I consider it more sophisticated than any device used in any Apple II series computers, including the microprocessor. The point I'm trying to make is, I feel Computist would invite new subscribers with experience in software and hardware if the magazine was not enveloped in software deprotection. Although deprotection would stay the theme of the magazine, other topics should be covered in depth. The result would be quite a few more subscribers and a more stimulating magazine for experienced programers, as well as beginners.

I will gladly offer any help I may . My experience with writing 6502 code dates back to 1977 and I have been an active Engineer since 1978. With some of the other subscribers volunteering their help, I believe the columns would become an important addition to Computist.

That's a great idea and I'm all for it. Now, all we have to do is convince other readers to start writing about something other than deprotection. How about it readers, got any other subjects you'd like to talk about? . . . . RDEXed

Compiler Help Needed

I was introduced to LISP about 12 years ago and there are now versions for just about every micro except the Apple II series computers. Since there is no other way to duplicate the power of Lisp, I decided to write my own version, Tiny Lisp. Naming it, was without question, the easiest part of this endeavor.

To complete this project in the very near future, I need at least one other programmers help. Part of the coding is completed.

To keep a block structure, variable definitions will be lexically scoped. In keeping with other Lisp standards, cons space will contain pointers for linked lists and atom space will contain data values that can be organized into lists.

The interpreter functions (generally cons space) are completed and I've just about completed the Editor. With a little help, I would also like to add the ability to evaluate expressions to the Editor, although it is not a priority at this point.

All of the parsing logic is completed on paper, now all I need is some assistance coding the compiler functions. As far as a debugging utility, I have not touched it yet. There are quite a few other subjects that would have to be discussed. Free time to devote to this project is the biggest problem and with one individual helping, it could be completed in a matter of a few of months.

If you have compiler experience but you are not too familiar with Lisp, your coding experience is still needed.

Here are a few programs I was recently asked to deprotect;

Softkey for...

Mastery Arithemetic Games

Mastery Developement

■ Requirements

☐ DOS 3.3
☐ COPYA
☐ Blank Disk

Mastery Arithemetic is an excellent introductory math package for children. The graphics and animation sequences enhance the clear and concise examples covered.

The protection on this disk is a simple format alteration where the address prolog alternates between D5 AA 96 and D4 AA 96 and invalid data checksums are used.

Use COPYA with some patches to RWTS to copy the disk.

```
RUN COPYA
©C
POKE 47426,24          ignore checksum errors
POKE 47444,41          ignore 1st addr prolog byte
POKE 47445,0
70                     Delete line 70
RUN                    Restart the program
```

Softkey for...

Algebra 1
Algebra 2

Edu-Ware

■ Requirements

☐ Starter Kit (Super IOB) or FID
☐ Blank Disk

The fundamental concepts of Algebra are divided into modules and covered in depth. I recommend either program as an excellent tutor for a student learning Algebra.

Again the protection consists of format alterations and checksum errors. The address epilog is DA AA instead of DE AA.

[1] Boot the Starter Kit disk and get into BASIC.

```
POKE 47426,24          ignore epilog & checksum errors
INIT EDU-WARE          Init the blank disk
```

[2] Load Super IOB and merge the fast controller. Change TK in line 1010 to 3 instead of 0 and start the program.

```
LOAD SUPER IOB
EXEC FAST.CON
1010 TK = 3: LT = 35: ST = 15: LS = 15: CD = WR: FAST = 1
RUN
```

In step 2, FID can be used to transfer the files if you prefer not to use Super IOB.

Softkey for...

Sargon III

Hayden Software

■ Requirements

☐ Any Copier
☐ Blank Disk

As far as chess programs go, Sargon III is probably one of the best chess programs available for the Apple. Unfortunately, it is protected and I must admit I was kind of surprised when the parms for the popular Bit Copiers would not work for this version. Possibly it's a newer release, so I decided to deprotect it. I don't usually have the time to deprotect software anymore and I was hoping this would at least be a challenge. The last Apple product I found somewhat challenging was Flight Simulator II.

While describing the deprotection process to a friend he wrote everything down to send to Computist. My version of the process consists of a couple of sentences but maybe the longer version might help some of the readers.

The Search

First things first, lets make a copy to work with. Any fast copier will do at this point, and if there are any format alterations, they will show up during the copying process.

Okay, the format is not altered and the disk copied without a problem. Now boot the copy to get some idea of what and where the protection is. First, a prompt appeared during the boot, so we can assume somewhat of a standard DOS. Second, the disk check appears to be quite some time after the boot, so a Boot code trace is not a good approach.

I decided to start with a normal DOS 3.3 environment and tried cataloging the disk. Approximately 40 files show up in the directory including a binary HELLO. To insure the Hello file is in fact the first file BRUN by DOS, I renamed it to "H" and rebooted the copy. "File not found" appeared, so, lets BLOAD the HELLO file and see what the code looks like. First, get into the monitor and find the load address.

```
BLOAD HELLO
CALL -151
AA72.AA73
```

This is the load address of the last BLOADed file (in lo-byte, hi-byte order). Now type the load address and an "L" to list the code. Here's what I found.

```
1000 LDA #$C0              needed later for return
     STA $03F2             thru $FA62
     LDA #$13
     STA $03F3
     EOR #$A5
     STA $03F4
     JMP $1638
```

This code sets the reset vector to point to $13C0 and jumps to $1638. Nothing unusual, so lets continue at $1638.

```
1638 LDA #$15
     PHA                   Put the return address
     LDA #$F5
     PHA                   for the RTS on the stack
1648 LDA #$56              set pointers
     STA $08               for indirect
     LDA #$11              addressing to
     STA $09               EOR next stage
     LDA #$96
     PHA
     LDY #$00
1655 PLA
     EOR ($08),Y           actual EOR of
     STA ($08),Y           next stage
     PHA
     DEY
     BNE $1655
     INC $09
     LDA $09
     CMP #$15
     BNE $1655
```

After the code is EOR'd, the return address is pulled of the stack and execution resumes at $15F6. If you are not sure what is on the stack for an RTS, the following bit of code will pull the stack twice for you and place the return address at $300;

```
68        PLA
8D 00 03  STA $300
68        PLA
8D 01 03  STA $301
4C 59 FF  JMP $FF59
```

To see the results of the code being EOR'd type the following,

```
167D:0 N 1638G
```

This will run the code that EOR's the next stage and break execution before continuing to $15F6.

Now, continuing on to $15F6 is where we find the calls to the signature check. Again, another address is stuffed on the stack ($FA61). Since the protection consists of over 500 bytes of code, I will not include all of it.

```
15FC JSR $1570             position head thru RWTS
15FF LDA $C0EA             select drive 1
1602 LDA $C0E9             turn drive on
1605 LDA #$AB              set comparison table
1607 STA $1562             for nibbles read
160A LDA #$AF              from disk
160C STA $1561
160F JSR $159A             read signature off disk
1617 JSR $1570             position head
161A LDA $C0E9             drive on
161D LDA #$04              modify branch in signature
161F StA $15D4             check to branch to $15D9
```

I'm including this memory wipe routine because similar routines are often used and can be a real headache when deprotecting software.

```
15D9 PLA
     PLA
     PLA
     PLA
     LDA $C0E8             turn drive off
     SEI
     LDA #$00
     TAX
15E4 STA $15F6,X           zero out memory
15E7 INX
15E8 BNE $15E4
15EA STA $14E4,X
15ED DEX
     BNE $15EA
     DEC $15EC
     JMP $15EA
```

The $04 placed at $15D4 alters the branch to a memory wipe at $15D9. If the proper nibbles are found on the disk, the branch at $15D4 is bypassed and execution resumes thru $FA62. If the vectors don't point at $13C0 then the system does a warm start to BASIC. ($13C0 was stuffed in the reset vectors at $1000).

At this point, it's time to decide what code is needed for the program to load properly. NOPing out a few areas of code around $15F6 will produce a working copy. The code there is only used to check for the original disk.

The Crack

[1] Make a copy of the original with any copier.

[2] Boot the copy and hit ⊖ **Reset** when the prompt appears.

[3] Load the Hello file and enter the monitor.

```
BLOAD HELLO
CALL -151
```

[4] Enter the patch and run the code at $1638. This will generate the code we need to load the rest of the program.

```
1000:4C C0 13
167D:0
1638G
```

[5] Save the EOR'd code as the new Hello file.

```
BSAVE HELLO, A$1000, L600
```

Put the original away.

Bill Jetzer

Displaying Hi Res and Double Hi Res Pictures from BASIC

To Andy Borne's question (COMPUTIST #62) about how to display DHR pictures from BASIC.

It is easier to display HR pictures, so I'll cover that first. Here is the HR slide show program. It works under both DOS 3.3 and ProDOS and requires only 48K of memory:

```
10 TEXT : HOME : PRINT CHR$ (21):A = 8192: ONERR GOTO 50
20 READ F$: PRINT CHR$ (4) "BLOAD" F$ ",A" A
30 POKE 49234,0: POKE 49239,0: POKE 49232,0: POKE 49236
   + 1 * (A > 8192),0
40 A = 8192 + 8192 * (A = 8192): GOTO 20
50 POKE 49168,0: WAIT 49152,128: GET A$: HOME : TEXT :
   END
60 DATA PICTURE1,PICTURE2,PICTURE3,ETC
```

Checksums

10 - $6BE8	30 - $D812	50 - $90A3
20 - $BBF7	40 - $AE95	60 - $F393

Line 10 clears the text screen and initializes the variable A to 8192, which is the BLOAD address for page 1. It also sets ONERR to go to line 50 after the last picture is loaded.

Line 20 loads the next picture onto either page 1 or page 2, depending upon the value of A.

Line 30 turns on 1) full screen, 2) HR, 3) graphics, 4) the correct HR page.

Line 40 toggles A between 8192 (page 1) and 16384 (page 2) and loops back to line 20.

Line 50 waits for a keypress while the last picture is displayed and then exits to BASIC.

Line 60 is the data line. Put the names of your pictures here.

DHR pictures (stored in Dazzle Draw format) are a little more difficult to display. If you have any important data stored in the /RAM disk, you must save it to disk so as not to destroy it.

When you BLOAD a DHR picture, it is loaded at $2000 and goes up to $5FFF in memory. To display the picture, main HR page 1 must be moved to aux page 1 and main page 2 must be moved to main page 1. With 80 column firmware active, the enhancer at $C05E, which allows DHR to be displayed, must be switched on. To display page 2, main page 1 must be moved to aux page 2 and main page 2 is left where it is. While displaying page 2, it is important that nothing is printed to the screen, even the cursor during a GET. For this reason a WAIT must be used to get a keypress.

Here is the DHR slide show program. It works only under ProDOS and requires 128K of memory:

DHR Slide Show program

```
10 TEXT : HOME : PRINT CHR$ (21) :B = 769:E = 775:A = 8192:
   ONERR GOTO 80
20 FOR X = 768 TO 790: READ Y: POKE X,Y: NEXT : DATA
   160,32,132,61,132,67,160,63,132,63,160,255,1
   32,62,200,132,60,132,66,56,76,17,195
30 READ F$: PRINT CHR$ (4) "BLOAD" F$ ",A" A ",L$2000"
40 POKE B,32 + 32 * (A > 8192) : POKE E,63 + 32 * (A > 8192) :
   CALL 768
50 PRINT CHR$ (4) "BLOAD" F$ ",B$2000,A" A
60 POKE 49234,0: POKE 49239,0: POKE 49246,0: POKE
   49165,0: POKE 49232,0: POKE 49236 + 1 * (A > 8192),0
70 A = 8192 + 8192 * (A = 8192) : GOTO 30
80 POKE 49168,0: WAIT 49152,128: POKE 49247,0: POKE
   49164,0: GET A$: TEXT : END
90 DATA PICTURE1,PICTURE2,PICTURE3,ETC
```

Checksums

10 - $9E31	40 - $655A	70 - $CE5C
20 - $5A21	50 - $E04B	80 - $6E5D
30 - $117D	60 - $BDFE	90 - $975E

Line 10 clears the text screen and initializes the variables B and E for the AUXMOVE routine. A is set to 8192, and ONERR is set to line 80.

Line 20 pokes in the AUXMOVE routine.

Line 30 loads the first half of the next picture into memory.

Line 40 moves the first half of the picture into aux memory.

Line 50 loads the second half of the picture.

Line 60 turns on 1) full screen, 2) HR, 3) enhancer, 4) 80 columns, 5) graphics 6) the correct HR page.

Line 70 toggles A between 8192 and 16384 and loops back to line 40.

Line 80 WAITs for a keypress while the last picture is being displayed, and then exits to BASIC.

Line 90 is the data line. Put the names of your pictures here. Have fun with them.

To George Sabeh: Some books that I use quite frequently are the "Apple II Reference Manual" and "About Your Enhanced Apple IIe: Programmer's Guide." Both of these contain F800 ROM listings, and the enhanced IIe manual also contains $C100-$CFFF ROM listings. "The DOS Manual" contains a detailed example on how to use the RWTS, and much information about the VTOC and the way files are stored on disk. "Beneath Apple DOS" and "Beneath Apple ProDOS" contain much information about diskette format and using the disk drives independent of DOS and ProDOS in addition to just about everything you ever wanted to know about DOS or ProDOS.

To Scott F. Earnest: the Merlin Pro assembler comes with a program called SOURCEROR.FP that disassembles Applesoft. It works by scanning the resident copy of Applesoft present in your computer, so it works with the Apple II, II+, IIe, and IIc. If you have a Laser, you're probably out of luck. It will print out 105 pages (including the symbol tables) of Applesoft, complete with comments, in about an hour and a half (at 80 cps). You might use an entire printer ribbon, but it won't cost you $50. However, Merlin Pro will cost you about $80. But if you are in the market for an assembler, and you won't find too many that are better than Merlin Pro, it would definitely be worth the price.

Kay Jun

I would like to thank **all of the writers of softkeys** for the softkeys you have been sending to COMPUTIST. Jack Nissel and Edward Teach, I've become your fan!. I've learned so much from reading COMPUTIST.

I work with and teach programming, word processing, and data processing on IBM, Radio Shack(dinasours), and Apple computers, all at the same time! How about that for a headache? Needless to say, I now have COMPUTIST helping to cure my headaches.

I spend thousands of dollars on software each year. In some cases I have purchased software packages for $1200 and then found out I must spend another $275 dollars for backups. In the past, I have been forced (most of the time) to allow 6-13 year old students to use original disks! Until I found COMPUTIST, I gritted my teeth and cried on many occasions.

Edward Croft Jr, (COMP. 64) said it all. The deprotection game can be ugly to some people, but I do not feel like a pirate or a person who is committing a crime. I do feel like a banker protecting my many investments.

I have not been at the deprotection game long, but I hope some COMPUTIST readers can use what I have to offer this time. I didn't want to send anything in until I had learned enough to be able to explain the logic of what I was doing, but I really can't go into a lot of detail about each deprotection because I'm not absolutely positive that I can explain what I've done on some of the disks.

Softkey for...

Microzine Jr. #3. Disk 1 & 2
Microzine #28
Microzine #29
Scholastic

These are ProDOS based but strangely the following made a copyable disk. (Please, someone tell me why?)

[1] Boot a DOS 3.3 disk.

POKE 47426,24
RUN COPYA

...answer the prompts for each disk you wish to copy.

Softkey for...

Microzine #27
Decimals Disk 1 & 2
Scholastic
ACT Preparation
CBS Software

[1] Boot a DOS 3.3 disk.

RUN COPYA
⊖C
CALL -151
B993:00
B99D:00
B957:00
⊖C
70
RUN

...answer the prompts to copy both disk 1 and disk 2.

Softkey for...

Mini Putt
Accolade

Us a sector editor with a search routine to scan the disk for 80 04 C2 30 38 6B (I found it on track 02, sector 0A, side 01) and change the 04 to 21. It will branch to RTL and bypass the protection check. After I had already removed the protection on my disk I saw a couple of articles on removing the protection in COMPUTIST that involved a lot of changes. I found that the one change had worked perfectly. (Could I possibly have some feedback on this from some of you?)

Softkey for...

Geometry Disk 1-5
Educational Courseware

[1] Boot a DOS 3.3 disk and use COPYA to copy all 5 disks.

RUN COPYA
⊖C
CALL -151
B942:18
BE48:18
3D0G
70
RUN

Softkey for...

Algebra Disks 1-6
Equations II
Binomial Multiplication
Equations
Factoring Algebraic Expressions
Graphing Linear Functions
Simutaneous Linear Equations
Microcomputer Workshop Courseware

[1] Boot a DOS 3.3 disk and use COPYA to copy the disks.

RUN COPYA
⊖C
CALL -151
B942:18
BE48:18
B993:00
B99D:00
B957:00
70
RUN

There is one other disk, Solving Quadratic Equations, that I was unable to make COPYA-able. Maybe someone can help?

The following is a list of disks that I need help with:

Writing Skills Series by Edu Ware @1984
ACT Natural Science Series by NASSP
Science Series: Earth Science, Body Defenses, Nervous System by Prentice Hall
Carmen-USA and -World (Not one of the articles helped.)
Baron's Computer ACT
Let's Explore Basic by Milton Bradley
Bank Street Writer III by Scholastic (None of the articles helped.)
Go to the Head of the Class by Milton Bradley (Media Materials)
Game of the States by Milton Bradley (Media Materials)
Micro Power and Light software
The Teacher's Tool Kit Series by Hi Tech of Santa Cruz (the article in COMPUTIST #65 pg. 18 did not work.)
Montana Reading Program (The article in COMPUTIST #63 pg. 8-9 didn't work.)
Mavis Beacon Teaches Typing 5¼" disks. (All of the protection removal is for IIgs versions, didn't help.)

Jeff Hurlburt

The PRODUCT MONITOR

Ratings

☆☆☆☆☆	SUPERB
☆☆☆☆	EXCELLENT
☆☆☆	VERY GOOD
☆☆	GOOD
☆	FAIR
✋	POOR
✋✋	BAD
✋✋✋	DEFECTIVE

Impossible Mission II

$44.95 Epyx

☆

■ **Requires:**

☐ 512K Apple IIgs
☐ one 3½″ drive
☐ joystick recommended

As you may recall, Elvin Atombender was launched upon his life of crime when, in his formative computer brat years, a power glitch wiped out months of work. Vowing to "make the world pay", the crazed genius aimed to trigger a nuclear cataclysm. The good news is that he was stopped ("Impossible Mission"-1); the bad news is that Elvin is loose again (!), richer, meaner, and more determined than ever to exact his revenge.

Epyx's "Impossible Mission II" casts you in the role of an acrobatic CIA agent with just eight hours to shut down yet another project for world destruction. The site is Atombender Industries' sprawling new HQ complex (eight 4-6 room towers surrounding a core tower topped by a penthouse Control Room) near downtown LA. To prevent the launching of the world's nuclear missiles, you must 1.) crack, at least, six safes to obtain the complete musical key for the core tower's elevator; 2.) reach Elvin's main terminal; and 3.) 'pull the plug'.

As in IM-1, each room of the new complex is a cleverly contrived action puzzle composed of terraces, walls, sliding platforms, and lifts— all keyed to one or more in-room terminals and linked to the patrol activities of robot guardians. Here, hidden in furnishings and equipment, is where you can find the items needed to unlock the larger puzzle represented by a tower. Searches may yield Terminal Control Icons (to deactivate robots, turn on lights, reset lifts, etc.), bombs for blowing safes, mines (to blow robots), and Security Key digits. Whether or not you succeed in cracking a tower's safe, entry to an adjoining tower is blocked unless you possess the correct 3-digit 'key' code.

The better your grasp of how a room 'works' (and the more control icons you've collected), the less you will need to depend upon arcade ace-class expertise. A well thought-out set of on-screen locator, music recording, and code-cracking tools helps too; but, basically, the 'trick' is still to run, leap, and ride your way through the room puzzles without getting lasered, crushed, or otherwise reduced to a grease spot. (Each fatality deletes several minutes from the countdown timer; and— an undocumented feature— too much time without progress, ends the game.) Speedy Save/Restore functions let you preserve hard-won gains; and, win or lose, an on-disk five-position High Scores roster recognizes those special 'great performances'.

Presented in full-color super-res "Impossible Mission II" delivers crisply detailed side views of elevator and room areas, realistic agent animation, and hordes of nifty little robots. Control response is smooth; but movement variety qualifies as merely adequate. You can not, for example, change the length of a jump; nor can you easily separate "search" and platform movement functions. Since the program is 'designed around' its controls, winnability is not seriously impacted; but, for best play in a high resolution 'action puzzle' environment, it's time Epyx moved beyond single-button IIe-class inputs.

The new "Impossible Mission" wraps you in digitized sound, offers an exciting challenge, and, all in all, manages to seem very "Possible". (Victory is just around the corner; you see the 'light at the end of the tunnel', etc. etc.) For those who crave long-playing arcade-spiced adventure, this is addictive stuff.

Jinxter

by Magnetic Scrolls

$34.95 Rainbird

☆

■ **Requires:**

☐ 64K Apple II series
☐ one 5¼″ drive
☐ Second drive and printer optional

For years, the legendary Bracelet of Turani maintained good fortune for Aquitania's populace and prosperity for the land's stochastic executives. Indeed, the bureaucracy charged with "directing the bracelet's luck" —a task formerly handled by Turani alone, in his spare time— has become a paper shuffling marvel and a major employer. When it becomes apparent that someone is compromising the bracelet's efficacy, Aquitanian society is, quite naturally, shaken to its foundations.

As the one chosen to put things aright, your first task is to make sense of one of Apple text adventuring's wackier, more convoluted, scenarios. Originally, you see, Turani crafted the bracelet as a defense against the mischief of Aquitania's green witches. It worked so well that the witches agreed to 'cool it'. The bracelet, however, also produced an unforeseen bounty of good luck, just by virtue of its presence in Aquitania. Thus, in a mood for compromise, the witches proposed to guard the bracelet. They would tolerate its spell dampening influence, preferring this to having some renegade mortal access the full powers of such a potent artifact. Since witches can't stand to touch even one of the five charms on the bracelet, the non-witches agreed, seeing the deal as the perfect way to prevent tampering with Aquitania's 'golden goose'.

Everything worked fine until one witch, Jannedor, conspired with greedy mundanes to have the charms removed and dispersed throughout the land. Now, unless YOU can locate all the bracelet's pieces and reassemble them, a powerful spell will destroy the weakened artifact, free the witches of any restraint, end Luck, AND put numberless bureaucrats out of work!

Distinctly Zorkian in flavor, "Jinxter" is a two-diskette multi-setting challenge embracing not only a town's shops, dwellings, (towers, entertainment centers, etc.), but, in addition, the surrounding countryside, woods, mines, a weird temple, and more. Here and there, you are assured, each of the charms may be discovered; and, just maybe, the special powers of some can help locate the others. Hardly 'a night's work', this is a large adventure loaded with enough diversions, puzzles, and places within places to rate, at least, an "Intermediate" on the Infocom difficulty scale.

"Jinxter" support goodies include two "Old Moose Bolter" coasters, a copy of the background memo (from the Deputy Under-Secretary's Assistant General Secretary), and, hot off the press, the latest The Independent Guardian ("Quality News for the Hard of Thinking"). Packed with meaty current events items, the latter also supplies numerous examples of how one interacts with the game's parser AND several pages of coded solutions for specific problems. Should you become 'dead stuck', entering the appropriate code during play produces a translation. (Note: The I.G. fails to note one important detail; you must be at the site of the problem when entering the code.)

Finally, it seems, someone besides you-know-who has discovered 'the secret' for producing first rate text adventures. In Rainbird's "Jinxter" you will find the excellent parsing, engaging puzzles, picturesque descriptions, and humor that make quality 'words-only' adventuring a special treat.

Fast Frames, Updates, Etc.

NOT a Bug

Some years back, my dad caught what appeared to be a sure bug in the original "Ultima": when an attribute's value increments beyond 99, it resets to zero. Well, we called Richard Garriot/Lord British and were told that the recycling is, most certainly, not a bug. No indeed, Richard explained, it is a "feature" intended "to keep players from being too greedy".

Evidently, there are many fewer bugs and many more features floating around than one might suppose. Last month, I reported an apparent "Wizardry V" bug which produces unexpected Good-to-Evil shifts in orientation. Since the game does not permit forming a party including both Good and Evil members (though, interestingly, it will not boot out misfits at the time the shift occurs), the result can be the loss of valuable personnel. A problem? Yes. A bug? No.

According to Sir Tech's Brenda Garno, when the manual says: "Certain peaceful monster types may offer your party a truce" (p.58), it is referring to those times you encounter messages such as "You see a wandering group of Arch Devils Attack/Leave alone". Choosing to attack is viewed as evidence of evil inclinations. (Like, clearly, you'd have to be really corrupt to beat-up on "peaceful" Arch Devils.) Due to a random factor, just one such attack may trigger a shift; or, it may take several.

Roadwar 2000

This SSI map adventure ($44.95, for 512K IIgs) is set in a post-disaster North America ruled by roving road gangs and petty tyrants. Beginning as a small-time ganglord, you strive to recruit and upgrade your 'troops', win road battles, and secure sources for food, medicine, gasoline, and other vital supplies. Your objective, in short, is to become a bigtime ganglord. The game's main map, a beautiful scrolling pictorial affair in full-color super-res, is so well done that it is very hard to believe that the rest of the game doesn't 'click into place'; but, in numerous tries, it never has. One weakness is documentation: it leaves a gaping information vacuum between the scenario story and listings of raw statistics. A more serious problem becomes evident once you have figured out what's going on. Somehow, when it comes to planning and other activities, especially combat, the information or display you need just isn't there. So, for "Roadwar 2000", neither am I.

Tax Help

Should you or 'a friend' have put off income tax return preparation to the last minute, it may be time for some high-powered help from your computer. "J.K. Lasser's Your Income Tax 1989" ($75, from Simon & Schuster) runs on 512K IBM compatibles (e.g. PC, PS/2, Compaq, and Tandy) with DOS 2.0 or later and two 5¼″ or 3½″ drives and/or a hard disk. The package, which includes a 40-page manual and Lasser's telephone directory-sized book, is designed to help you organize and file your return as well as to plan the best tax strategy for the coming year. Thus, for instance, you can read-in files from "Quicken", compare 1040 tax scenarios (e.g. filing joint versus separate returns), and even print IRS approved 1040 forms. Aside from, in all probability, a discount on next year's update, bonuses include Tax Alert Hotline support and a 3-month trial subscription to the J.K. Lasser Tax Service Newsletter.

Masters Help

If, on the next visit to your favorite software emporium, you should happen to purchase one of the Epyx "Masters Collection" games, there is a new accessory which may be of real interest. Epyx's own Advanced Hint Book delivers 46 pages of no-nonsense, low-fluff player assistance for "Legend of Blacksilver", "Space Station

Oblivion", "LA Crackdown", and "Sub Battle Simulator". Coverage is on a 'what you need is what you get basis'; so " 'Blacksilver", a large fantasy adventure, commands a full 27 pages packed with general and specific hints plus a complete set of maps. (The more specific clues are partially alphabet-substitution encoded to facilitate accessing only the information of interest.)

" 'Oblivion" players get some 'start up' details, specific sector-by-sector directions, and a copiable polyhedron map cut-out; while, for "Sub Battle'", the booklet supplies strategy and tactics hints, including several diagrams. Helpful "tips" begin the " 'Crackdown" entry, followed by a detailed Case Report flow chart. Except for " 'Crackdown", each section opens with a brief 'insider discussion' of the ideas behind the game's creation. Featuring red and boldface highlighting plus expertly done graphic aids, this is a quality production; and, at $7.99, a bottleneck-busting bargain.

Platoon

Featuring Rambo-theme arcade action, Data East's "Platoon" ($34.95, for 128K Apple II) delivers loads of play spread across six challenging episodes. Ideally, your aim is to bring your five-man team, intact, through all six tests; realistically, you have specific objectives to accomplish (e.g. blow up a bridge, etc.) and, of course, points to score. (Unfortunately, though, there is no on-disk record of high scores.)

Thanks to excellent game speed and responsive four-quadrant controls, the single soldier figure representing your party can race along jungle paths, leap over traps, kneel and blast enemy troops as they drop out of trees, etc., all in colorful double-hires. Each episode incorporates several screens— nice because soldier figures and terrain features can be large enough for decent detail; but also something of a problem, when it comes to getting where you need to be. The game's maze-like trails seem to encourage mapping; yet, with no PAUSE option and constant enemy pressure, mapping is nearly impossible. "Platoon" easily ranks among the better IIe format gaming parlor translations. As is almost always true of such products, the game would be even more fun if translators could remember that no one is feeding-in quarters for each replay.

Vendors

Data East: 470 Needles Drive, San Jose, CA 95112 (408-286-7074)
Epyx: 600 Galveston Drive, P.O. Box 8020, Redwood City, CA 94063 (415-366-0606)
Rainbird Software: 3885 Bohannon Drive, Menlo Park, CA 94025 (415-322-0412)
Simon & Schuster: 1 Gulf and Western Plaza, New York, NY 10023 (212-373-8142)
Sir-tech: P.O. Box 245, Charlestown Mall, Ogdensburg,NY 13669 (315-393-6633)
Strategic Simulations Inc.: 1046 North Rengstorff Ave., Mountain View, CA 94043 (415-964-1353)

Michael Reese

I tried to use a softkey on page 16-17 of COMPUTIST #59 (Sept., 1988) written by Mike Egnotovich, and could not get the early and straight forward instructions to work. I checked the BUGS sections of all later issues, but could not find any update. I think there may be an error, and wanted to ask if you could refer my letter to Mike Egnotovich to check on the possibility of an error?

The problem occurs after the 4th monitor step, where we are supposed to boot the program disk by typing 9600G (Step #4). That is very simple and logical, but my computer just locks up. I have retried many times, with the same results. The usual * prompt disappears, and does not return unless I press **Reset**, followed by a CALL-151. The drive does turn on if I type the original address (Step #2) at $C600G.

Is it possible that there is an error in the first few steps? Your assistance would really be greatly appreciated.

I don't mean to pick on you, Michael, but it's letters like this one that get ignored by most readers. So please, everyone, when you ask a question, give all the information you can about your system and what you tried, even if it doesn't seem to relate to your problem. Let the reader who is trying to answer your question, decide what is or is not important.

Here is my best guess, based on the info you provided. Since the boot0 code is being modified, the proceedure should work, at least past step 4. I can think of two reasons that it wouldn't. First, your disk controller is not in slot 6. The boot0 code uses the its' page alignment to determine what slot to access. (IE. A controller in slot 5 would need its code moved to $9500 or $8500 or $7500, etc.) Second, the boot0 code may not be the same. If you are using other than Apple drives or if you are using an Apple IIc, the boot code and the necessary patch would be different. Again, I'm only guessing.

If you would like to write to Mr. Egnotovich, the procedure is for this is printed in the front of each issue, under writing to RDEX. .... RDEXed

A. L. Head,Jr.

A Treatise on Deprotection

(using Gradebook III)

Softkey for...

Gradebook III
Schoolhouse Software

■ **Requirements**

- ☐ Apple IIe 48K min.
- ☐ COPYA and FID
- ☐ NMI capability
- ☐ 2 drives preferred

Foreword

Each issue of COMPUTIST reproduces a part of the copyright law that guarantees the owner of a software program the right to make certain copies of the program for his own adaptation or for archival purposes. In accordance with these owner rights, methods for doing this copying need to be understood. The general purpose of this article is to demonstrate some of these methods. Conversely, a programmer (software developer or seller) that has slaved over his particular masterpiece wants some just reward for his labor. He does not want unlawful copies made and distributed. The copyright law strikes the balance between the needs of the owner and programmer.

Most users, including programmers, have either possessed or do possess pirated software. However, often a pirated copy leads to a bona fide purchase to get the latest version and the documentation. As for me, I must have the latest and greatest of the software that I use. So, in the end the programmer benefits and some of us have fun trying to beat the protection. Pirated software provides potential purchasers with the opportunity to try the software prior to purchase. How often does one buy a canned program only to find that it does not meet his needs when he uses it? This subrosa evaluation of software does not cost programmers as much as they advertise because of the ensuing purchases of worthy programs that never would have happened otherwise. All of this is not a defense for unlawful copying; it is, however, an observation of conditions as they exist.

There is no end to the protection schemes that can be devised. This fact is what makes the challenge to hackers so intriguing. Some of us have been known to buy a $25 to $30 game just for the challenge of the cadillac protection scheme it may offer. The real game is the quest for and the removal of the protection.

Another observation is that today's hacker is tomorrow's programmer and vice versa. Come on people! Most of us have been on both sides of the protection/deprotection issue. All of you hackers have your own protection scheme that is certainly the greatest ever that you use to confound your friends. And you, programmers!?. How many of you are hackers with respect to your competition? Perhaps we have discovered the enemy, and he is us!!

With the above profound philosophical gems in mind, let's relax and get on with this article. As implied previously, knowledge is the key to the successful application of any tool. It is my desire to convey knowledge of the what, where, and how concerning the softkeys that I submit to this publication. Much of this type of detail will be boring to the experienced hackers in our midst. For those that are just beginning, it will be a refreshing breeze.

A great educational program has come to my attention that suits my purpose for this article. This software is called Gradebook III. It is distributed by a small company in Denton, Texas called Schoolhouse Software. At last count it had 11 different kinds of protection on it. I suspect that the programmer is also a hacker as alluded to above. In this particular case, I will expose the protection, define its implementation, and remove it. To alleviate the programmer's anxieties, I suggest that all you hackers and teachers buy Gradebook III; use it to hone your deprotection skills; and then donate it to the nearest school.

Introduction

Gradebook III (GB3), released in 1986, is a program designed to assist teachers in recording, calculating, and reporting student grades and averages. A separate data disk is used to store class and student records. Up to 8 class periods may be stored on a single data disk. Each period will store as many as 40 students with 40 grades each. When student grades are entered for an assignment or test, GB3 updates each students' records, providing for instant reports on student progress.

Gradebook III will accommodate the many different ways that teachers use to calculate student averages. When a new class is created, the teacher can choose 1 to 7 different categories into which grades may be classified. A different weighting factor may be placed on each category so that the overall averages reflect the values selected. These factors may be changed at any time.

Over 15 different reports can be selected for viewing or printing. One is the "Parent Letter". This report prints personalized letters to the parents or guardians of any student regarding that students progress. These include letters of commendation, marginal performance, and failing performance. All of these letters are contained in a BASIC Program that can be customized to a teacher's specific preference by anyone familiar with BASIC Programming. The reports are contained on a data disk called "Parent Letter" provided as the back side of GB3. This software program is a very good tool for teachers.

Gradebook III will run on either a single or dual drive system.

The Examination

One must develop a systematic procedure for determining whether or not a program is protected and what the protection is. The best way to start is to try to make a backup copy using standard copy procedures. I use Locksmith 6.0 (LS6) Fast Disk Backup (FDB) for this purpose because it is fast and it will display, on a track/sector grid, any errors it encounters during the copy process. If no errors are encountered, a dot will be displayed for that sector. An "A" will be displayed for an address field error, and a "D" will be shown for a data field error. In less than 15 seconds, much can be determined about a disk. Applying this procedure to GB3 shows address field errors on all of side 1, the program disk. No errors are indicated on side 2, Parent Letter.

The type of errors found by FDB are caused by formatting alterations. The next step is to identify what these are. One of the best tools available to do this is the LS6 Disk Editor (Nibble Editor). Boot into the main menu of LS6 and press **N** to enter the Disk Editor. Read in a track, say track $0A, for example. Press **R 2 A return** to read in track $0A from drive 2. Press **D**. This is a 16-sector address decode command. The screen display will change showing the buffer address for each sector, the volume number, the track/sector numbers, and error indicators if any. If "?" is displayed, the address field checksum or epilogue is incorrect. If "CS" is shown, the data field checksum is bad. If "*" is displayed, something is wrong with the data field information--usually prologue or epilogue alterations. Applying this technique to GB3 shows errors of "?" and "*" on every sector. This means that the address field checksums and/or epilogues are non-standard and that the data field prologues and/or epilogues are corrupted. The volume number shown is $FE (254). Applying the same procedure to the Parent Letter disk shows everything to be normal (standard format).

Next, press **space** to return to the Disk Editor display. Using cursor movements, find several address fields followed by the corresponding data fields. It can be seen that the lead-in sync field is of mixed values—not the usual

$FF's. This may or not be important. Note what the 4 or 5 values are just prior to the address prologue, especially if they are the same for each sector. I found these values to be (EB) (AB) (FF) (BF) (EB) D5 AA 96 ..... The parentheses ( ) indicate sync-bytes. Examination of the other data in the address and data fields show standard prologues and altered epilogues. The epilogues for both the address and data fields are $FF $FF instead of the normal $DE $AA. The value of the sync bytes before the data prologue is $AB. It still has not been exclusively determined whether or not the address field checksums have been altered.

The next step in the examination is to patch the disk operating system (DOS) to accommodate the altered marks and then see if the protected disk can be read. This can be accomplished within LS6 by pressing **esc** to escape from the disk editor to the main menu. Then press **L** to load Inspector/Watson (I/W), the sector editor. When this is done, press * to patch the resident DOS. It is standard. Type the following:

B935 FF	*was DE, 1st data epilogue byte*
B93F FF	*was AA, 2nd data epilogue byte*
B991 FF	*was DE, 1st address epilogue byte*
B99B FF	*was AA, 2nd address epilogue byte*

To see what is in a memory location, type **SHOW B935 return**, for example. The hexadecimal (hex) value in that location will be displayed. When done, press **esc** to return to the menu. Now, press **I** to call I/W. The idea is to read some meaningful tracks and sectors. Look at the normal catalog track. From I/W press **D** to toggle to drive 2 and then **T 11 return**, **S F return** to set track $11, sector $0F. Press **R** to read the sector into the buffer. The sector is read with no I/O errors. Press **A** to toggle the hex/ASCII display. In the ASCII mode, one should be able to read the name of the first group of files, if a file structure is on the disk at the standard position. Some things can be seen that seem recognizable, but most is written in gibberish. Try another track and sector where there is code. Toggle back to the hex display by pressing **A** then press ⊖**D** to disassemble the code. Either intelligible code or gibberish will be displayed. In our case the sectors are read but the code is corrupt. This is a sure sign that that some non-standard encoding procedure has been used. Usually this is done by tampering with the read and write translate tables located on page $BA of memory, or on track $00, sector $04 of a standard DOS 3.3 disk. The read and write tracks and sectors (RWTS) part of DOS is loaded during boot stage 1; that is, track $00, sectors $00 through $09 are loaded into pages $B6 through $BF of memory, respectively. Boot ROM routines are used to do this, page $C6 in ROM. The read/write translate tables are being read in by standard encoding techniques unless a more severe DOS alteration is being used. It is worth a try to see if track $00, sector $04 can be read using standard read/write encoding tables. From I/W press **T 0 return**, **S 4 return**, **R**. The write translate table extends from byte $29 through $68, and the read translate table extends from byte $96 through $FF. The programmers of GB3 have not moved the tables. They are there and readable! Further examination of these tables shows that $AA and $96 have been interchanged. Specifically, byte $29 in the write translate table has been changed from $96 to $AA. This means that a value of $00 in memory will be written to disk as $AA instead of $96. The corresponding value at byte $AA in the read translate table is $00 instead of the standard value of $AA. This means that if $AA had just been read from the disk, it would be translated as $00 instead of $AA. Therefore, the encoding to and decoding from disk is affected. If standard tables are used to read a disk written with altered tables, gibberish results for any byte related to either $AA or $96.

To test the above findings, set the buffer to page $BA. From I/W press **B BA**, **return**. This shows the read/write translate tables in LS6, which are standard DOS 3.3. Edit byte $29 from $96 to $AA. Press **E 29**, **space AA**, **return**. Also, edit byte $AA. Press **E AA**, **space 00**, **return**. This patches memory. Reset the buffer to $4000. Press **B 40**, **return**. Reset track and sector to track $11, sector $0F. Press **T 11**, **return**, **S F**, **return**. Press **R** to read it in. Toggle to ASCII with a press of **A**. Presto, the first group of files can now be read. Try a few other sectors. Especially, try track $01, sector $09. This is the sector in standard DOS 3.3 that contains the name of the startup file. Press **T 1 return**, **S 9 return**, **R**. Stay in the ASCII display. Beginning at byte $75 "HELLO" is displayed. This is the name of the startup file. It is obvious that the code from disk is now being translated correctly. Referring to a previous comment, the ability to read sectors correctly does prove that the address field checksums have not been altered.

The examination to this point has provided the following information:

1. The lead-in sync fields on GB3 have been changed to be (EB) (AB) (FF) (BF) (EB) prior to the address prologue instead of FF's.
2. The address and data epilogues have been altered on GB3 to be FF FF instead of DE AA.
3. The read/write translate tables of GB3 have been modified to interchange AA and 96.
4. Sectors $00 through $09 of track $00 are written with standard read/write translate tables.
5. Gradebook III is at least partially file based with a startup file of HELLO.
6. The back side, "PARENT LETTER", is in standard format, catalogable from normal DOS 3.3, and apparently unprotected.

Resolve one of the questions by making a standard backup of PARENT LETTER, the back side. Use FDB from LS6, standard COPYA from the DOS 3.3 System Master Disk, or any other whole disk copier. Boot the backup. It will boot and run its HELLO file. Label this backup "PL". It is important to note that this is a data disk for GB3 that is included as an example and that it has files on it needed in the operation of GB3. This means that GB3 must be able to read and write to a standard disk. Therefore, GB3 must have a DOS switching subroutine somewhere, because it also has to read itself, a protected disk.

The Normalization

The effort will now be focused on GB3; however, there is more than one way to proceed. If the protected program is file based, it is usually best to liberate the files and transfer them to a standard format. All DOS modifications are by-passed with the use of standard DOS 3.3 or one of the fast-DOS's. The files can then be examined more conveniently. If there is no additional protection, the program will run.

Another approach is to use SUPER IOB to copy the entire protected original to a standard format and then patch the DOS on disk to circumvent all remaining protection schemes, such as disk checks and DOS modifications. For this approach to succeed on GB3, the first ten sectors ($00-$09) of track $00 will have to be copied using standard read/write translate tables. All other sectors must be read with the altered read/write translate tables and written to disk with standard tables.

The deprotection procedure is different for each protected disk. It involves a trial-and-error process based on what is known. Using LS6, the original has been read by patching the standard DOS in LS6 to recognize the altered epilogues and then patching the read/write tables consistent with the the altered values of GB3. A controller for Super IOB (input/output control block) can be written to do the same thing and that will eventually be done. Everyone may not have Super IOB, but nearly everyone does have COPYA from the DOS 3.3 System Master Disk. COPYA is an excellent copy program, and it can be patched to read most protected disks and re-patched to write in standard format. Some of the more notable adaptations of COPYA are COPYB by Krakowicz and ADVANCED COPYA by the Disk Jockey. Both permit the use of a captured RWTS from a protected program to read while writing with a standard RWTS, in much the same way that is done with a swap controller in Super IOB. All that is needed here is COPYA and COPY.OBJ0 on the same disk. Both of these are on the DOS 3.3 System Master Disk.

TABLE 1 is an executable text file named COPYA.GRD.BK.TXT, included at the end of this article. When executed, this file will load COPYA, patch it to copy GB3, and then run it. Put COPYA.GRD.BK.TXT, COPYA, and COPY.OBJ0 on the same disk and place that disk in the drive of your choice. Do the following:

[a] With standard DOS resident, place the disk containing the COPYA files in the drive of your choice. Place a blank disk in the other drive.

EXEC COPYA.GRD.BK.TXT, Dx *x=drive # of COPYA files*

[b] Follow the prompts. When the duplicate is finished, exit COPYA. Label the duplicate "BU1". Then:

CATALOG, Dy *y=drive # of duplicate*

The catalog of GB3 should now be displayed in its fullness. There are 5 Applesoft (including HELLO), 2 text, and 1 binary files. See TABLE 2.

The disk "BU1" will not boot. Remember the first ten sectors of track $00? However, the files can be transferred to a disk containing standard DOS 3.3. It is known that the startup file is HELLO. To complete this approach, boot into standard DOS 3.3 using the System Master Disk or boot into a fast-DOS, such as Diversi-DOS. Get into BASIC and do the following:

[c] Place a blank disk in drive 2.

NEW
INIT HELLO, D2
DELETE HELLO

[d] Place a disk containing FID in drive 1. (FID is on the DOS 3.3 System Master Disk.)

BRUN FID, D1

[e] Select **1** COPY FILES. Answer the prompts for source and destination slot and drive numbers. In response to the FILENAME prompt, enter "=". In response to DO YOU WANT PROMPTING, enter "N".

[f] Place BU1 in drive 1 and transfer all the files to the initialized blank in drive 2. When completed label the disk in drive 2 "BU2".

Note: An alternate way to copy the files from BU1 to BU2 is to use Copy II Plus Utilities for those that have Copy II Plus.

The disk labeled BU2 has a standard DOS with the files of GB3. If no other protection is present, it should boot and run. Place the disk labeled "PL" in drive 2. Be sure that it is write enabled. Boot BU2 from drive 1. The program loads and the logo appears. In response to the query about number of drives, enter "2". The menu appears next. In response to any of the options that require access to the data disk, the program bombs into the monitor. Therefore, additional protection does exist. This is not surprising, since the program execution switches from the protected program disk to the normal data disk and vice versa. The resident DOS must be switched correspondingly. The next task is to find the switching calls.

The brute force way to search for the DOS switching subroutine is to load the Applesoft programs and look. Place "BU2" in drive 1 and boot it. Press ⊖**C** immediately to force a break. At the break, the file "HELLO" has been loaded but not executed. List it by typing **LIST return**. This file loads graphics (the logo), displays it, pokes an error handler subroutine into page $03 for ONERR, and RUNS HELLO II. There are no switching calls here.

[g] Continuing with the program flow, type:

NEW
LOAD HELLO II
PR#3 *Switch to 80-columns*
LIST

Remember, a ⊖ **S** key press will stop the listing and any other key press will start it again. The listing does not proceed very far before an interesting CALL is seen in line 20. This line instructs the user to "INSERT GRADEBOOK III DISK & PRESS RETURN". A "CALL 47721" is then given prior to accessing GB3. Next, in line 26 a "CALL 47741" is given prior to accessing the data disk. The memory addresses in hex notation are $BA69 and $BA7D, respectively. These are on the same page of memory that contains the read/write translate tables previously discussed, and they are in a crack in normal DOS 3.3 (an unused part of memory) that is located between the write and read translate tables ($BA69-$BA95). To view this area of memory one needs a non-maskable interrupt (NMI) capability. The original cannot be interrupted with ⊖ **C** or ⊖ **reset**. I use the Senior PROM to interrupt programs protected this way. Without explanation of how Senior PROM works, the following must be done. The GB3 program execution is interrupted after DOS has been loaded, and the monitor is entered. A disassembly of the code beginning at $BA69 shows the subroutine. The next move is to save the GB3.RWTS for future reference. It must be moved down in memory out of the way of a re-boot. For example, to move the entire RWTS give the monitor move command, **8600<B600.BFFFM return**. Next, turn off the NMI switch, then warm boot a DOS 3.3 slave disk with a null HELLO, **6 ⊖P, return**. When the Applesoft prompt (]) appears, save the entire RWTS by typing: **BSAVE GB3.RWTS, A$8600, L$A00 return**. A shortened version, usually used with a Swap Controller in Super IOB, extends from $8800 to $8FFF. I have written a short program to make a text file out of the disassembled code of interest. TABLE 3 shows this annotated disassembly listing. For those without an NMI capability, this code can be seen on track $00, sectors $04 and $06 of the original GB3 disk. Boot LS6; defeat the altered epilogues; enter I/W; set the track, sector, and drive; and read the sector into the buffer. Disassemble the code with a ⊖D key press. The procedure was explained previously when discussing alterations in the read/write translate table.

Referring to TABLE 3, the "CALL 47721" ($BA69) installs the DOS modifications. The "CALL 47741" ($BA7D) standardizes DOS. It can be seen that there are jumps to subroutines (JSR) at $BCDF and $BCEC. The memory range of $BCDF through $BCFF is another crack in normal DOS 3.3. The code to change the epilogues is placed here. All of these modifications except one have already been determined. This one, shown at $BA69 and $BA7D, alters the write routine. When altered, sync $AB values are written prior to the data field header. When normalized, this value is sync $FF. This alteration would only be used during the initialization of the original GB3 disk. As noted during the initial examination, sync values of $AB are indeed written before the data field headers of the original GB3. The initialization of data disks from the option to "CREATE DATA DISK" will write the normal sync value of $FF prior to the data field header. This alteration has no significance relative to normalizing and reading GB3. The major benefit of finding this DOS Switching Subroutine, is the validation of what the DOS modifications are.

With the positive identification of the DOS switching calls, the remaining task is to find all occurrences of these calls and remove them. This can continue to be done by the brute force method previously started; that is, load each Applesoft file in turn and search through it. An easier way is to use a program line editor such as the Global Program Line Editor (GPLE) published by Beagle Bros. This utility functions much like a word processor for Applesoft programs. Using the search and replace command, all occurrences of "CALL 47721" and "CALL 47741" can be quickly found and replaced, if desired, with a null string. I have done this for all of the files on GB3 and on PL. The file edits needed to remove these calls are contained in two text files named GRADE.BK.TXT and PARENT.TXT, TABLES 4 and 5, respectively. When executed, these text files successively load, edit, and save each Applesoft file that needs editing. Do the following:

[h] From a normal or fast-DOS, place the disk containing GRADE.BK.TXT in the drive of your choice. Place the BU2 disk in the other drive.

EXEC GRADE.BK.TXT, Dx *where x=drive number of GRADE.BK.TXT disk.*

[i] Follow the prompts and use PARENT.TXT and PL to complete the procedure.

The edited BU2 and PL disks are now completely deprotected. They will boot in much less time than the original if a fast-DOS was used in step (c) above.

Additional Study of the Protection

Steps (a) through (i) above define a way to produce a deprotected backup. By normalizing the format, transferring the files to a disk with a standard DOS 3.3, and editing the files to remove calls to a DOS Switching Subroutine a completely normal backup has been obtained. Often this basic method is successful. It is instructive to take a closer look at the protection that has been by-passed by the approach just defined.

Boot code tracing will always show what is happening, but is is not always easy to follow. This approach is based on the fact that the computer must be able to read track $00, sector $00 of any disk to begin the boot process. This is done by a boot code located in read-only-memory (ROM) on the Disk II Controller Card (page $C6, if the controller card is in slot 6). This code is called Boot Stage 0. Its purpose is to read track $00, sector $00 (Boot Stage 1) into page $08 of memory and then execute it beginning at $801. This boot stage loads the RWTS and Boot Stage 2 into memory and then jumps to Boot Stage 2 to load the rest of DOS into memory. Hence, this is called a boot strap process, because it is like hauling one's self up by one's boot straps. If the program is file-based, a startup file (usually called HELLO) is executed to run the program. Boot code tracing has been described many times in this publication and elsewhere. A few of the steps will be defined here for completeness. Do the following:

[aa] Turn the computer on with no disk in the drive and press ⊖reset.

CALL -151	*enter the monitor*
8600<C600.C6FFM	*Move boot ROM down*
86F8:4C 59 FF	*add jump to monitor*

[ab] Insert the original GB3 disk in drive 1.

8600G	*Execute relocated boot ROM*
C0E8	*Turn the drive off*
1 ⊖P 801LLLLLLL	

The above steps produce a print-out of a disassembly listing of Boot Stage 1. A pertinent part of this listing is shown as TABLE 6. As indicated on the listing, it is standard through $849. At $84A there is an indirect jump through $BBFE. On a normal disk this would be an indirect jump through $8FD to $B700 (Boot Stage 2). One is looking for any kind of change from normal. When found, it must be pursued.

[ac] On with the trace.

86F8:4C 01 88 N 8800<800.8FFM

At this point the boot ROM at $8600 has been altered to point to Boot 1 at $8801. This code will have to be altered so it will run at this location. All direct addressing to $08xx has to be changed to $88xx. There are ten of them. Disassemble the code (8801LL), find the locations that need to change ($08 to $88), and make the changes. After the changes are made, do the following:

884A:4C 59 FF N 8600G	
C0E8	*Turn the drive off*
1 ⊖P BB00LLLLLLL	*List page $BB to printer*

At the end of step (ac), you have a disassembly listing of page $BB. This is code loaded into the DOS primary buffer. It is included as TABLE 7. If garbage is listed, review the changes made in step (ac). If one continues to have trouble, use an utility like I/W in LS6. Defeat the epilogue check and read track $00, sector $00 of the original GB3 into the buffer. This code is Boot Stage 1. Disassemble it and one has TABLE 6. Likewise, read track $00, sector $05 into the buffer and disassemble it. This produces TABLE 7. These codes can be found on a disk using a search utility; however, it is obvious that GB3 is using a standard load of the protected RWTS. This means that sectors $00 through $09 of track $00 load into pages $B6 through $BF of memory, respectively.

A casual glance at this code in TABLE 7 reveals that it is the disk check subroutine. It is completely annotated for the benefit of the beginners. Remember the exit of Boot Stage 1 through $BBFE? A review of this address shows that the execution jumps to $BB00. The following are the highlights of this disk check:

1. The Run flag is set ($BB00). This means that any command will cause an Applesoft program to run.

2. The string "EB D5 AA is sought. The search will go on forever if not found.

3. Pages $08 through $95 are filled with the value $4C. This is part of a later check.

4. The DOS read error check is disabled and the IOB is set up to read track $00, sector $0A. A buffer is established at $4000

5. The primary and secondary DOS buffers are moved to $6B00-$6C55.

6. The sector is read on the first pass, then the secondary buffer is moved to page $5C. $6B02 is stored at $06. The sector is read again and $6B02 is stored at $07. Again, the sector is read with $6B02 stored at $08. The DOS secondary buffer is compared byte-by-byte with the values transferred to page $5C during the first read. If any of the compared valued are unequal a failure branch is taken. If the comparison succeeds, then $06 is compared with $07. If $06 <> $07, the good branch is taken. If $06=$07=$08 ($4C for instance), the disk check is repeated.

7. If the good branch is taken, the read error check is re-established and the DOS buffers are restored to their normal position. Execution then jumps to $B700 (Boot Stage 2).

This is a rather involved disk check and it takes some understanding. First, the run flag is set. Second, the sync $EB just prior to the address prologue is checked. This is an answer to an earlier question. Third, a disk check is made of track $00, sector $0A. An examination of track $00, sector $0A shows that it is a sector of non-standard length, that has the first $55 (86) nibbles as reasonable disk nibbles. The balance of the sector contains mostly illegal nibbles with no recognizable data trailer.

To understand the check of sector $0A, one needs to thoroughly understand how DOS 3.3 processes data to and from disk. "Beneath Apple DOS" published by Quality Software is an excellent reference for all things relative to DOS 3.3. Here, I will describe what happens without trying to explain why. When a sector is read, the secondary buffer, which has been moved to $6C00-$6C55, contains the first $55 (86) data nibbles from the disk, high to low. This corresponds with the first $55 nibbles in sector $0A. Therefore, each time the sector is read, these values will be the same. The balance of the sector is read into the primary buffer now located on page $6B, the entire page. Disk nibbles are placed here, low to high. In the general flow of data processing, these nibbles in the DOS buffers, $155 (342) in all, are translated into $100 (256) data bytes that fill a page of the memory buffer defined in the IOB. All of this means that the check to compare $5C00,Y with $6C00,Y on a byte-by-byte basis should show that corresponding bytes are equal. The next check is to compare $06, $07, and $08. The values stored in these locations are the values of $6B02 for three successive reads of the sector. In general these values will not be equal for an original GB3 because of the illegal disk nibbles. This is an insidious disk check scheme. If one copies the original by ignoring unreadable sectors, the first part of the check will pass but the second part will not. When an unreadable sector is encountered, the copy is left filled with zeros; therefore, the check for the copy will show that $06=$07=$08, and the disk check will go into an endless loop. It is obvious that the only reasonable way to duplicate this sector is with a bit copier.

Another Approach to Normalizing Gradebook III

Under the Normalization Section above, a quite valid method of standardizing GB3 was presented. That method involved the use of tools that almost anyone has. In this section another method of obtaining a backup is given. This method uses Super IOB 1.5 to completely deprotect GB3. A controller for Super IOB 1.5 must be written that performs all of the things previously discussed to normalize the backup. The approach is to normalize the format, standardize the DOS that is on GB3, defeat the disk checks, and edit all of the files that need editing. When all of this is done, the backup obtained will function normally; however, a ⊖reset key press will clear the screen with inverse "C's" and re-boot. Bah! One got by. When ⊖reset is pressed, execution jumps through the reset vector at $3F2-$3F3. Normally, this vector points to $9DBF in memory. The jump to this address reconnects DOS 3.3 and leaves the machine in Applesoft basic. In GB3, this reset vector has been altered to point to some other subroutine. Booting GB3 again and using the NMI of the Senior PROM to break into the monitor allows one to examine the code at $3F2. This shows values of "5B B7 12" which means that the reset handling routine is located at $B75B. Page $B7 is loaded from track $00, sector $01. Use I/W to read this sector, disassemble it, and examine the code at byte $5B. This code must be changed to regain control of the reset function. A patch could be placed at byte $5B, but it is more informative to place it at byte $65. For now enter "4C 59 FF" beginning at byte $65. This will cause execution to jump into the monitor if ⊖reset is pressed. Write this change back to disk and re-boot GB3. Press ⊖reset when the logo is on the screen. When the monitor prompt (*) appears, type **9DBFG return** to enter BASIC. Then type **LIST return**. The BASIC program in memory runs and the logo re-appears. This means that the run flag has been set. Press ⊖reset again to enter the monitor. Type **D6 return** and an $FF is displayed, confirming that the run flag is set. Remember, the run flag was set at the beginning of the disk check subroutine on page $BB (TABLE 7); however, an edit at $84A in boot stage 1 (line 1220 of controller) by-passes this subroutine. This means that the run flag is being set again somewhere else. The code to set the run flag looks like this: A9 FF 85 D6 (Load the accumulator with FF and store it at D6). The backup disk now being used can be scanned to find all occurrences of this code. Using the search function within I/W, this string is found at track $00, sector $05, byte $00, as previously noted, and track $00, sector $0D, byte $30. Edits can now be added to track $00, sectors $01 and $0D to normalize the reset function and clear the run flag. A controller that will do all of the above is presented as TABLE 8. A detailed commentary about each line of this controller follows:

1000 REM Statement identifying the controller.

1002 REM Statement identifying the publisher.

1003-1010 Instructions concerning the backup and related code.

1015 Initializes Super IOB to copy tracks $00 thru $22. The poke causes unreadable sectors to be ignored.

1020 Alters the write and read translate tables as discussed in the Examination section.

1025 Toggles "CD" to read, restores the data pointer to zero, uses altered epilogues as contained in the data statement of line 1200, reads a range of tracks, sets the entry conditions for the sector editor, jumps to the sector editor to execute any appropriate edits contained in the data statements of lines 1210 thru 1290, and reset TK.

1030 Toggles "CD" to write, normalizes epilogues, normalizes write and read translate tables, and writes the range of tracks to disk.

1032 Checks if TRK = 35. If true, jumps to line 1050.

1034 Checks if TRK =1 and LT = 1. If true jumps to line 1055.

1040 Resets TK and ST for start of next range of tracks and then jumps to line 1020.

1050 Checks to see if line 340 is located in memory at normal position and then changes 15 to 09. This is an edit on a running basic program that sets the sector editor for editing 10-sector reads. If all checks, jumps to line 1054

1051-1053 Instruction regarding 10-sector patch if patch fails.

1054 Initializes Super IOB to read first 10 sectors of track $00, alters IOB.OBJ0 to reflect sector $09 as the last sector, and jumps to line 1025. This setup re-reads track $00, sector $00 thru $09 and writes them back using normal write and read translate tables.

1055-1110 Instructions about GRADE.BK.TXT, then executes GRADE.BK.TXT. Note line 1110 that reconnects DOS and repeats line 1100.

1200 Data statement containing the altered epilogues.

1210-1290 Data statements containing 25 sector edits.

Notice that this controller EXECutes the text file GRADE.BK.TXT (TABLE 4) which in turn EXECutes the text file PARENT.TXT (TABLE 5). This feature of the controller illustrates the power of properly conceived EXECutable text files. Also, notice the patch to a running BASIC program in line 1050. This is another unusual feature. This controller demonstrates that almost any disk can be normalized with Super IOB if the original disk is written in a regularly sectored format and if the protection details are known.

The edits performed by the controller normalize the modified DOS on the disk and alter the bit map to free eleven (11) unused sectors on track $02. The EXECution of the text files by the controller, edit the Applesoft files to remove all calls to the DOS switching routine. Install the controller in Super IOB 1.5, accept the format option when given, follow the prompts, and complete the procedure. The backup that results will have all protection stripped away and will be COPYA-able. A fast-DOS can be added, if desired, to speed up disk access by a factor of about two.

Sector edits performed by the controller

Trk	Sct	Byte	From	To	purpose
$ØØ	$ØØ	$4A	6C	4C	*Bypass disk check by*
		$4B	FE	ØØ	*jumping directly to*
		$4C	B3	B7	*Boot Stage 2 (B700)*
$ØØ	$Ø1	$65	A9	4C	*Normalize reset vector*
		$66	ØØ	BF	*to point to $9DBF. Jump*
		$67	8D	9D	*into BASIC.*
$ØØ	$Ø2	$3E	AB	FF	
		$4A	A9	48	
		$4B	AB	68	
		$4C	2Ø	2Ø	*Normalize write routine*
		$4D	8D	B9	
		$4E	B7	B8	
		$9E	FF	DE	*Write data epilogues*
		$A3	FF	AA	
$ØØ	$Ø3	$35	FF	DE	*Read data epilogues*
		$3F	FF	AA	
		$91	FF	DE	*Read address epilogues*
		$9B	FF	AA	
$ØØ	$Ø4	$29	AA	96	*Normalize write/read*
		$AA	ØØ	AA	*translate tables*
$ØØ	$Ø6	$AE	FF	DE	*Write address epilogues*
		$B3	FF	AA	
$ØØ	$ØD	$31	FF	ØØ	*Clear run flag*
$11	$ØØ	$4Ø	ØØ	FF	*Free eleven unused*
		$41	ØØ	EØ	*sectors on track $02*

A Protected Backup

The final way of producing a backup of GB3 will use the bit-copier, LS6. This method will yield a protected backup. One might wonder why a protected backup would be desired. Sometimes it is the only kind that can be obtained. The more important reason is that the protection is preserved for those of us that study such things, permitting the protected backup to be used for further deprotection efforts. Attempting to backup GB3 with all of the popular bit copiers show that they all choke on track $00 to some extent. They have trouble finding the track start. Locksmith 6.0 will successfully copy GB3 if the default procedure is changed. Boot LS6 and do the following:

***** *press asterisk at the menu*
PAT7 EB D5 AA 96 FF FE ? ? AA AA ? ? FF FF
esc *press escape to return to the main menu*
B *and follow the prompts.*

If the backup will not boot, recopy track $00 only. Be sure and make the change to PAT7 each time.

With the protected backup one can use LS6 to disable the epilogue checks, and then use I/W to read track $00, sector $05 into the buffer (TABLE 7). The procedure for doing this was described under the Examination section. By editing this sector to place jumps into the monitor at strategic locations, one can learn much about the protection. For instance, a jump into the monitor can be placed at byte $CC after all the checks have been made. Edit byte $CC by entering 4C 59 FF (jump to the monitor). Write this change back to the disk. Soon after the boot begins, the monitor prompt (*) will appear. Then, all of the areas of memory that have been discussed, can be examined. This procedure illustrates one way a protected backup can be used to understand the protection better. Generally, the boot stage sectors on the disk can be edited to jump into the monitor at any position of the boot. This can take the place of an NMI when one can decipher the boot stage codes.

Conclusions

The purpose of this article was to provide a deprotection treatise about some of the protection techniques in use today. Gradebook III was used as an example because of the plethora of protection it offers. A summary of the protection is given below:

DOS Modifications

- Address and data sync-field alterations
- Address and data epilogue alterations
- Alter the write routine
- Read/write translate table alterations
- Set the run flag twice
- Re-route the reset vector to re-boot

Disk Checks

- Check for $EB prior to address prologue
- Disable DOS read error check
- Read a corrupted sector (Trk $00, Sec $0A) three times and perform sophisticated checks

DOS Switching Routine

- Reads both protected program disk and normal data disk

There are three (3) types of protection and eleven (11) applications of this protection. This qualifies GB3 as one of the heavily protected programs. It is certainly an outstanding example upon which to practice.

It should be clear that a systematic procedure is required to discover what the protection is. A log should be kept about everything that is found and about what is done about it. Copious notes, listings, disassembly listings, and backup attempts are all very helpful, if not required. Date each of them because a deprotection effort usually extends over a period of time. The best way to describe it is to keep notes as if you were going to write an article about the experience. Write to yourself, if no other. I keep files on each program I work on.

A knowledge of the DOS being used is most helpful. In the Apple II world DOS 3.3 and ProDOS are the main operating systems in use. There are still a few DOS 3.2 programs. DOS 3.3 is being phased out leaving ProDOS as the main one in use. There is no alternative to learning what these operating systems do and how they do it, if one wants to become proficient at removing protection. This implies that assembly language must be learned. As demonstrated in this article, it is at the assembly language level that most protection schemes are implemented. This knowledge does not easily come to someone unfamiliar with number systems to different bases or unfamiliar with how computers work. However, assembly language books are available at almost any book store for any microprocessor of interest. One must dive in and start. It is important to add that one does not have to become an assembly language programmer to understand it sufficiently. It is like the person that moves to a foreign country. He can understand a new language long before he can speak and write it.

As important as having a knowledge of the subject, is having the tools of the trade. An expert mechanic cannot do much without his tools. In this business, the basic tools are mentioned in this article. A method of interrupting the execution of a program is almost essential. Senior PROM and Replay II serve this purpose. Wildcard II is another. The bit-copier programs and their utilities are very helpful. All are needed. I use LS6 and Copy II Plus most. The utilities provide sector editors, nibble editors, disk scan capability, memory search capability, disk speed checks, and others. Two other very useful programs are The CIA Files and Bag of Tricks 2. Bag of Tricks 2 has most of the utilities for either DOS 3.3 or ProDOS. Also, Senior PROM has most of the utilities needed on a PROM, providing instant access from an interrupt.

In this article, the use of LS6 has been demonstrated while performing the deprotection tasks. The Senior PROM provides the same capability as LS6 in almost the same format. It does not have the bit-copier option, however.

This bring to a close this rather long article. If it has been educational and has made lucid some of the protection techniques, it has achieved its mark. If not, well, maybe next time it will.

That's all folks.

Table 1: COPYA.GRD.BK.TXT

```
NEW:HOME
LOADCOPYA
177POKE47426,24
197POKE47413,255:POKE47423,255:POKE47505,255:POKE
    47515,255:POKE47657,170:POKE47786,0
225VTAB5:HTAB24:?"^":IFPEEK(713)=1THEN258
248POKE47426,56:POKE47413,222:POKE47423,170:POKE
    47505,222:POKE47515,170:POKE47657,150:POKE
    47786,170
258POKE47426,56:POKE47413,222:POKE47423,170:POKE
    47505,222:POKE47515,170:POKE47657,150:POKE
    47786,170:IFPEEK(713)=1THEN290
RUN
```

Table 2: GB3 CATALOG

```
C1983 DSR C#254
248 FREE
A 005 HELLO
A 098 HELLO II
A 006 DAVID INIT
A 065 CHANGE INFORMATION 3
T 001 PERIODS
T 002 CHECK
A 048 CREATE 3
B 034 GRADE BOOK II
```

Table 3: DOS Switching Subroutine on Pages $BA & $BC
(See track $00, sector $04, byte $69)

```
BA69 A9 AB     LDA #$AB     Change sync byte prior to
BA6B 8D 4B B8  STA $B84B          data header to $AB
BA6E A9 FF     LDA #$FF     Change 1st & 2nd epilogue
BA70 20 DF BC  JSR $BCDF                bytes to $FF
BA73 20 EC BC  JSR $BCEC
BA76 A9 AA     LDA #$AA     Change 1st value in write
BA78 8D 29 BA  STA $BA29         translate table to $AA
BA7B 60        RTS                  Return to caller
BA7C 60        RTS
BA7D A9 FF     LDA #$FF     Normalize sync byte prior to
BA7F 8D 4B B8  STA $B84B          data header to $FF
BA82 A9 DE     LDA #$DE     Normalize 1st epilogue
BA84 20 DF BC  JSR $BCDF                byte to $DE
BA87 A9 AA     LDA #$AA     Normalize 2nd epilogue
BA89 20 EC BC  JSR $BCEC                byte to $AA
BA8C A9 96     LDA #$96     Normalize 1st value in write
BA8E 8D 29 BA  STA $BA29         translate table to $96
BA91 60        RTS                  Return to caller
...                                 (See T$00,S$06,B$DF)
BCDF 8D AE BC  STA $BCAE    1st address epilogue to write
BCE2 8D 9E B8  STA $B89E    1st data epilogue to write
BCE5 8D 35 B9  STA $B935    1st data epilogue to read
BCE8 8D 91 B9  STA $B991    1st address epilogue to read
BCEB 60        RTS                  Return to caller
BCEC 8D B3 BC  STA $BCB3    2nd address epilogue to write
BCEF 8D A3 B8  STA $B8A3    2nd data epilogue to write
BCF2 8D 3F B9  STA $B93F    2nd data epilogue to read
BCF5 8D 9B B9  STA $B99B    2nd address epilogue to read
BCF8 60        RTS                  Return to caller
```

Table 4: GRADE.BK.TXT

```
HIMEM:38400:HOME:NEW
?CHR$(4)"LOADHELLO^II,D";:?(PEEK(43624)=1)+1
20TEXT:HOME:VTAB12:?"INSERT^ GRADEBOOK^ III^ DISK^ &^
    PRESS^RETURN";:GETK$:HOME:VTAB12:RETURN
26?D$;"OPEN^ PERIODS,L5":?D$;"READ^ PERIODS":FORI=0
    TO 8:INPUTPD(I):NEXT:?D$;"CLOSE^PERIODS":ONERR
    GOTO305
29VTAB12:?"NO^ CLASS^ PERIODS^ ARE^ ON^ THIS^ DISK":?:?
    CHR$(7):FORI=1 TO 3333:NEXT:IFPEEK(0)<>76 THEN
    POKE 43624,1:HOME:VTAB10:?"RETURNING^ TO^ THE^
    DIRECTORY...":GOTO11
98IFC$="R" AND PEEK(0)<>76 THEN POKE43624,1:GOTO11
395HOME
SAVEHELLO^II
NEW
LOADDAVID^INIT
180?CHR$(4);"INITHELLO"
184IFPEEK(0)<>76 THEN ?CHR$(7):HOME:VTAB10:?"THE^
    DATA^ DISK^ IS^ READY^ TO^ ENTER^ PERIODS.":?:?:?
    "RETURNING^ TO^ THE^ DIRECTORY...":?D$;"RUNHELLO^
    II,D1"
230?CHR$(4)"RUNHELLO^II"
SAVEDAVID^INIT
NEW
LOADCHANGE^INFORMATION^3
109
20000HOME:IFPEEK(0)<>76 THEN VTAB10:?"RETURNING^TO^
    THE^DIRECTORY...":POKE43624,1:GOTO20065
20010VTAB12:?"INSERT^ GRADEBOOK^ III^ DISK^ &^ PRESS^
    <RET>";:GETC$:?C$
60000HOME:IFPEEK(0)<>76 THEN VTAB10:?"RETURNING^TO^
    THE^DIRECTORY...":POKE43624,1:GOTO60010
60001POKE-16368,0:VTAB12:?"INSERT^ GRADEBOOK^ III^
    DISK^&^PRESS^<RET>";:GETC$:?C$
63556HOME:VTAB10:?"WRONG^DISK":?:?"INSERT^DATA^DISK^
    &^PRESS^RETURN";:GETC$:?C$:GOTO63550
SAVECHANGE^INFORMATION^3
NEW
LOADCREATE^3
180IFPEEK(0)<>76 THEN POKE43624,2:GOTO360
340
3540POKE6,0:HOME:IFPEEK(0)<>76 THEN VTAB10:?"RETURN
    ING^ TO^ THE^ DIRECTORY...":POKE43624,1:?D$;"RUN^
    HELLO^II"
3560VTAB12:?"INSERT^ GRADEBOOK^ III^ DISK^ &^ PRESS^
    <RET>";:GETC$:?C$
3800IFPEEK(222)=8 THENVTAB8:?"THE^DRIVE^DOOR^IS^OPEN^
    OR":?:?"THIS^DISK^IS^UNINITIALIZED!":?:?"PLEASE^
    CORRECT^AND^PRESS^RETURN^";:GETK$:?K$:CALL768:
    POKESK,0:HOME:GOTO140
SAVECREATE^3
NEW
10HOME:VTAB8
20?"TO^EDIT^PARENT^LETTER^INSERT^THE^BACKUP"
25?"OF^";:INVERSE:?"PARENT^LETTER";:?"^IN^D";PEEK
    (43624);".":NORMAL
30?:?:?"INSERT^ THE^ DISK^ CONTAINING^ ";:INVERSE:?
    "PARENT.TXT":NORMAL
35?"^IN^D";(PEEK(43624)=1)+1
40?:?:HTAB8:?"PRESS^THE^C-KEY^TO^CONTINUE."
45?:HTAB8:?"PRESS^<ESCAPE>^TO^EXIT."
50POKE-16368,0:B=PEEK(-16384):POKE-16368,0:IFB=27
    THEN80
55IFB<>27 AND B<>67 THEN50
60?CHR$(1):?CHR$(4)"EXECPARENT.TXT,D";(PEEK(43624)
    =1)+1
70CALL976:GOTO60
80HOME:VTAB8:HTAB12:INVERSE:?"THAT'S^ ALL^ FOLKS":
    NORMAL:TEXT
RUN
```

Table 5: PARENT.TXT

```
NEW
?CHR$(4)"LOADPARENT^LETTER,D";:?(PEEK(43624)=1)+1
380IFK$="R"THENHOME:VTAB10:?"PLACE^ GRADEBOOK^ III^
    DISK^ IN^ DRIVE^ 1.":VTAB12:?"PRESS^ ANY^ KEY^ TO^
    CONTINUE...";:GETK$:?K$:POKE43624,1:?D$"RUN
    HELLO II"
SAVEPARENT LETTER
NEW
10HOME:VTAB8:?"THE^BACKUPS^ARE^UNPROTECTED.^INSTALL^
    A"
20?"FAST-DOS^ON^THE^PROGRAM^DISK^TO^SPEED"
30HTAB12:?"DISK^OPERATIONS."
40VTAB14:HTAB12:INVERSE:?"THAT'S^ALL^FOLKS":NORMAL
    :TEXT
RUN
```

TABLE 6: BOOT STAGE 1
(See track $00, sector $00, byte $01)

```
0801 A5 27     LDA $27
0803 C9 09     CMP #$09
0805 D0 18     BNE $081F
0807 A5 2B     LDA $2B
0809 4A        LSR
080A 4A        LSR
080B 4A        LSR
080C 4A        LSR
080D 09 C0     ORA #$C0
080F 85 3F     STA $3F
0811 A9 5C     LDA #$5C
0813 85 3E     STA $3E
0815 18        CLC
0816 AD FE 08  LDA $08FE
0819 6D FF 08  ADC $08FF
081C 8D FE 08  STA $08FE      Standard Boot 1 Loader from
081F AE FF 08  LDX $08FF                 $801 thru $849
0822 30 15     BMI $0839
0824 BD 4D 08  LDA $084D,X
0827 85 3D     STA $3D
0829 CE FF 08  DEC $08FF
082C AD FE 08  LDA $08FE
082F 85 27     STA $27
0831 CE FE 08  DEC $08FE
0834 A6 2B     LDX $2B
0836 6C 3E 00  JMP ($003E)
0839 EE FE 08  INC $08FE
083C EE FE 08  INC $08FE
083F 20 89 FE  JSR $FE89
0842 20 93 FE  JSR $FE93
0845 20 2F FB  JSR $FB2F
0848 A6 2B     LDX $2B
084A 6C FE BB  JMP ($BBFE)    Indirect jump to $BB00
084D 00        BRK            (Disk Check). Usually jumps thru
                              $8FD to $B700 (Boot 2 Loader)
```

Table 7: Disk check subroutine on page $BB
(See track $00, sector $05, byte $00)
(Disassembled from page $8B loads into page $BB)

```
8B00 A9 FF     LDA #$FF                       Set RUN Flag
8B02 85 D6     STA $D6
8B04 BD 8D C0  LDA $C08D,X
8B07 A9 FF     LDA #$FF
8B09 EA        NOP
8B0A 30 05     BMI $8B11
8B0C A2 B1     LDX #$B1
8B0E 4C F0 BB  JMP $BBF0
8B11 AD FD FF  LDA $FFFD
8B14 A9 00     LDA #$00
8B16 F0 05     BEQ $8B1D
8B18 A2 B2     LDX #$B2
8B1A 4C F0 BB  JMP $BBF0
8B1D BD 8C C0  LDA $C08C,X
8B20 A9 00     LDA #$00
8B22 8D 00 02  STA $0200
8B25 BD 8C C0  LDA $C08C,X
8B28 10 FB     BPL $8B25
8B2A C9 EB     CMP #$EB                       Find $EB
8B2C D0 F7     BNE $8B25
8B2E BD 8C C0  LDA $C08C,X
8B31 10 FB     BPL $8B2E
8B33 C9 D5     CMP #$D5                       Find $D5
8B35 D0 EE     BNE $8B25
8B37 BD 8C C0  LDA $C08C,X
8B3A 10 FB     BPL $8B37
8B3C C9 AA     CMP #$AA                       Find $AA
8B3E D0 E5     BNE $8B25
8B40 A9 4C     LDA #$4C
8B42 A0 00     LDY #$00
8B44 99 00 95  STA $9500,Y
8B47 88        DEY
8B48 D0 FA     BNE $8B44     Fill pages $08-‡$95 with $4C
8B4A CE 46 BB  DEC $BB46
8B4D AD 46 BB  LDA $BB46
8B50 C9 07     CMP #$07
8B52 D0 EC     BNE $8B40
8B54 A9 18     LDA #$18      Disable DOS read error check
8B56 8D 42 B9  STA $B942
8B59 A9 0A     LDA #$0A      Setup IOB to seek sector $0A
8B5B 8D ED B7  STA $B7ED
8B5E D0 05     BNE $8B65
8B60 00        BRK
8B61 02        ???
8B62 01 02     ORA ($02,X)
8B64 06        ???
8B65 A9 00     LDA #$00      Setup IOB to seek track $00
8B67 8D EC B7  STA $B7EC
8B6A 8D F0 B7  STA $B7F0          Buffer address Lo-byte
8B6D A9 40     LDA #$40
8B6F 8D F1 B7  STA $B7F1          Buffer address Hi-byte
8B72 A9 01     LDA #$01
8B74 8D F4 B7  STA $B7F4             Set CMD=1 (Read)
8B77 8D F8 B7  STA $B7F8       Set drive no. found to 1
8B7A 8D EA B7  STA $B7EA       Set drive no. sought to 1
8B7D 8E E9 B7  STX $B7E9      Slot no. ($60 for slot ±6)
8B80 8E F7 B7  STX $B7F7                 Slot no. found
8B83 A0 6C     LDY #$6C
8B85 8C 10 B9  STY $B910
8B88 8C CE B8  STY $B8CE      Change primary DOS buffer to
8B8B 8C D2 B8  STY $B8D2           page $6B and secondary
8B8E 88        DEY                 buffer to $6C00-$6C55
8B8F 8C 21 B9  STY $B921
8B92 20 E7 BB  JSR $BBE7     Jump to RWTS & make 1st read
8B95 A0 55     LDY #$55
8B97 B9 00 6C  LDA $6C00,Y
8B9A 99 00 5C  STA $5C00,Y        Move $6C00-$6C55 to
8B9D 88        DEY           $5C00-5C55 (secondary buffer)
8B9E 10 F7     BPL $8B97
8BA0 AD 02 6B  LDA $6B02
8BA3 85 06     STA $06
8BA5 20 E7 BB  JSR $BBE7     Jump to RWTS & make 2nd read
8BA8 AD 02 6B  LDA $6B02
8BAB 85 07     STA $07
8BAD 20 E7 BB  JSR $BBE7     Jump to RWTS & make 3rd read
8BB0 AD 02 6B  LDA $6B02
8BB3 85 08     STA $08
8BB5 A0 55     LDY #$55
8BB7 B9 00 6C  LDA $6C00,Y       Compare $5C00-$5C55 with
8BBA D9 00 5C  CMP $5C00,Y                  $6C00-$6C55
```

```
8BBD D0 2F    BNE $8BEE      If $6C00,Y <> $5C00,Y,
                                  take failure path
8BBF 88       DEY
8BC0 10 F5    BPL $8BB7
8BC2 A5 06    LDA $06
8BC4 C5 07    CMP $07        If $06 <> $07, take
8BC6 D0 04    BNE $8BCC             good path
8BC8 C5 08    CMP $08        If $06=$07=$08,repeat disk
8BCA F0 17    BEQ $8BE3        check thru $8BE3
8BCC A9 38    LDA #$38       Restore DOS read error
8BCE 8D 42 B9 STA $B942                  checks
8BD1 A0 BC    LDY #$BC
8BD3 8C 10 B9 STY $B910
8BD6 8C CE B8 STY $B8CE      Normalize primary and
8BD9 8C D2 B8 STY $B8D2      secondary buffer locations
8BDC 88       DEY
8BDD 8C 21 B9 STY $B921
8BE0 4C 00 B7 JMP $B700      Exit to Boot 2 Loader
8BE3 4C 54 BB JMP $BB54      Repeat disk check
8BE6 09       ???
8BE7 A9 B7    LDA #$B7       IOB address Hi-byte
8BE9 A0 E8    LDY #$E8       IOB address Lo-byte
8BEB 4C B5 B7 JMP $B7B5      Call RWTS
8BEE A2 B3    LDX #$B3
8BF0 20 58 FC JSR $FC58      Clear screen
8BF3 20 2D FF JSR $FF2D      Print 'ERR'
8BF6 8E 03 04 STX $0403
8BF9 4C 5B B7 JMP $B75B      Sets reset pointer to $B952
8BFC 60       RTS            Reboots via $0000 ($C600)
8BFD 02       ???            -Return to caller
8BFE 00       BRK
8BFF BB       ???
```

Table 8: Gradebook III controller

Controller

```
1000 REM GRADEBOOK III CONTROLLER
1002 REM SCHOOLHOUSE
1003 HOME : VTAB 8: PRINT "THIS^CONTROLLER^IS^SET^TO^
     BACKUP^GRADE-"
1004 PRINT "BOOK^III,^SIDE^1.^PARENT^LETTER,^SIDE^2"
1005 PRINT "IS^UNPROTECTED.^A^BACKUP^OF^SIDE^2^IS"
1006 PRINT "NEEDED^TO^EDIT^CALLS^TO^MODIFIED^DOS."
1007 PRINT : HTAB 8: PRINT "PRESS^ THE^ C-KEY^ TO^
     CONTINUE."
1008 PRINT : HTAB 8: PRINT "PRESS^<ESCAPE>^TO^EXIT."
1009 POKE - 16368,0:B = PEEK ( - 16384): POKE - 16368,0:
     IF B = 27 THEN END
1010 IF B < > 27 AND B < > 67 THEN 1009
1015 PRINT CHR$ (1): HOME :TK = 0:LT = 35:ST = 15:LS =
     15:CD = WR:FAST = 1: POKE 775,96
1020 POKE 47657,170: POKE 47786,0
1025 GOSUB 490: RESTORE : GOSUB 170: GOSUB 610:T1 = TK:TK
     = T1 + 6: GOSUB 310:TK = T1
1030 GOSUB 490: GOSUB 230: POKE 47657,150: POKE
     47786,170: GOSUB 610
1032 IF PEEK (TRK) = 35 THEN 1050
1034 IF PEEK (TRK) = 1 AND LT = 1 THEN 1055
1040 TK = PEEK (TRK) :ST = PEEK (SCT) : GOTO 1020
1050 IF PEEK (3208) = 49 AND PEEK (3209) = 53 THEN POKE
     3208,48: POKE 3209,57: GOTO 1054
1051 HOME : VTAB 8: PRINT "THE^PATCH^FOR^10^SECTORS^HAS^
     FAILED."
1052 PRINT "LINE^#340^MUST^BE^EDITED^TO^CHANGE^15"
1053 PRINT "TO^9.^MAKE^EDIT^THEN^TYPE:^RUN1054<RTN>"
     : END
1054 HOME :TK = 0:LT = 1:ST = 9:LS = 9:CD = WR:MB = 109:
     POKE 900,9: GOTO 1025
1055 HOME : VTAB 8: PRINT "INSERT^THE^DISK^CONTAINING^
     " ;: INVERSE : PRINT "GRADE.BK.TXT" : NORMAL
1060 PRINT : HTAB 6: PRINT "IN^THE^SOURCE^DRIVE,^DRIVE^
     " ;D1; "."
1070 VTAB 14: HTAB 8: INVERSE : PRINT "PRESS^ANY^KEY^TO^
     CONTINUE." : NORMAL
1080 WAIT - 16384,128
1100 PRINT CHR$ (1): PRINT CHR$ (4)
     "EXECGRADE.BK.TXT,D" ;D1
1110 PRINT CHR$ (7): CALL 976: GOTO 1100
1200 DATA 255,255,255,255
1210 DATA 25^CHANGES
1220 DATA 0,0,74,76,0,0,75,0,0,0,76,183
1230 DATA 0,2,62,255,0,2,74,72,0,2,75,104,0,2,76,
     32,0,2,77,185,0,2,78,184,0,2,158,222,0,2,
     163,170
1240 DATA 0,3,53,222,0,3,63,170,0,3,145,222,0,3,
     155,170
1250 DATA 0,4,41,150,0,4,170,170
1260 DATA 0,6,174,222,0,6,179,170
1270 DATA 17,0,64,255,17,0,65,224
1280 DATA 0,1,101,76,0,1,102,191,0,1,103,157
1290 DATA 0,13,49,0
```

Checksums

1000 - $356B	1030 - $F07F	1100 - $1A40
1002 - $724C	1032 - $E874	1110 - $7D74
1003 - $CCE2	1034 - $507D	1200 - $CA3D
1004 - $DB2E	1040 - $201A	1210 - $E526
1005 - $3C15	1050 - $6EDA	1220 - $2ACD
1006 - $864C	1051 - $E194	1230 - $4C21
1007 - $8055	1052 - $A0C1	1240 - $2E06
1008 - $67C2	1053 - $497F	1250 - $05D9
1009 - $9430	1054 - $CDE0	1260 - $E489
1010 - $4069	1055 - $4184	1270 - $9BD7
1015 - $C3EC	1060 - $C837	1280 - $7A41
1020 - $97D9	1070 - $A61F	1290 - $20D7
1025 - $B5EB	1080 - $136E	

Betta B. Goode

IIgs Softkey for...

The Last Ninja GS
Activision

The Last Ninja GS is a karate action game which involves skill with the joystick or keyboard, as well as skill in collecting strength points and weaponry. It can be copied quite easily by any of the standard copy programs for the 3½" disk. But, after you have safely stored your Master Disk and sat down comfortably at your computer, ready to meet the challenges in The Last Ninja, you'll find that you must dig your Master copy out of its place of refuge, because you will not be able to play the program without inserting the Master Disk when prompted. This upset me, more than just a little. I do not like to expose my Master disks to possible damage. So, I set about examining the code on the disk.

Deprotecting my copy involved inspecting the code in the file named "NINJA.SYS16" to discover where the subroutine was that required that I have my Master near by. I found it after only a short search. If your copy is like mine, you should have no problem following these steps.

1. Copy your original onto another 3½" disk.

2. Use your favorite sector editor and search the disk for 22 A8 00 E1 22 00.

It occured only once on mine at Block $CD, beginning at byte 1B0. This instruction tells the program to Jump Long to a subroutine (JSL) at $E1/00A8 and read a block. Easily defeated!

Immediately following this sequence, you should find F1 C4 00 00 90 C0. The culprit is the BCC (branch if carry clear) to $117C, the 90 C0. If the carry is clear, the program will branch to the "Insert Master" routine. There are two instructions that branch to this routine, the second is a BNE (branch if not equal) after an instruction to compare. All we have to do to defeat the second is to "leap-frog" right over it to where the program continues as normal. This occurs at $11D2. Consequently, a BRA (Branch Always) instruction will do the trick.

3. Change the 90 C0 to 80 16. On my disk this occurs at bytes $1BA and $1BB.

NOTE: If the protection lies elsewhere on your copy, you should still be able to modify the above to defeat the protection. On my disk the instruction to continue as normal is a BEQ (Branch if Equal) instruction within 12 bytes after the 90 C0 shown above. You will merely have to change the "16", which follows the BRA (80) instruction, to match the address referenced by the BEQ instruction found on your disk.

4. Write the changes to your disk.

Store your original in a safe place and use the copy to meet the challenges that lay ahead.

Vince Ruggiano

Softkey for...

Millikens Pre-writing Series Branching-Brainstorming-Nutshelling
Milliken

Larry Rando's softkey for Discovery! (COMPUTIST #39) gave me the impetus for this softkey.

1. Boot your DOS 3.3 system disk.

2. Tell DOS to ignore checksum and epilog errors and use COPYA to copy the disks.

POKE 47426,24
RUN COPYA

3. Make the following sector edits to the copy you just made.

Trk	Sct	Byte(s)	From	To
$00	$02	9E	DF	DE
$00	$03	35	DF	DE
$00	$03	91	DF	DE

4. Search the disks for 99 AD and change to 84 9D. I found these bytes at different locations on each disk.

5. Search the disks for 4C 00 C6 and change to EA EA EA.

Make sure to write all edits to your disk! Your disks should now be copyable with normal COPYA (from DOS 3.3). For some reason copies made with Copy II Plus "Disk Copy" did not run. Only COPYA copies work. If anyone knows why this phenomenom is occurring please let me know. Put your originals in a safe place and enjoy!

Softkey for...

Math Shop
Scholastic

1. Copy both sides of Math Shop diskette using any disk copier.

2. Delete the files "PRODOS" and "MATH. SYSTEM" from side 1 of the diskette.

3. Copy a PRODOS system file from another source to side 1 of the diskette. I used the PRODOS file from Copy II Plus.

4. Use a sector editor to search side 1 for 4C 00 C6 and change to EA EA EA. My copy had these bytes on track 10.

5. Search side 1 for 10 ED and change to EA EA. My copy had these bytes on track 3.

Make sure to write all sector edits back to the disk!

Mike Maginnis

First of all, I would like to complement you on an excellent magazine that has proved invaluable more times than I can count. I plan to keep subscribing and hope to see many more issues in the future.

? I have a problem with Michael Horton's Alternate Reality Character Editor. I am an avid adventure gamer and I thought a little help with this one would be nice. Unfortunately, even after changing the '#' signs to '^', the editor still does strange things. Usually, it won't read the character disk, and crashes. On the few occasions it does read, the names are trashed and I can't get to the stats editor menu. Can anyone help me on this one?

? Also, in response to COMPUTIST's question about good BBS's, GBBS and ProGBBS are excellent for the Apple. I also have a public domain program called ABBS (Apple BBS), but I can't seem to get this to work. The drive accesses the boot disk for about a minute before stopping and hanging. Can anyone give me a suggestion for this?

? Next, I have a copy of Copy II+ v8.2. When I go to Bit Copy any disk that's not an original, I get this message about how I should be using an original disk for copying. If I run it with an original, I don't get this message. My question is, how does the program know whether or not the disk is an original? I have tried write protecting copies and using un-write protected originals and it still seems to know whether or not I'm using an original. Can anyone provide some insight as to how they do this?

Ⓟ Finally, I recently purchased a copy of Wasteland by Electronic Arts. This game is excellent and for anyone into futuristic/Sci-fi fantasy RPG's on the computer, I heartily suggest this classic. Unfortunately, but predictably, this program is protected. Bit copies of the program failed miserably, rebooting after several seconds in an endless loop. In COMPUTIST #62 (December) and I saw the softkey for Deathlord, also by Electronic Arts. I am not a great player in the game of software deprotection, but I know enough to be able to at least figure out most protection schemes and use COMPUTIST to help me from there. From Blain Johnson's letter, as near as I can tell with a nibble editor, Wasteland's protection is identical to Deathlord's (1 readable sector on track $0, 10 4&4 sectors, D5 AA + track, sector & checksum bytes headers, no epilogues (that I could find), D5 9C + 342 encoded bytes data prolog, C9 data epilogue). Unforunately, when I tried his method, the drive would format the blank disk and then crash with a PROGRAM TOO LARGE error and stop at line 200. After some experimentation, I ran the program, let it crash, removed lines 10 to 30 from the DEATHLORD.BAS program and ran it again. This time, the program got further, reading for sevral seconds from the Wasteland disk before again crashing with a PROGRAM TOO LARGE error, stopping with line 530. Can anyone offer a hint?

Ⓟ Also, does anyone know where I can obtain a copy of Beneath Apple DOS?

Order the book direct! See page 28, 1st column, 2nd paragraph. .... RDEXed

Playing Tip for...

Wasteland
Electronic Arts

Beware reading any further, too many dead givaways. .... *RDEXed*

- The canteen is a must in the desert, without it you will die of heat exhaustion.
- The cave in Highpool (which I never did find) is useless - don't waste time looking for it
- It's good to have high Lockpicking, Medic and Perception abilities.
- The answers to the riddler's questions in Scott's Bar in Quartz are TOAST, R, and URABUTLN
- The password to the courthouse it MUERTE. A passage leads from the Stagecoach Inn to the Courthouse if you want to save time.
- Most bums want snake squeezins
- The combination to Ugly's safe is Hewey, Dewey and Louie.
- Ugly's hideout password it KAPUT
- The Club Acapulco's password is ACAPULCO
- The launch code is MOTEKIM
- Only carry the basics. It's easy to fill up your inventory with useless items and end up not being able to get vital equipment in a time of dire need (I speak from experience)
- The Black Market's Password is CRETIN
- Fat Freddy's Password is BIRD
- Faran Brygo's Password is KESTREL
- Fat Freddy doesn't like to be told 'NO'
- FARAN BRYGO sends you to Charmaine, but don't get too close to her.
- EINSTEIN is the Holy One
- YES, you did see the great glow
- The sewers are the most difficult part of the game; you meet up with countless killer androids. Come with a full stock of ammunition and powerful weapons.
- Put pieces found in the Sewers into Max until he comes to life.
- The power controller in the northeast most room is missing a converter.
- Go to the room with the power controller and cut the power long enough to get through the energy fields into the power converter room. Do it to get out again (you need to split the party up for this).
- Have a person with high IQ train in the helicopter simulator
- Collect and keep all secpasses (there should be four: 1, 3, A, & 7)
- secpass 7 allows entry to The Underground (Finster's Chambers)
- The password for the elevator is PROTEUS
- connect the Cyber helmet to the Mindlink
- Finster is an android
- The answer to get out of the first arena is 32, the second is 512, the third is 20.
- The next riddle's answer is FINSTER
- You need luck to get out of the showers
- The fifth arena's answer is NOTHING
- ICEBERG is next
- Don't leave here until you have secpass B. Remember everything you do here is permanent.
- Fight your way into the Guardian's Citadel to get the Quasar Key.
- The Guardian's Outer Sanctum has the Nova and Blackstar Keys.
- Use perception to find the Blackstar key
- The Inner Sanctum has extremely powerful weapons and the access code to this area is ROSEBUD.
- The password to Savage Village is REDHAWK and that's who the Junkmaster wants.
- Use the helicopter in the Inner Sanctum to fly to Base Conchise (use the person who you trained to fly the chopper).
- Don't mess with the computer on level 1 of Base Conchise.
- Vax will help you destroy the robot making machines on Level 2 of Base Conchise
- Level 3 of Base Conchise is riddled with land mines
- All the computers on level 3 must be turned on
- In the four rooms of the final Base Conchise, turn the keys in this order: Blackstar, Nova, Pulsar, Quasar. The keys should be in these rooms, respectively: Blackstar - Reactor Core Room, Nova - Robot Maintenance Room, Pulsar - Security Electronics, Quasar - OSHA. The final color sequence is red - nuclear reactor, yellow - security electronics, green - OHSA, and blue - robot maintenance. Waste no time in getting as far away from here as possible.

I.B. Darn

Softkey for...

MECC 1988-89 Copy System
MECC

■ Requirements

- ☐ Any Apple II computer w/64K
- ☐ Super IOB w/Standard.Con (from Starter Kit)
- ☐ Sector editor
- ☐ One blank disk

The MECC Copy System is a fast copy which will utilize the extra memory of the computer. Copies can be produced in about 9 seconds. The Copy System will make copies of any MECC disks for which they give permission. The Copy Sytem will not duplicate itself, but will duplicate all normal Apple disks, plus some protected programs not produced by MECC.

The MECC disk uses two copy protection schemes. The first is by altering the 3rd Address Marker from 96 to FF and the 3rd Data Marker from AD to FE on tracks $01 through $22. The second scheme is a disk signature check on track $02, sector $02. A good bit copier set for Nibble Count will copy the original disk, but that is not the purpose of this article.

To produce a normalized COPYAable disk:

[1] Boot your Starter Kit disk.

[2] Insert a blank disk into the drive and initialize it.

[3] Load Super IOB and merge the Standard Controller.

```
LOAD SUPER IOB
EXEC STANDARD>CON
POKE 47468,0          ignore third addr byte
POKE 47358,0          ignore third data mark byte.
RUN
```

Do *not* re-format the disk when prompted.

[4] Use a Sector Editor to patch track $00 on the disk you just made.

Trk	Sct	Byte(s)	From	To
$00	$07	$8A-8B	A9 FE	A9 AD
	$08	$29-2A	C9 FE	C9 AD
		$97-98	C9 FF	C9 96
	$0B	$B1-B2	A9 FF	A9 96

The disk is now bootable, but will not run because of the nibble count/signature check on track $02, sector $02.

[5] To disable the nibble count/signature check, perform the following sector edits:

Trk	Sct	Byte(s)	From	To
$02	$02	$0C-0D	D0 11	EA EA
$02	$02	$0E-0F	A6 FE	A9 03
$02	$02	$16-17	D0 07	30 FD
$02	$02	$18-19	A6 FE	A6 FF

You should now have another fast disk copier to add to your collection. Thanks to RDEX/COMPUTIST. Perhaps one of you readers will be able to break the program from the disk as a stand alone program.

Softkey for...

Label Utility
MECC

■ Requirements

- ☐ Apple IIe or Apple IIc computer
- ☐ Super IOB w/Standard.Con (from Starter Kit)
- ☐ Sector Editor
- ☐ One blank disk

This MECC Disk is on the reverse side of the MECC Copy System. The disk is only protected by a change in the 3rd address mark byte from 96 to FF and the 3rd data mark byte from AD to FE on tracks $01 to $22.

To produce a normalized COPYAable disk:

[1] Boot your Starter Kit disk.

[2] Insert a blank disk into the drive and initialize it.

[3] Load Super IOB and merge the Standard Controller.

```
LOAD SUPER IOB
EXEC STANDARD.CON
POKE 47468,0          ignore third addr byte
POKE 47358,0          ignore third data mark byte.
RUN
```

Do *not* re-format the disk when prompted.

[4] Use a Sector Editor to patch track $00 on the disk you just made.

Trk	Sct	Byte(s)	From	To
$00	$0C	$86-87	A9 FE	A9 AD
	$0B	$A2-A3	C9 FE	C9 AD
		$1C-1D	C9 FF	C9 96

You should now have a deprotected COPYAable disk. Thanks to RDEX/COMPUTIST.

Rick Davis

Softkey for...

Snoggle
Broderbund

[1] Boot your Starter Kit disk. (Or DOS 3.3 System Master.)

[2] Insert a blank disk in the drive and init a Slave disk.

```
INIT HELLO
DELETE HELLO
```

[3] Enter the monitor.

```
CALL-151
9600<C600.C6FFM      Move Disk II boot code into RAM.
96FA:98              Make it jump to $9801 instead of boot 1 at $0801.
9801:2C E8 C0 4C 59 FF  To turn the drive off and jump to reset.
9600G                Run the changed code.
```

[4] Move boot 1 up to $9800 so we can modify it.

```
9800<800.8FFM
```

```
9805:98        Make the program move our code instead of boot 1.
9843:93                    Make the program exit to our code.
9301:2C E8 C0 4C 59 FF  Turn disk drive off and jump to reset.
9600G                                         Run the code.
```

[5] Move Boot 2 up to where we can modify it.

```
9300<300.3FFM
9343:4C 00 90                          Jump to our code.
9000:A5 3E C9 5D D0 03 4C 5D
9008:02 2C E8 C0 4C 59 FF      This code looks at the current
```

pointer and checks it against $5D. If its not equal to $5D, it returns to Boot 2. If it is equal to $5D, it will stop the disk drive and goto RESET.

```
9600G                                      start the code
```

[6] List the first instruction at $B742.

```
B742L
```

You should see:

```
B742- 20 00 4A  JSR $4A00
```

This is the start of the routine that does the title page.

[7] List the instruction at $B75F.

```
B75FL
```

You should see:

```
B75F- 4C 00 50 JMP$5000
```

This is the entry point of the main program.

[8] Modify a unused section of Boot 1 to stop the drive and goto RESET.

```
9844:2C E8 C0 4C 59 FF
9009:A9 44 8D 60 B7 A9 02 8D 61 B7 4C 00 B7    Change the
```

jump to the main program to jump to $0244. (The location where Boot 1 is moved to).

```
9600G                                      Run the code
```

[9] Move the program code down to an area where it won't be overwritten.

```
2000<5000.AAFFM
```

[10] Put the slave disk in the drive and boot it.

```
C600G
```

[11] Go into the monitor and enter a short move routine to restore the program to its original location, and jump to the start of it.

```
CALL-151
1F00:A0 00 B9 00 7A 99 00 AA
1F08:C8 D0 F7 CE 04 1F CE 07
1F10:1F AD 04 1F C9 1F D0 E8
1F18:4C 00 50
```

[12] Save the entire program to disk.

```
BSAVE SNOGGLE, A$1F00, L$5BFF
```

You now have a BRUN'able file 94 sectors long.

Softkey for...

Alien Rain
Demon Derby

Broderbund

The above procedure will also work for Broderbund's Alien Rain and Demon Derby with the following modifications.

Skip steps 6 and 7, as locations are different.

Modify the second line of step 8 to.

```
9009:A9 44 8D 58 B7 A9 02 8D 59 B7 4C 00 B7
```

Skip step 9.

For Alien Rain, skip Step 11. For Demon Derby only, change step 11 to.

```
1000:4C 00 70
```

In step 12, for Alein Rain type.

```
BSAVE ALIEN RAIN, A$1000, L$67FF
```

Or for Demon Derby, type.

```
BSAVE DEMON DERBY, A$1000, L$67FF
```

Now Alien Rain and Demon Derby are fully deprotected.

Gary Kowalski

Shape Tables? Scott Ernest from COMPUTIST #62 is still using shape tables? I haven't heard of them for years. However, I shouldn't scoff since that's where I started when I was programming my Apple II+ back in 1982. I guess vector shapes are the easiest to deal with from Applesoft.

The "Apple II User's Guide", from McGraw-Hill was my guide, however I don't know if the current edition kept the shape table info or not. But shame on the RDEXed for not knowing that CORE #1 was a special issue on graphics and is still available from Computist as a back issue. CORE #1 explains shape tables and other graphics basics.

If Scott or any other reader would like a shape table editor, they can have "Shape Table Manager" that I wrote years ago for only $10. The program allows you to create vector shapes on-screen and save them. Shape tables can be built from a library of vector shapes you build.

For more advanced graphic technics (assembly language knowledge required) see Nibble Magazine back issues "Graphics Workshop " series 1984-1985 for an excellent discussion of block shapes.

Gary Kowalski
414 W. Russell Ave
Santa Ana, Ca 92707

Bob Colbert

Laser 128 -- The Dream Machine!

I have been cracking software now for about 4 years, but I never owned a computer until about 2 years ago. My first experience at cracking was at my high school, my teacher had a couple of disks he needed copied. I think the first was the Disk Repair Kit, and if I remember correctly, I Bfiled it. After graduation though, I didn't have a computer to work on. Then my brother got an Apple IIe Enhanced, and I thought it was the greatest thing since sliced bread.

I was so impressed with the Apple computer, that I wanted one of my own. When I priced the Apple series, I lost all faith. $1200 for a good system? Then came along the Laser series of computers, and I was quick to pick one up for $400.

Bringing it home,I remember hoping that all of my software would work on it. After testing it out, I found that only about 2 % didn't work! And later on I was able to make these work (maybe in a different letter).

It wasn't until about 8 months ago though that I fell in love with it. Here are some features that you may and may not know of, which I will explain later.

- Non Maskable Interrupt built in.
- Super Hi-resolution (no I don't mean Double Hi-res, I MEAN Super Hi-res!)
- Double the ROM space of the Apple IIe
- Ability to change the border color
- Softswitch to turn off video totally

Non Maskable Interrupt

I actually found this out from an old issue of COMPUTIST. Pressing CTRL-M-reset (or CTRL-RETURN-reset), will break into the monitor in any program, but contrary to past articles, pressing it again will not get you back into the program, the only time that will happen, is if you break into the monitor and the reset vector is set for the program, pressing ⊖**reset** jumps to the beginning of the program (IE. Copy II Plus).

This is not actually a Non Maskable Interrupt. It is a RESET routine that looks at the keyboard. If the RETURN key is pressed then the RESET routine exits to the monitor, otherwise, it performs a normal RESET. .... RDEXed

Super Hi-resolution

I found this one out last night. Enter the monitor.

```
Call-151
C025
C027
C050
C052
C054
C057
C00D:00
C05E:00
```

You should now see Super (Quadruple) Hi-res. Double hi-res screen 1 is on top, double hi-res screen 2 is on bottom. You can also do this without being in the double hi-res mode, where hi-res screen 1 is on top and hi-res screen 2 is on bottom, which is also 'double high-res', but in a different manner.

The softswitches are C025 and C027, C025 makes the characters on the screen a little fuzzy, then C027 puts the new mode on. C024 turns C025 off and C026 turns C027 off.

This is very interesting when used to play a game, you can see both hi-res or double hi-res screens at the same time; just do softswitches C025 and C027, then reboot using 6 ⊖**P**, no game turns these switches off since they do not know they are there.

Also, storing different values at $C023 gives different shades of border (they may be color, but I only have a green screen). And accessing $C02F toggles the video output of the computer on or off!

Double the ROM space!

If you turn the Laser over, you will find a door. Opening it will reveal an EPROM (Erasable Programmable Read Only Memory). This chip is the Lasers ROM code from $C000-$FFFF. Further inspection will reveal that the chip is a 27x256, which is 32K, but the Apple has only 16K of ROM space. Well I figured out how to access this extra space with softswitches. Let me make a map of the chip memory and the computer memory.

chip memory	Laser memory
$800-$FFF	$C800 bank 0
$1000-$1FFF	$D000-$DFFF BANK 1
$2000-$2FFF	$E000-$EFFF
$3000-$3FFF	$F000-$FFFF
$4000-$47FF	$C000-$C7FF
$4800-$4FFF	$C800 bank 1
$5800-$5FFF	$C800 bank 2
$6800-$6FFF	$C800 bank 3
$7800-$7FFF	$C800 bank 4

You can switch banks 1-4 into the $C800 area by doing the following softswitches...

```
CFFF                                 turns off C800 area
C100                                       selects bank 1
CFFF
C200                                       selects bank 2
CFFF
C300                          selects bank 3 (normal bank)
CFFF
C600
C100                                       selects bank 4
CFFF
C600
C200                                       selects bank 5
```

When you switch these in however, you can not list the banks from the monitor, since the monitor turns them off when listing. But, if you have the LISA assembler, you can boot it up and type BRK, which allows you to be in the LISA machine language monitor, and this monitor does not reset the softswitches, so you can view the code.

Try this...

```
CFFF
C100
CADEG
```

You should now get the port configuration program built into the Laser!

Modified ROMs

I have modified my ROM to include utilities for cracking. Here is a list of features that I have added...

- **Auto boot trace:** When T is pressed upon booting, the computer will display the track, sector, and address of the code being loaded using the ROM read routine. Then you can proceed to view the text or hi-res screens, or continue tracing the program!
- **Auxread and Auxwrite:** Using the format xxxx<yyyy.zzzzW, you can move to location x, in the upper 64k, the block y to z, in the lower 64k. Also using the format xxxx<yyyy.zzzzR, you can move to location

x, in lower memory, and the block y to z, in upper memory.

• **Textview:** Using the format 3000T, you can view $200 bytes of text from $3000 on. Pressing T from there on continues the process, so the next time you enter T you would see from $3200 on.

• I also have made it so that pressing "Ctrl-Closed triangle-Reset" jumps to memory location $2000. I don't know why, but I was bored.

All of this is done without removing a single function from the Laser! If you removed the port configuration program, you would have $800 bytes (2K) of totally free space to play with, and you could even swap ROMs. I am considering writing a ROM for nothing else but Cracking. I will have to see what happens.

If anyone would like to contact me, feel free to drop me a letter. I am considering selling modified Laser ROMs, but do not know what the legal implications are. If I did, it would not be an outrageous price, maybe $35-$40.

Chris Willis

A.P.T. for...

Roadwar 2000
SSI

The following are just a few item locations in Roadwar 2000 but they are the most useful to me. A sector editor is required, as well as your save game disk. The items are on track $03, sector $0B.

BYTE	WHAT IT IS
37	Number of vehicles
38-39	Food
3A-3B	Tires
3C-3D	Fuel
3E-3F	Ammo
40-41	Guns
53-53	Medical

There is some disparity in this data and the more extensive data given by Aaron Schoeffler in COMPUTIST #57. Perhaps we are talking about two different versions. Here's a peek at some of Aaron's data:

Name of gang	$23-26
Food	$39-3A
Tires	$3B-3C
Fuel	$3D-3E
Ammo	$3F-40
Guns	$41-42
Medical supplies	$53

.... RDEXed

Scott M. Simon

Super COPYA 1.1

COPYA from Apple has been and still is a great program. This program has stood the test of time, been modified into COPYB, and Advanced CopyA. Anyone who has been kracking for quite a while has COPYA always near his computer. Ready to poke those DOS changes and get that new program into a normal format. Who would have known originally that COPYA would be so well used and easily modified.

For all the newcomers, just look in any issue of COMPUTIST and you will see kracks for half of the programs that use COPYA in some form or fashion. A few article's have been printed showing all those great poke's and machine language changes to modify DOS 3.3's RWTS.

What makes COPYA so great is it's ability to be able to copy most disks in what ever format they are in, like PRODOS, DOS 3.3, PASCAL, SOS, and even some modified DOS's too. So if you are new to the computing scene and don't have COPYA, then stop by your local users group or friend and get it.

This Super COPYA 1.1 program has been on my mind for a number of months now. I guess what inspired the need for it was the constant jumping into BASIC and back and forth to get all the pokes into COPYA so I could copy a disk into normal format. It always seems that I can never remember all those poke's and besides I hate having to look them up. So, I said, why not develop a menu screen which pokes all those things with a press of the key. Even better was the ability to pick and choose the ones I wanted to use. Well, that is what Super COPYA 1.1 is all about.

Designing the menu screen was easy. Getting the codes poked in was done with simple pokes laid out by the menu. The menu gives you the ability to pick and choose which changes to the DOS RWTS you want make. I do not recommend more than 3 changes to the RWTS. While it is possible and I have made 5 changes or more to DOS at a time. Remember, when you poke changes to the RWTS, it is possible to poke so many changes that the DOS RWTS will not be able to tell where a track or sector starts. So go ahead and experiment. But stick to the basic rule of three.

One problem I encountered when first making the program was that DOS constantly unhooked itself when it loaded using the original COPYA method. A simple hello loader was created and, I might add, a technique used for creating a text file with BASIC commands written to a text file and then EXEC'd. For beginners to BASIC programming, this is an easy technique for entering DOS commands from within a running program. A simple way you can load a lot of programs in fast. This loads in COPYA.OBJ0, then writes a text file, EXEC's the written text file and loads in Super COPYA 1.1. (But best yet is the fact that DOS stays hooked with this method.)

Super COPYA has been put through the pace's recently and works great. I am sure if you are a COPYA user you will find this the best addition to your copy programs since Copy II Plus.

1 Boot your DOS 3.3. master disk and enter BASIC.

2 Insert a blank disk in the drive and initialize it.

INIT HELLO

3 Take out your favorite file copy program (Copy II Plus or equivalent) and copy "COPYA" and "COPY.OBJ0" from the DOS 3.3 master to the freshly initialized disk.

4 Boot the new disk and get into BASIC.

NEW
RENAME COPYA,C
LOAD C
70

5 Type the Super COPYA 1.1 listing

6 Save the changed program as C.

SAVE C

7 Clear memory and type the Hello program.

NEW
—enter the hello program here—
SAVE HELLO

Now you are finished and have a great addition to your copy programs.

Super CopyA v1.1

```
10 REM *
20 REM * SUPER COPYA
30 REM * VERSION 1.1
40 REM * BY SCOTT M. SIMON
50 REM * DECEMBER 19, 1988
60 REM *
80 GOTØ490
85 HOME : PRINT : PRINT : PRINT
89 HOME : PRINT : PRINT : PRINT : PRINT : PRINT
18Ø INPUT "--^PRESS^'RETURN'^KEY^TO^BEGIN^COPY^--" ; I$
185 HOME : VTAB 2: HTAB 5: PRINT "SUPER^COPYA^1.1"
195 VTAB 7: HTAB 24: INVERSE : PRINT "READING" : NORMAL
225 VTAB 7: HTAB 24: PRINT "^^^^^^^" : IF PEEK (713) = 1
    THEN 29Ø: REM 7 SPACES
246 VTAB 1Ø: HTAB 24: INVERSE : PRINT "INITIALIZING" :
    NORMAL
251 VTAB 1Ø: HTAB 24: PRINT "^^^^^^^^^^^^" : REM 12 SPACES
256 VTAB 1Ø: HTAB 24: INVERSE : PRINT "WRITING" ; : NORMAL
    : PRINT "^^^" : REM 3 SPACES
265 VTAB 1Ø: HTAB 24: PRINT "^^^^^^^" : REM 7 SPACES
3Ø5 TEXT : HOME : CALL 672: END
49Ø HOME
495 POKE 47426,24
5ØØ VTAB 1: PRINT "SUPER^COPYA^1.1"
5Ø1 VTAB 2: PRINT "MODIFIED^BY^SCOTT^M.^SIMON"
5Ø2 VTAB 3: PRINT "DECEMBER^19,^1988"
5Ø5 VTAB 6: INVERSE : PRINT "CHANGES^FOR^RWTS" : NORMAL
5Ø8 VTAB 7: PRINT "DOS^ERROR^CHECKING^IS^SET^TO^OFF"
5Ø9 VTAB 8: PRINT "MAKE^NO^MORE^THAN^3^CHANGES^TO^RWTS"
51Ø VTAB 9: PRINT "PRESS^LETTER^TO^CHANGE^DOS^RWTS"
515 VTAB 11: PRINT "A.^IGNORE^ADD^HEADER^&^CHECKSUMS"
52Ø VTAB 12: PRINT "B.^ ALLOW^ D4/D5^ IN^ ADDRESS^
    PROLOGUE"
53Ø VTAB 13: PRINT "C.^IGNORE^ADD^&^DATA^EPILOG^ERRORS"
535 VTAB 14: PRINT "D.^IGNORE^1ST^EPILOGUE^BYTE"
54Ø VTAB 15: PRINT "E.^CONTINUE^READING^ON^ERRORS"
545 VTAB 16: PRINT "F.^IGNORE^1ST^BYTE^OF^PROLOGUE"
55Ø VTAB 17: PRINT "G.^IGNORE^DATA^CHECKSUM^BYTES"
57Ø VTAB 19: PRINT "I.^DOS^CHANGES^SELECTED^-^START^
    COPYING"
6ØØ VTAB 22: PRINT "SELECT^LETTER^&^HIT^RETURN" : INPUT
    "^" ;Z$
6Ø5 IF Z$ = "" THEN 6ØØ
61Ø IF Z$ = "A" THEN 7ØØ
615 IF Z$ = "B" THEN 71Ø
625 IF Z$ = "C" THEN 72Ø
63Ø IF Z$ = "D" THEN 73Ø
635 IF Z$ = "E" THEN 74Ø
64Ø IF Z$ = "F" THEN 75Ø
642 IF Z$ = "G" THEN 76Ø
655 IF Z$ = "I" THEN 85
7ØØ POKE 47444,41: POKE 47445,Ø: POKE 47498,Ø: GOTO 6ØØ
71Ø POKE 47444,74: POKE 47445,2Ø1: POKE 47446,1Ø6: POKE
    47447,2Ø8: POKE 47448,239: GOTO 6ØØ
72Ø POKE 47497,24: POKE 47498,96: POKE 47397,24: POKE
    47398,96: GOTO 6ØØ
73Ø POKE 475Ø6,234: POKE 475Ø7,234: GOTO 6ØØ
74Ø POKE 929,24: GOTO 6ØØ
75Ø POKE 47447,ØØ: GOTO 6ØØ
76Ø POKE 47397,24: POKE 47398,96: GOTO 6ØØ
8ØØ END
```

Checksums

1Ø	$BADD	49Ø	$9EFØ	6Ø5	$EAE4
2Ø	$9B13	495	$DBBC	61Ø	$7887
3Ø	$4D3B	5ØØ	$2Ø9C	615	$C371
4Ø	$AD92	5Ø1	$524A	625	$A8A7
5Ø	$C899	5Ø2	$89F6	63Ø	$2ØA5
6Ø	$FF65	5Ø5	$EEØB	635	$1C19
8Ø	$3D1C	5Ø8	$867A	64Ø	$A682
85	$877B	5Ø9	$Ø98B	642	$Ø21B
89	$327D	51Ø	$D194	655	$1D39
18Ø	$7EAØ	515	$673F	7ØØ	$9984
185	$619E	52Ø	$6C86	71Ø	$86FF
195	$FA2E	53Ø	$7Ø95	72Ø	$C9Ø2
225	$EF36	535	$C54A	73Ø	$389B
246	$A2CC	54Ø	$DDB5	74Ø	$34A5
251	$7C67	545	$AF26	75Ø	$EEØ8
256	$8DØ2	55Ø	$5C98	76Ø	$E4D3
265	$B155	57Ø	$1FEC	8ØØ	$B8C1
3Ø5	$31F7	6ØØ	$D5AC		

Super Copy 1.1 Hello

```
Ø TEXT
7Ø PRINT "BLOAD^COPY.OBJØ" : REM A$2CØ
83 VTAB 9: HTAB 2: PRINT "SUPER^COPYA^1.1"
85 VTAB 12: HTAB 2: PRINT "IS^LOADING!"
1ØØ D$ = CHR$ (4)
11Ø PRINT D$; "OPEN^CMDS"
12Ø PRINT D$; "WRITE^CMDS"
14Ø PRINT "RUN^C"
16Ø PRINT D$; "CLOSE^CMDS"
17Ø PRINT CHR$ (4); "EXEC^CMDS"
```

Checksums

Ø	$A951	1ØØ	$BA6C	16Ø	$C92C
7Ø	$53B7	11Ø	$AE27	17Ø	$4447
83	$D459	12Ø	$AF8F		
85	$F972	14Ø	$D8EE		

Tim Rafert

An even BETTER bootable Thexder

After reading Mountain Man's article on "A Better Bootable Thexder" (COMPUTIST #62, pp.9-10), I thought I'd try to get it to work with GS/OS in the same way. It is actually very similar to his bootable Thexder with just a few changes. So, here is yet another way to make a self-booting Thexder!

■ Requirements

☐ GS/OS system disk
☐ Deprotected copy of Thexder by Sierra On Line
☐ Copy II Plus (or something similar)
☐ Blank 3½" disk

This backup holds all 16 levels and uses the new and somewhat faster GS/OS.

1 Make a copy of the deprotected Thexder.

2 Copy the file PRODOS from the GS/OS system disk to the new Thexder disk.

3 Create a subdirectory named SYSTEM on the new Thexder disk.

4 Copy the following three (3) files from the GS/OS disk out of the SYSTEM subdirectory and into the new SYSTEM subdirectory on the new Thexder disk: START.GS.OS, GS.OS, ERROR.MSG

5 Create a new subdirectory named TOOLS in the SYSTEM subdirectory.

6 Copy the file TOOL025 from SYSTEM/TOOLS on the GS/OS disk into the new TOOLS subdirectory on the new Thexder disk.

7 Copy the whole subdirectory called SYSTEM.SETUP on the GS/OS disk to the new Thexder disk under the SYSTEM subdirectory. (Or create SYSTEM.SETUP as a new subdirectory on the new Thexder disk in the SYSTEM subdirectory and then copy the two files, TOOL.SETUP and TS2, from the GS/OS disk into this new subdirectory.)

8 Copy the whole subdirectory called FSTS on the GS/OS disk to the new Thexder disk under the SYSTEM subdirectory. (Or create FSTS as a new subdirectory on the new Thexder disk in the SYSTEM subdirectory and then copy the two files, PRO.FST and CHAR.FST, from the GS/OS disk into this new subdirectory.)

9 Rename the file called THEXDER on your new Thexder disk to THEXDER.SYS16.

10 If your new disk is not already named THEXDER, change it to THEXDER.

11 If you want to, you can delete the two files, FINDER.ROOT and FINDER.DATA, from this new bootable Thexder disk.

The new Thexder disk's directory should be arranged as follows:

```
/THEXDER/ the volume name
   DATA/ DIR of original stuff (all of it's original data files)
   THEXDER.SYS16 original stuff-just changed the name
   PRODOS from GS/OS disk
   SYSTEM/ subdirectory you created
      START.GS.OS
      GS.OS
      ERROR.MSG all 3 from GS/OS disk
      TOOLS/ subdirectory you created
         TOOL025 only tool we need from GS/OS disk
      SYSTEM.SETUP/ whole thing from GS/OS disk
         TOOL.SETUP
         TS2 both from GS/OS disk
      FSTS/ whole thing from GS/OS disk
         PRO.FST
         CHAR.FST both from GS/OS disk
```

Zorro

Softkey for...

Into the Eagle's Nest

Mindscape Software

■ Requirements

☐ Fast Copier (Locksmith Fast Copy, Disk Muncher, etc.)
☐ Sector Editor (Copy II Plus v7-up)
☐ Blank Disk

Into the Eagle's Nest is an interesting game with a similar objective as in Beyond Castle Wolfenstein, with a few new twists added. As in BCW, you must (in advanced levels) plant charges and blow up the castle, but you must also rescue your captured buddies, save precious art treasures, and kill countless Nazi soldiers and drunken officers. The graphics and sounds are quite good, with multicolored double hi-res shapes and a realistic explosion sound if you foolishly shoot at open crates of TNT (in this one, you can hear yourself getting completely fried.) For the benefit of newcomers to the softkeying process, I will provide a lengthy explanation as to how I discovered and defeated this fairly simple protection scheme.

The Protection

The protection on this recent Mindscape release consists of a small nibble count routine. I discovered this little devil by listening carefully when I booted up a copy made by Disk Muncher. After I booted, about 3 seconds passed when I heard the disk head recalibrate, followed by the drive stopping dead in its tracks. This told me that there was indeed a nibble count and that the program somewhere executed the instruction LDA $C088,X to turn off the disk drive and leave my Apple hanging. Mostly by trial and error, I made several copies of the original and scanned them for the string 8D 88 C0, using the scan Hex feature of Copy II Plus. I then changed each occurence of that string to EA EA 60, one at a time, until the change on track $00, sector $08, finally did it the second attempt. Mind you, I didn't eliminate the entire nibble count; I just bypassed it by replacing the 'turn off disk drive' command string with 'do two NOP's and Return to wherever you came from.' This successfully returned control to the main program, and now it works like a charm.

Cookbook Style

1 Make a copy of the original disk using a fast copier which will ignore errors.

2 Change the following bytes using the sector editor:

Trk	Sct	Byte(s)	From	To
$00	$08	$63-65	BD 88 C0	EA EA 60

Enjoy.

Playing Tip for...

Maniac Mansion

Lucasfilm

● To use the radio, find the tube and place it in the socket. Can't find the tube? Well, in what antiquated (but still useful) items are they used?
● There is a significant key to the game hidden behind the hamster.
● If you give Weird Ed his package, you'll be his friend (as long as you do what he asks of you).
● To thoroughly investigate Nurse Edna's room, do the following: Station one kid at the phone, another right outside of Edna's room. Dial the number found on the 3rd floor behind the mummy (you did find the faucet handle, right?), and then enter her room. Be quick to look around, however, because you have only several minutes before Edna hangs up and gets smart to your plans.
● To get the rusty key: Find the record up in the green tentacle's room, get it, then go downstairs until you get to the room with the T.V. If you don't have the blank cassette tape then, get it now. Insert the cassette into the cassette recorder, push record, place the record on the Victrola, play it. You should hear a momentary high-pitched sound. Turn off the recorder, pick up the cassette. Go down to the room preceding the den, and after you place the tape into the player, turn on the player, watch what happens, and pick up the key to freedom.
● The gleaming key can open two special locks.
● What would you use to suck up a puddle?
● If you are in a destructive mood, Use the yellow key in the Edsel!
● Whatever you do, don't touch that red button in the deeps!!
● O.K. You've gone down into the depths after turning on the valve. You found what was there and you immediately go to turn off the valve. WAIT A MINUTE!! Don't turn that valve off yet! You'll drown your kid!

To Marc Batchelor: Concerning your Autoduel Car Editor, for some reason, it doesn't work on my copy. I typed the whole thing in myself, paying close attention to my typing (the checksums matched perfectly). When I RAN it, I edited a car, but when I returned to playing the game, I discovered that I could not leave New York by any means. Right before the road comes up on the screen, I hear a strange 'click,' followed by the program hanging. Help with this will be appreciated.

To Jan Recourt: I, too, wish I could put the Ultimas on a 3½" disk. But it seems quite complex. Somehow, you would need to make major modifications on the program so that it would access only the slot which has the 3½" drive. In addition, one would have to go through the program and find some space to put in a PREFIX command, since you'll probably be using ProDOS, and since some of the filenames on the separate disks are identical (either that or go through and change all of the filenames in the catalog, and then change the program so it will load those files). Geez, I wish more companies would just put their programs in separate FILES. That would make it MUCH simpler to transfer to a hard- or micro-disk.

Gary Rohr

Softkey for...

Broadsides v2.0

Strategic Simulations Inc.

■ Requirements

☐ Original Broadsides v2.0 diskette
☐ An intialized diskette with the hello program deleted that contains SUPER IOB, swap controller, standard controller, and DISKEDIT
☐ One blank diskette

About a year ago I attempted to de-protect this game, without success. After reading Craig Meekin's softkey in the November 1988 issue of Computist, I tried again -- it didn't work.

I used the nibble editor on Copy II Plus to check the prologues and epilogues of my disk versus that of Craig's and found they weren't even close to the same - so I decided to see if I could crack mine this time around.

Armed with a Wildcard, so I could break into the monitor at will, I started by capturing the RWTS. Using SUPER IOB, it was a simple matter to convert the protected diskette into one which was readable. I then searched the diskette for the standard BD 8C C0 sequence, indicating that a byte was being read from the diskette and altered the checks for prologues and epilogues to the standard format.

I tried booting the diskette, but it still wouldn't work. I decided to perform some boot code tracing to find out where it was hanging - it worked fine all the way up until it attempts to use the RWTS. I went back to examine the code being used and found it was slightly different from that which is normally used - I tried replacing it, but that didn't work either. After comparing the BROADSIDES RWTS with a normal one, and seeing it wasn't even close, I decided to try replacing the protected RWTS with a normal one. This time the boot went farther before it hung. I checked the RWTS parms to see what was being read and found a $20 error code (volume mismatch). I scanned the diskette for all locations which access $B7EB (the RWTS parm volume number) and found three locations where it is set to $FF. I tried using SUPER IOB to copy the diskette to a new one initialized with volume #255 - the boot didn't even get as far as it did before. I checked the RWTS parms error code, and found it was looking for volume #254. I decided to go back to the other diskette and change the $FF bytes to $FE bytes. I tried booting the de-protected version once again - this time it worked!!

1 Boot the Broadsides v2.0 diskette and interrupt as soon as the DOS prompt appears.

[2] Enter the monitor.

CALL -151

[3] Move the RWTS to a safe location.

1900<B800.BFFFM

[4] Boot the SUPER IOB diskette with the hello program deleted.

C600G

[5] Save the RWTS to disk.

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

[6] Load Super IOB and merge the swap controller.

LOAD SUPER IOB
EXEC SWAP.CON

[7] Change the following lines.

1010 TK=1:ST=0:LT=35:CD=WR
10010 IF PEEK(6400)<>56 THEN PRINT CHR$(4)"BLOAD RWTS.BROADSIDES,A$1900"

[8] Run the program. Answer yes to the format question and use a volume number of 254.

RUN

[9] When done, change the following lines and run again.

1010 TK=0:ST=1:LT=1:CD=WR
1060 GOSUB 490:TK=T1:ST=1:GOSUB 360
RUN

[10] Then, reload Super IOB and merge the standard controller. Modify the controller to copy the RWTS (trk $00, sct $02-09). Use the the SUPER IOB disk as the source disk.

LOAD SUPER IOB
EXEC STANDARD.CON
1010 TK=0:ST=2:LT=1:CD=WR
1030 GOUB 430:GOSUB 100:ST=ST+1: IF ST<10 THEN 1030
1060 GOSUB 490:TK=T1:ST=8
RUN

[11] Clear Applesoft pointers and run Diskedit.

FP
RUN DISKEDIT

[12] Read track $00, sector $00 from the original Broadsides disk, make the following sector edits, and write it out to the new "de-protected" disk.

Trk	Sct	Byte(s)	From	To
$ØØ	$ØØ	$4A	AA	D5
		$53	D5	AA
		$5D	AB	96
		$88	AA	D5
		$91	D5	AA
		$9B	EB	AD

[13] Perform the following sector edits on the new "de-protected" diskette.

Trk	Sct	Byte(s)	From	To
$ØØ	$Ø3	$1A	AA	D5
		$23	D5	AA
		$2D	AB	96
		$57	AA	D5
		$6Ø	D5	AA
		$6A	EB	AD
$ØØ	$Ø6	$CB	FF	FE
$ØØ	$Ø7	$DØ-D1	49 FF	A9 FE
		$EB	FF	FE
$Ø2	$Ø3	$68-6A	49 FF FF	A9 FE FE

Enjoy!

Douglas Bancroft

Playing Tip for...

Marble Madness
Electronic Arts

While playing Marble Madness, I discovered an easier way to finish Level 3. Normally, you would probably go down the track, taking a right and then a sharp left. Instead, near the end, take the narrow path that leads onto a belt of some sort. Ride the belt down toward the edge of the screen and hop off of the belt onto another path. From that path take a right and go to the striped finish area. Note: I have only done this with a joystick. Warning: If you try to get onto the belt and you hit a pile, you will be flung off. Also, if you stay on the belt too long, you will lose your marble.

Matthew D. Bancroft

A.P.T. for...

Marble Madness
Electronic Arts

Micheal A. Horton's article in Computist #50 on page 7 was very helpful but unfortunatly it had some problems they were:

- If you die, the marble will keep restarting where you died
- He showed you where the entrance to the water-maze was, but didn't tell you how to get over to the maze entrance.
- If you were to put in the Unlimited Time APT, you couldn't get into the 'Secret Maze'.

I have discovered ways to 'Fix' these problems:

- For the 'you're history' problem, search for A9 ØØ 9D 77 6Ø A9 Ø7, and change it to 6Ø EA 9D 77 6Ø A9 Ø7.
- For the problem of not being able to get to the entrance to the water-maze; simply get into the first level, go down to the first curve from the bottom and "Super-charge" your marble. Go back up to the peak of the second curve from the bottom and get off of it onto the side with the entrance to the water-maze.
- For the unlimited time patch, search for BD 77 6Ø FØ Ø8 A9 Ø2 and change it to BD 77 6Ø DØ Ø8 A9 Ø2. This will allow you to manuever in all mazes except ice. In that situation, when the counter hits zero, you are stuck forever. Can someone improve this fix?

A hint: It seems that you do not want to run across the thing that looks like a bridge in the first water maze level. You don't even want to hit it. It is a dam and if you hit it it will break. The current of the river will be going so fast that you will be washed down stream and die.

Bob Igo

APT Scanner

This program can be most effectively used in conjunction with the DeathSword article I sent outlining APT-finding Techniques (COMPUTIST #65).

Type in the BASIC program, save it (**SAVE APT SCANNER**) and then enter the assembly language program at $6000 and save it (**BSAVE SCANNER, A$6000, L$4E**).

I'll see you at the bottom.

APT SCANNER

```
80 REM --APT SCANNER--
85 TEXT
95 IF PEEK (24576) < > 169 THEN PRINT CHR$ (4) "BLOAD^
    SCANNER"
100 IF PEEK (768) < > 169 THEN PRINT CHR$ (4) "BLOADDOS^
    UTILITY"
110 HOME :TR = 784:SE = 789:RW = 794:R = 1:W = 2: POKE
    799,0: POKE 804,32:A = 8192:P = 768: POKE 774,96:
    POKE 779,1
112 DIM T$(34),S$(15)
115 FOR T=0 TO 34: READ T$(T) : NEXT : FOR S=0 TO 15: READ
    S$(S): NEXT
120 INPUT "STARTING^TRACK^(0-34):^" ;ST
130 INPUT "END^TRACK^(0-34):^" ;ET
132 PRINT "SLOT^FOR^OUTPUT:^" ; : GET SL$:SL = VAL (SL$):
    PRINT SL
135 HOME : PRINT "I^WILL^SCAN^FOR^LDA,^LDX,^AND^LDY."
    : PRINT "WHAT^VALUE^SHOULD^I^SEARCH^FOR^AFTER" :
    PRINT "EACH^ OF^ THESE?" : PRINT : INPUT "VALUE^
    (0-255):^" ;VA
136 POKE 24912,VA: HOME : POKE 34,1
137 POKE RW,R: PR# SL
138 VTAB 3: PRINT "TRACK" , "SECTOR" , "BYTE"
139 VT = 4
140 FOR T = ST TO ET
150 FOR S = 0 TO 15
160 POKE TR,T: POKE SE,S
161 PR# 0
165 VTAB 1: HTAB 1: PRINT "TRACK^$" T$(T), "SECTOR^$"
    S$(S)
166 VTAB VT
170 CALL P
175 PR# SL
180 CALL 24576
185 PR# 0
195 VT = PEEK (37) + 1
200 NEXT S
210 NEXT T
300 DATA 00,01,02,03,04,05,06,07,08,09,0A,0B,0C,0
    D,0E,0F,10,11,12,13,14,15,16,17,18,19,1A,1B,1C
    ,1D,1E,1F,20,21,22
310 DATA 00,01,02,03,04,05,06,07,08,09,0A,0B,0C,0
    D,0E,0F
```

Checksums

80 - $D244	135 - $390A	166 - $28D0
85 - $18D6	136 - $1C8A	170 - $D7DC
95 - $CE87	137 - $4096	175 - $924C
100 - $BD54	138 - $6EF5	180 - $B8A1
110 - $0BCC	139 - $2842	185 - $5592
112 - $9664	140 - $E63D	195 - $CBD4
115 - $681B	150 - $BFF9	200 - $5557
120 - $2A30	160 - $2D2C	210 - $A543
130 - $DE81	161 - $50E3	300 - $9450
132 - $2DEF	165 - $06DD	310 - $4D2A

SCANNER

```
6000: A9 A9 8D 00 61 A9 A2 8D      $8439
6008: 01 61 A9 A0 8D 02 61 A0      $6F86
6010: 00 A2 00 B9 00 20 DD 00      $A68E
6018: 61 F0 09 E8 E0 03 D0 F3      $7E10
6020: C8 D0 EE 60 B9 01 20 CD      $E1A7
6028: 50 61 D0 EF A9 00 85 24      $1BB4
6030: AD 10 03 20 DA FD A9 10      $E487
6038: 85 24 AD 15 03 20 DA FD      $2ADE
6040: A9 20 85 24 98 20 DA FD      $22C0
6048: 20 8E FD 4C 1B 60            $F92F
```

SCANNER disassembly

```
A9 A9     LDA #$A9
8D 00 61  STA $6100
A9 A2     LDA #$A2
8D 01 61  STA $6101
A9 A0     LDA #$A0
8D 02 61  STA $6102
A0 00     LDY #$00
A2 00     LDX #$00
B9 00 20  LDA $2000,Y
DD 00 61  CMP $6100,X
F0 09     BEQ $6024
E8        INX
E0 03     CPX #$03
D0 F3     BNE $6013
C8        INY
D0 EE     BNE $6011
60        RTS
B9 01 20  LDA $2001,Y
CD 50 61  CMP $6150
D0 EF     BNE $601B
A0 00     LDA #$00
85 24     STA $24
AD 10 03  LDA $0310
20 DA FD  JSR $FDDA
A9 10     LDA #$10
85 24     STA $24
AD 15 03  LDA $0315
20 DA FD  JSR $FDDA
A9 20     LDA #$20
85 24     STA $24
98        TYA
20 DA FD  JSR $FDDA
20 8E FD  JSR $FD8E
4C 1B 60  JMP $601B
```

Okay. Now that we've gotten the tedium out of the way (unless you really really love to type), here are some instructions:

Run the BASIC program (**RUN APT SCANNER**), then insert the disk to be scanned in drive 1. Enter 0 as the starting track and 34 as the end track. To make things easier, I decided to use my printer as output, slot 1 in my case. You next enter the number of lives the game gives you. The next bit is easy....we wait.

On Saracen, the technique I outlined in my DeathSword article must be modified to work. Instead of DECrementing the location directly, the program loads the current number of lives into the Accumulator and then uses the SBC #$01 command to subtract one from the Accumulator. The new number is then stored back into memory where the old one was. I don't know why they do it this way, maybe just to annoy me.

Buck Rogers was pretty straightforward. You scan for the number of lives with APT SCANNER and then use the technique I outlined in my DeathSword article. Actually, instead of doing that exactly like I described, the expert (or advanced beginner) can look at the disassembly of the locations which the scanner found to see where the theoretical number of lives is stored in all cases. Then, after writing them down, play the game and get "killed" once. The number of lives should now be one less than what was originally searched for. Reset into the monitor (or somehow break the program's execution) and check all the locations where you thought the number of lives could have been stored. If you find one which holds the current number of lives, then you now know which edit to make. Then, run Editor Creator and make a quick editor program. Next, send it in to Computist!

A.P.T. for...

Saracen

Datasoft

For unlimited lives.

Trk	Sct	Byte(s)	From	To
$2Ø	$ØB	$A4-A5	E9 Ø1	EA EA
$2Ø	$ØB	$DD-DE	E9 Ø1	EA EA

A.P.T. for...

Buck Rogers

?

For unlimited ships.

Trk	Sct	Byte(s)	From	To
$Ø2	$ØA	$98-99	C6 AE	EA EA
$Ø2	$ØB	$Ø3-Ø4	C6 AE	EA EA
$ØA	$ØA	$3F-4Ø	C6 AE	EA EA
$ØA	$ØC	$A8-A9	C6 AE	EA EA

Addendum to the A.P.T. for...

Kid Niki Radical Ninja

Data East

The Kid Niki edits in Computist #60 were not where they were stated to be on my copy. On mine, they appeared on track $07, sector $0F. Other than that, they are the same bytes in the same positions on the track.

Phil Goetz

A.P.T. for...

Montezuma's Revenge

Parker Brothers

■ **Requirements**

☐ Cracked Disk with program on DOS 3.3 as normal Bfile

Presuming you already have Montezuma's Revenge as a DOS file, here's some good cheats:

Location	Contents
E0	# of men
E1	# of items (up to 5)
E2-E6	Items

Item #	Item
2B	Gold (useless?)
2F	Snakeproofing
33	Sword
37	Torch
3B	Key A
3F	Key B
43	White key
47	Chest (useless?)

Keys A and B are different colors, but I know not which because I have a B&W monitor.

To give yourself an item, change the reset vector to point to the monitor (see program below), and press reset in the game. Change the values, then type

C050	*graphics*
C057	*hi-res*
6026G	*resume game*

The game will act strangely until you move into a new room.

Here's a program which will let you redirect the reset vector, have an infinite number of men, be fireproof, fallproof, and/or monsterproof, and/or not need keys for doors:

Montezuma's Revenge - Reset fix and APT

```
2 PRINT CHR$ (4) "BLOADMONTEZUMA'S^REVENGE"
5 FOR N = 1 TO 6: READ V$(N,Ø),V$(N,1),D$(N),L(N),N(N):
   NEXT
1Ø TEXT : HOME :A$ = "MONTEZUMA'S^REVENGE" : GOSUB 9ØØ:
   PRINT : PRINT
3Ø FOR N = 1 TO 6:D(N) = PEEK (L(N)) = N(N): PRINT : PRINT
   N ".^" ;: IF D(N) THEN INVERSE
4Ø PRINT V$(N,Ø);: NORMAL : PRINT "/" ;: IF NOT D(N) THEN
   INVERSE
5Ø PRINT V$(N,1);: NORMAL : PRINT "^" D$(N): NEXT
6Ø PRINT : PRINT "7.^RUN^MONTEZUMA'S^REVENGE" : PRINT
   : PRINT "8.^EXIT^TO^BASIC" : PRINT : PRINT
8Ø PRINT : PRINT "ENTER^YOUR^CHOICE:^" ;: GET A$: PRINT
   A$
85 N = VAL (A$): IF NOT N THEN 1Ø
9Ø ON N GOTO 1ØØ,2ØØ,3ØØ,4ØØ,5ØØ,6ØØ,7ØØ,8ØØ: GOTO 1Ø
1ØØ IF D(1) THEN POKE 51Ø5,89: POKE 511Ø,255: GOTO 1Ø
11Ø POKE 51Ø5,4: POKE 511Ø,96: GOTO 1Ø
2ØØ IF D(2) THEN POKE 27917,169: POKE 27918,Ø: GOTO 1Ø
21Ø POKE 27917,198: POKE 27918,224: GOTO 1Ø
3ØØ IF D(3) THEN POKE 27419,24: POKE 2742Ø,96: GOTO 1Ø
31Ø POKE 27419,169: POKE 2742Ø,15: GOTO 1Ø
4ØØ IF D(4) THEN POKE 27653,Ø: POKE 27654,76: POKE
   27655,47: POKE 27656,1Ø5: GOTO 1Ø
41Ø POKE 27653,26: POKE 27654,141: POKE 27655,4Ø: POKE
   27656,21: GOTO 1Ø
5ØØ IF D(5) THEN POKE 26779,24: POKE 2678Ø,96: GOTO 1Ø
51Ø POKE 26779,2Ø1: POKE 2678Ø,7: GOTO 1Ø
6ØØ IF D(6) THEN POKE 31938,27: GOTO 1Ø
61Ø POKE 31938,176: GOTO 1Ø
7ØØ CALL 51Ø4
8ØØ END
9ØØ PRINT TAB( 21 - INT ( LEN (A$) / 2 + .5))A$: RETURN
1ØØØ DATA "RESTART" , "MONITOR" , "RESET" ,51Ø5,4,
   "FINITE" , "INFINTE" , "MEN" ,27917,198, "NORMAL"
   , "FIREPROOF" , "MEN" ,27419,169, "DIE" , "LIVE" ,
   "AFTER^A^FALL" ,27653,26
11ØØ DATA "FEAR" , "DON'T^FEAR" , "MONSTERS^&^CHAINS"
   ,26779,2Ø1, "NEED" , "DON'T^ NEED" , "KEYS"
   ,31938,176
```

Checksums

2 - $E388	9Ø - $Ø254	5ØØ - $8BAA
5 - $DØ6C	1ØØ - $A557	51Ø - $5E11
1Ø - $AC72	11Ø - $Ø9CB	6ØØ - $E37E
3Ø - $FBEA	2ØØ - $Ø124	61Ø - $2F8F
4Ø - $F445	21Ø - $AFDE	7ØØ - $7146
5Ø - $786D	3ØØ - $22A9	8ØØ - $D12A
6Ø - $1D42	31Ø - $F2E6	9ØØ - $A3B8
8Ø - $3C54	4ØØ - $B9E8	1ØØØ - $3B4F
85 - $62FØ	41Ø - $F4DØ	11ØØ - $491Ø

Mike Maginnis

Softkey for...

Times of Lore

Origin Systems

Rule #1 of the Software Deprotector's Guide: *a company will usually use the same protection on several different programs.* Times of Lore, the newest release from Origin Systems has exactly the same protection as Ultima V, another of Origin's epic adventures. Times of Lore is the arcade answer to Ultima V: the game can be played entirely by joystick. I consider Times of Lore the link between Gauntlet and the Ultima series. To copy Times of Lore (side 1; side 2 is not protected):

[1] Type in the machine language program an save it on your Super IOB disk.

```
CALL -151
1900: A2 00 BD B8 BF 18 69 11          $E96C
1908: 9D B8 BF E8 E0 10 D0 F2          $784F
1910: 60 A2 00 BD B8 BF 38 E9          $353D
1918: 11 9D B8 BF E8 E0 10 D0          $B7FB
1920: F2 60                            $E0D7
BSAVE TIMES.SC, A$1900, L$022
```

(or use ULTIMA 5.SC - used to copy Ultima V in COMPUTIST #61, it's the same thing)

[2] Merge the controller into Super IOB and copy side one of the game disk.

[3] Copy the back side with COPYA or Super IOB's standard controller.

Controller

```
1ØØØ REM TIMES OF LORE CONTROLLER
1Ø1Ø TK = Ø:ST = Ø:LT = 35:CD = WR
1Ø2Ø POKE 475Ø7,Ø: POKE 47517,Ø
1Ø3Ø UB = Ø:T1 = TK: GOSUB 49Ø
1Ø4Ø IF UB = 1 THEN 1Ø6Ø
1Ø5Ø IF TK = > 3 THEN CALL 64ØØ:UB = 1
1Ø6Ø GOSUB 43Ø: GOSUB 1ØØ:ST = ST + 1: IF ST < DOS THEN
   1Ø6Ø
1Ø7Ø IF BF THEN 1Ø9Ø
1Ø8Ø ST = Ø:TK = TK + 1: IF TK < LT THEN 1Ø4Ø
1Ø9Ø UB = Ø: GOSUB 49Ø:TK = T1:ST = Ø
11ØØ IF UB = 1 THEN 112Ø
111Ø CALL 6417:UB = 1
112Ø GOSUB 43Ø: GOSUB 1ØØ:ST = ST + 1: IF ST < DOS THEN
   112Ø
113Ø ST = Ø:TK = TK + 1: IF BF = Ø AND TK < LT THEN 11ØØ
114Ø IF TK < LT THEN 1Ø3Ø
115Ø POKE 475Ø7,174: POKE 47517,164: HOME : PRINT "COPY^
   DONE." : END
1ØØ1Ø PRINT CHR$ (4); "BLOAD^TIMES.SC,A$19ØØ"
```

Checksums

1ØØØ - $356B	1Ø6Ø - $D138	112Ø - $4681
1Ø1Ø - $3266	1Ø7Ø - $Ø586	113Ø - $2Ø68
1Ø2Ø - $5917	1Ø8Ø - $752E	114Ø - $D39Ø
1Ø3Ø - $51D2	1Ø9Ø - $DØØD	115Ø - $16C5
1Ø4Ø - $8CFB	11ØØ - $68D6	1ØØ1Ø - $9F86
1Ø5Ø - $DE3C	111Ø - $6483	

The credit on this one really goes to Captain Dan for his softkey - all I did was use it on another program.

Jim S. Hart

Softkey for...

Microzine #25

Schoolastic

■ **Requirements**

☐ Microzine #25 original disk
☐ Whole disk copy program
☐ Blank 5¼" disk
☐ Initialized DOS 3.3 disk with no files on it

Sometimes I forget to leave myself notes on how a deprotection was arrived at, so this softkey is short and simple.

1 Use your whole disk copier to copy the original Microzine #25 onto the blank 5¼" disk.

2 Boot the DOS 3.3 disk with no files on it to get into BASIC.

3 Take out the DOS 3.3 disk and insert the copy of the Microzine #25 disk.

4 Get into the monitor and load the file with the protection code.

CALL -151
BLOAD TOC.6

5 The file starts at $9000 and requires the accumulator to be zero, and the carry to be clear, for the protection call to be successful. Well, let's put the following three instructions at the start of the file, so that those two conditions will always be true:

```
9000 A9 00  LDA #$00   load accumulator with a zero
9002 18     CLC        clear the carry flag
9003 60     RTS        return to the calling routine
```

6 Now, save the file back to the copy of the Microzine #25 disk:

BSAVE TOC.6, A$9000, L$95

You're all done! Check out the TOC.6 file to see some protection code if you are new to the deprotection game.

† *Softkey for...*

Paint With Words & Word Art Show
MECC

■ **Requirements**

☐ MECC Paint With Words & Word Art Show original disk
☐ A freshly initialized 5¼" disk
☐ RWTS WORM program from COMPUTIST #61
☐ Super IOB v1.5
☐ NEW SWAP controller for Super IOB v1.5

The only protection here are simple format changes. Nothing hard at all, especially if you have my RWTS WORM program from COMPUTIST #61. The RWTS WORM does all of the hard work and gets the MECC RWTS for you. All you have to do is to use the NEW SWAP controller with the saved RWTS. Here's how.

1 Boot a normal DOS 3.3 disk and load the RWTS WORM program into memory.

BLOAD RWTS.WORM, A$9500

2 Insert the original MECC disk into slot 6, drive 1 and startup the RWTS WORM.

CALL 38144

3 Insert your disk with Super IOB v1.5 and save the RWTS. Then, load Super IOB, merge the NEW SWAP controller and change these two lines before you run it.

```
BSAVE RTWS.MECC, A$1900, L$800
LOAD SUPER IOB
EXEC NEW SWAP.CON
10010 PRINT CHR$(4); "BLOAD RWTS.MECC,A$1900"
1015 TK = 3
RUN
```

...answer NO to "Do you want to format the duplicate?" prompt.

You're done!

Softkey for...

Dive Bomber
Epyx

■ **Requirements**

☐ Dive Bomber original
☐ Blank 5¼" disk
☐ COPYA
☐ Sector editor

Dive Bomber is a somewhat interesting flight simulation game. Not the greatest graphics you'll find, but the game is pretty good nonetheless. The protection scheme used here is the good ol' stock Epyx protection of late. It is a variation of the protection on Boulderdash Construction Set, and in fact is the same variation that is used on the game Spiderbot (COMPUTIST #61). See my articles in COMPUTIST #57 and COMPUTIST #61 for a more in depth explanation of what's happening.

Here are the steps to deprotect it.

1 Boot your DOS 3.3 system disk.

2 Tell DOS to ignore checksum and epilog errors and use COPYA to copy the disk.

POKE 47426,24
RUN COPYA

3 Make the following sector edits to the copy you just made.

Trk	Sct	Byte(s)	From	To
$00	$09	$10-2B	A6 2B BD 89 C0 BD 8E C0 A9 80 85 FD C6 FD F0 71 20 AF 5F B0 6C A5 F9 C9 0F D0 F1 A0	A9 E7 85 F8 A9 FC 85 F9 85 FF A9 EE 85 FA 85 FD 85 FE A9 F3 85 FB A9 70 85 FC D0 50

You're all done!

Softkey for...

Kid Niki - Radical Ninja
Data East

■ **Requirements**

☐ Kid Niki original disk
☐ A blank disk
☐ COPYA
☐ Sector editor

Again, the lack of keeping good notes causes me to be able to tell you what to do, but not why it was done. Oh well.

1 Boot your DOS 3.3 system disk.

2 Tell DOS to ignore checksum and epilog errors and use COPYA to copy the disk.

POKE 47426,24
RUN COPYA

3 Make the following sector edits to the copy you just made.

Trk	Sct	Byte(s)	From	To
$00	$0E	$0F-$10	BD 89	D0 6E

Write that sector back to disk and you're all done!

Softkey for...

Introductory Algebra
Intermediate Algebra
Addison Wesley

Softkey for...

Fractions
Percents, and Decimals,Aquarius People Materials

■ **Requirements**

☐ Original disk
☐ A blank disk
☐ Super IOB v1.5 + NEW SWAP controller

Nothing too hard about these disks. Just simple format alterations.

1 Boot the original disk, and when you see the bracket prompt (]), press ⊖ C once. The computer should beep and you should see the Applesoft prompt (]).

2 Reset the Applesoft auto run flag.

FP

3 Get into the monitor and move the RWTS so we can use it.

CALL -151
1900<B800.BFFFM

4 Boot your Super IOB disk and save the RWTS.

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

5 Merge your NEW SWAP controller with Super IOB v1.5 and then add this line before you run it.

```
10010 PRINT CHR$(4)"BLOAD RWTS.PRG,A$1900"
RUN
```

Softkey for...

Computer Drill And Instruction: Mathematics 'Addition A' disk
Science Research Associates

■ **Requirements**

☐ ADDITION A original disk
☐ A blank 5¼" disk
☐ Whole disk copier

The protection on this disk is a good meaty one to sink your teeth into. Computists who are past 'POKE 47426,24, RUN COPYA' are advised to take a look here, because the protection scheme is a good one. Let's dig in!

Preliminaries

The disk can be copied by any whole disk copier, so there aren't any format changes to worry about. What kind of protection does that leave? If you said nibble count/signature check then you've been reading Computist! Yes, it is a nibble count/signature check, but the actual code itself is written in an extremely confusing way. So make a copy with COPYA (or equivalent) and put away the original - we won't need it any more.

Boot DOS 3.3 and insert the copied disk. Enter the monitor and load the file with the protection code.

CALL -151
BLOAD MPG CODE

The call to the protection code is not at the beginning of the file, $55FD, but rather it is at $5C31. Let's take a look at this code.

```
5C31L
5C31- A9 00     LDA #$00
5C33- 20 00 7B  JSR $7B00
5C36- 8D 6C 7A  STA $7A6C
...
```

The protection code itself lies at $7B00 and the disassembly you just looked at is where it is called from. I discovered that the protection is at $7B00 by tracing the code (not the best explanation for novices) as it was executed. This is something you can do after you have been at the deprotection business for a while - it just comes to you because of all the protection schemes you have been exposed to. Anyways, let's take a look at the start of the protection code:

```
7B00L
7B00- 20 0F 5A  JSR $5A0F
7B03- 20 1F 7C  JSR $7C1F
7B06- A9 0F     LDA #$0F
...
```

Looking at the code at $5A0F and $7C1F, I found that these subroutines decode the actual protection code. We'll go ahead and execute them so we can see the actual protection code:

5A0FG
7C1FG
7B00L

Looks different, huh? Now comes the interesting part. To make the code as confusing as possible, the authors have made branches to other sections of code that are always taken. By putting these branches into the correct places in the code, the true meaning of the code will not be apparent from a listing. Let's look at an example:

```
7B2AL
7B2A- A9 00     LDA #$00
7B2C- 48        PHA
7B2C- 28        PLP
7B2E- D0 01     BNE $7B31
7B30- 4C 20 E3  JMP $E320
7B33- 03        ???
...
```

All looks ok until we hit the JMP $E320. Why would they jump to somewhere in the Apple's ROMs? Take a look at the instruction right before the JMP $E320. Notice that if the zero flag is not set, then we branch to the middle of the JMP instruction! Hmmm...maybe the zero flag will ALWAYS NOT be set so that the branch is ALWAYS taken. Let's see what the code looks like if we put a NOP in place of the byte that is ALWAYS branched over:

7B30:EA
7B2AL

```
7B2A- A9 00     LDA #$00
7B2C- 48        PHA
7B2D- 28        PLP
7B2E- D0 01     BNE $7B31
7B30- EA        NOP
7B31- 20 E3 03  JSR $03E3
7B34- 85 01     STA $01
...
```

Now that looks a lot better! Notice that the JSR $03E3 is now revealed, which is a DOS 3.3 page three vector that locates the input parameter list for the RWTS. What is important to notice is that it is a call to a page three location, which is where DOS 3.3 puts it's vectors, and that it is an instruction that you normally wouldn't see, unless you made the change we did. If you were wondering if there are any more of these 'hidden' instructions, then let me assure you, there are lots more of them. I'm not going to list every hidden instruction for you; I leave that up to the individual who wants to sharpen their 6502 assembly and softkeying skills.

After I went through and found all of the hidden instructions and NOPed out all useless bytes, I found that on a successful protection check, the A-register contains a zero. All of the time the protection scheme's author must have spent writing confusing code, makes it seem foolish that the only results it needs is a zero in the A-register. What a waste of time. Well, it was a good mental exercise and those of you who went through and 'decyphered' the code are the better for it. The only thing left is to figure out how to modify the code so that it always reports the 'right' results. I found out at this point that the code modifies itself a bit more, so we will need to make a few more changes to the file, but nothing major.

1 Copy the original onto the blank with any whole disk copier.

2 Boot DOS 3.3 and get into BASIC.

3 Insert the original disk in the drive. Make sure it is write protected.

4 Enter the monitor and load the protection code file.

CALL -151
BLOAD MPG CODE

5 Decode the protection file.

5A0FG
7C1FG

6 Alter the file so that it forces a correct result.

7B99:A9 00 *was C9 AA*

7 Execute the protection code.

7B06G

8 NOP out the decoding subroutine calls:

7B00:EA EA EA EA EA EA *was 20 0F 5A 20 1F 7C*

9 Take out the original disk and insert the copy.

10 Keep a copy of the original protection file.

RENAME MPG CODE, OLD MPG CODE

11 Save the now docile protection file.

BSAVE MPG CODE, A$55FD, L$2630

You're done! You can replace the DOS on the disk with ProntoDOS, if you want faster boot times.

Playing Tip for...

Zany Golf
Electronic Arts

There is a secret level in the game Zany Golf. To find it, first get to the energy level. Now get to the top level and fall down through one of the holes (except for the one with the flag!). Notice the hole you come out through at the bottom level. Every few seconds a pair of 'eyes' appear. I leave it to you to figure out what to do now.

Apple IIgs+ rumors

I have heard and read quite a bit since last May about the rumored upcoming Apple IIgs+. I thought the readers might be interested to hear what I've heard.

??? The GS+ ???

RAM: 512K 'fast', 256K 'slow', 128K dedicated to Ensoniq sound chip

GRAPHICS: New VGC chip, makes old GS graphics look pale. Several new graphics modes
- 320x200 w/256 colors on screen at once
- 320x400 w/16 colors
- 640x200 w/16 colors
- 640x400 w/4 colors

SLOTS: New 'invisislots' or 'floating slots', allows a card in a slot and built in card for that slot to both be usable and active at the same time. For example, you could have your hard drive card in slot #4, which is the slot that has the built-in mouse card, and use both of them at the same time.

OTHER FEATURES:
- Built-in SCSI port
- ADB connector now on side of machine instead of back
- All tools now in ROM for faster boot time
- Maximum system speed is now 7.8 Mhz. Control Panel has three speed settings: NORMAL, FAST, and FASTER
- Control Panel is modified somewhat
- No dedicated graphics chip

There is some other info I have read on it, but the info above has pretty much stayed steady since I first heard about it last May. I also read about ProDOS 16 v2.0 and the IIc+ at the same time, and they both have appeared already. The original ProDOS 16 v2.0 was scrapped, and Apple decided it would be a better idea to port over the Macintosh's HFS to the GS. The result is GS/OS v2.0. The IIc+ was at first rumored to be a GS without slots, but it turned out to just be a 'IIc with a Zip Chip'. Another rumor that has been going around concerns a K-12 version of the Macintosh that Apple will be putting out soon. The GS+ and the K-12 Mac both seem, in my opinion, to be the same computer. Wouldn't that be great? Toss in a 1.44 meg HDFD 3½" drive and you would have the best system around.

Apple's mistake?

I think that Apple is really making a grave error with their recent price increases for the Macs. The MSDOS world right now is in disarray, with their different operating systems, standards, and program interfaces. Apple should be *lowering* the Mac prices so more and more folks would buy them. Market share should be the domineering drive at Apple now, not profit margin. Put more Macs out there and more people will buy them. This leads to more people and companies adopting the Mac as a business, home, and educational computer which in turn leads to more Mac sales. I work in our local community college and I also talk to many, many people about computers. Most of them tell me that they really would like to get a Macintosh (hear that, Apple?) but the Macs cost too much (hear that, Apple?), so these people end up getting a cheaper MSDOS clone because they can't afford a Mac (hear that, Apple?).

Let's face it Apple, the price of the Mac is out of reach of Joe and Josephene Public. Sure, it's worth the price (in my opinion), but they don't care; they want an affordable computer that will do the job. Why spend $1800 on a Mac Plus, when for the same price they could get an MSDOS clone that runs faster, has high resolution color, has a 20 meg internal hard drive, and has 640K for less money? Come on Apple - I don't want the MSDOS computers to become the computer standard. The Mac would be a much better and more logical standard, but the catch is that there have to be a *lot more* Macs out there in use. The home and small business markets, outnumber the large business, and Fortune 500 markets, in sheer number by such a large margin that it is ridiculous. But where is Apple targeting the Mac? Come on Apple, get with it and let the average person be able to own the class act (along with the IIgs) among computers. Since you are the 'only dealer' of the Macs, the ball is in your court...

A better 3½" drive?

I have an Apple IIgs with an Apple 3½" drive. I also have a Central Point 3½" drive. The Central Point drive doesn't write as fast, and it's a lot noisier than the Apple 3½" drive, but it has a few noteworthy advantages going for it. First of all, it's quite a bit cheaper. An Apple 3½" drive runs about $400, while the Central Point 3½" drive and accompanying Universal Drive Controller card cost about $300 (less through mail order). The main advantage is the Central Point 3½" drive's determination in reading a disk. If there is an error on a 3½" disk, the Apple 3½" drive gives up very quickly and reports an error when the bad spot is encountered. The Central Point drive, however, tries many times to read the bad spot before it gives up. This can be a lifesaver if your disk just happened to write a bad checksum for a block for whatever reason. This rugged determination has saved a few of my disks and data, and I am awfully glad I have the Central Point drive. I've had the drive for over 2 years now and I haven't had a single problem with it. Yes, there is a viable alternative to the Apple 3½" drive.

To Bill Jetzer: The strange 'M=00' that you get on your IIc is indeed not mentioned in the IIc manual. The 'M' register is the Machine State Register and tells you about the condition of your extended 80 column RAM card. It is organized as follows:

Machine State Register - (0=clear, 1=set)

- Bit 7 - Alternate page 0 of the LC active if set
- Bit 6 - Page 2 active if set
- Bit 5 - RAMRD active if set
- Bit 4 - RAMWRT active if set
- Bit 3 - ROLCROM active if set
- Bit 2 - LC bank 2 active if set
- Bit 1 - Alternate ROMBANK active if set
- Bit 0 - INTCXROM active if set

Hope that clears things up for you, Bill. Let me say your article in COMPUTIST #60 about reading from protected disks was well done. The tables on reading from protected DOS 3.2/3.3 and the standard DOS 3.3 RWTS locations should be of great help for beginning Computists.

To Alan Zimbard, M.D.: In your letter in COMPUTIST #60, you mentioned that the Fingerprint GSi card would allow you to go into the monitor, with one keystroke from the Fingerprint GSi menu. I have this card and no where in the manual does it even mention the feature you described. What keystroke takes you to the monitor? I have v2.0 of the card and I've put it into slot #3, as instructed in the manual. By the way, IIgs owners should look into getting this card. The screen dump capabilities are great, and you can edit the screen you want to save, as well as save it to disk. Posters as big as 100'x130' can be printed out from one graphic screen!

To Donald Jones: Converting an Appleworks file to an ASCII text is not all that hard, but it does take a few steps. Here's what to do:

■ **Requirements**

Appleworks program
Blank formatted DOS 3.3 disk
Blank formatted ProDOS disk (let's call it /DATA)
Your word processing file
DOS 3.3 to ProDOS conversion program, like Copy II Plus v6.0

1 Boot Appleworks and load your word processing file.

2 Insert /DATA into a drive.

3 Press ⌘ P to print the word processor file. When it asks you where to print it, specify 'to an ASCII text file'. When it asks you where to print it to, type in

/DATA/TEXTFILE

...and the file will eventually end up in a ProDOS ASCII text file called TEXTFILE on the /DATA disk.

4 Use your DOS 3.3 to ProDOS conversion program to convert and copy the ProDOS ASCII file onto the formatted DOS 3.3 disk.

5 That's all you have to do! Converting database and

spreadsheet files are just the same as converting a word processor file.

If you're going thru all that trouble just to send a letter to RDEX, don't bother. Send it as an Appleworks file on ProDOS. We can read those just fine. . . . . RDEXed

To Paul Johnson: You should be able to pick up a copy of Beneath Apple DOS from Quality Software directly. I bought the book for $19.95. Their address is:
Quality Software
21601 Marilla Street
Chatsworth, CA 91311

To Bud Myers: If you have any questions on deprotection or the Apple computer in general, you can always pass a letter my way and I'll see what I can do. Also, don't forget to send a copy of the letter to Computist, so others can try and help you out too.

To Andy Borne: The one way I know to load double hi-res screens, is with pictures that are saved as two binary files. The first file is BLOADed into main RAM, the second file is BLOADed and then moved into auxiliary RAM, and then the soft switches are set to display the double hi-res screen. The soft switches to set are in your extended 80 column card manual. I suggest that you get the program Beagle Graphics by Beagle Bros. The manual is chock full of goodies for programmers/hackers who want to do their own double hi-res programming.

To Ralph L. Jones: Your ProDOS Sensible Speller is not protected any more. The problem you have is that the name of a directory that Sensible Speller looks for is hard coded instead of being coded relatively. For example, Appleworks works fine on any disk, whether it be 5¼", 3½", RAM disk, or hard disk. That's because it looks for it's files all in the same directory, whatever and wherever that directory is. It doesn't care where the directory is, main or subdirectory, as long as all of the files are together in the same directory. Sensible Speller expects to find it's files on a disk called /SENSIBLE001 and it will not accept anything else. Now, if you were to hunt for all occurences of '/SENSIBLE001' on your Sensible Speller disk, and then changed them each to 'SENSIBLE001A', the program would look for the files in a subdirectory called SENSIBLE001A.

This subdirectory must be a first level directory, for example /HARDDISK/SENSIBLE001A or /MASTER /SENSIBLE001A. When you get a catalog of the main directory, the subdirectory name SENSIBLE001A must be in it. Now, if you want the program to be in a deeper subdirectory, then change the original '/SENSIBLE001' string to whatever you need, for example SENSIBLE/ABC if you want it to look in the subdirectory /HARDDISK/SENSIBLE/ABC. The only restriction is that the new string has to be the same length as the original string.

IIgs System monitor commands

The IIgs system monitor contains many commands that it's predecessor did not, in addition to having all of the old commands too. Here is a list of monitor commands that I have been able to dig up. Feel free to add any that I might have missed (there will probably be some obvious ones) or correct one that I explain incorrectly. If the monitor command is unchanged from the pre IIgs monitor command, then I'll just say it is the 'SAME as before'. Format is:

Command type, keystrokes to activate — comments on command

- USER command vector, ⊖ **Y** — SAME as before.
- Cold start BASIC, ⊖ **B** — SAME as before.
- Warm start BASIC, ⊖ **C** — SAME as before.
- Restore registers, ⊖ **R** — Restore registers to standard startup values.
- Examine registers, ⊖ **E** — SAME as before.
- Modify registers and flags, (value) = (register or flag) — Lets you alter a register or flag's value. Upper or lower case is important here.
- Call toolbox routine, \(# of bytes on stack) (# of bytes off stack) (parm#1...parm#N) (function #) (toolset #)\U — You really need a book with an explanation of the Apple IIgs toolbox to make good use of this command.
- List code in memory, (bank/address)L — Almost the same as the pre IIgs 'L' command. The status of the 'm' and 'x' flags determine the format of the disassembly.
- Run program in bank zero, (bank/address)G — SAME as before. Program must end in a RTS.
- Run program in any bank, (bank/address)X — Executes a program in any memory bank. Program must end in a RTL.
- Move memory, (destination)<(start).(end)M — SAME as before.
- Zap memory, (value)<(start).(end)Z — Fills memory range with (value).
- Verify memory, (destination)<(start).(end)V — SAME as before.
- Change memory, (bank/address):(value) ('inverted ASCII') ("literal ASCII") — SAME as before, if you enter in (value). "Literal ASCII" is enclosed in double quotes. Single quoted 'inverted ASCII' works the same as literal ASCII, except that the characters are entered in inverse order.
- Search memory, \(value) ('inverted ASCII') ("literal ASCII")\<(start).(end)P — This is my favorite monitor command on the IIgs. You can search memory for a byte, series of bytes, or literal ASCII characters. I do not use inverted ASCII in my searches at all and, in fact, I wonder why it is there. As an example of this command, say you want to search for the ProDOS 8 read block command (20 00 BF 80) from $800 to $9000 in bank 1. The search command you would use is:

\20 00 BF 80\<01/800.9000P

If this command is used in conjunction with the visit monitor CDA (ROM 01), cracking disks becomes much easier. It's kind of like having a poor man's Senior PROM in your IIgs.

- Set text screen, ⊖ **T** — Sets the screen to text page one.
- Normal, N — Same as before.
- Inverse, I — Same as before.
- Mini-assembler, ! — SAME as before.
- Change cursor, **ctrl** ^(character) — Lets (character) be the cursor.
- Exit memory range listing, ⊖ **X** — Stops the memory range listing.
- Redirect input, (slot#)⊖ **K** — Connects input hooks to (slot#).
- Redirect output, (slot#)⊖ **P** — Connects output hooks to (slot#).
- Dec to hex conversion, =(decimal value) — Converts (decimal value) to the hex equivalent.
- Hex to dec conversion, (hexadecimal value)= — Converts (hexadecimal value) to decimal equivalent.
- Addition, (value1)+(value2) — 32 bit addition.
- Subtraction, (value1)-(value2) — 32 bit subtraction.
- Multiplication, (value1)*(value2) — 64 bit multiplication.
- Division, (value1)_(value2) — 32 bit division.
- Display date and time, =T — Displays the date and time of your IIgs system clock.
- Change date and time, =T=(MM/DD/YY hh:mm:ss) — Allows you to change your system clock's time.
  - MM = month, range of 1-12
  - DD = day, range of 1-31
  - YY = year, range of 0-99
  - hh = hours, range of 0-23
  - mm = minutes, range of 0-59
  - ss = seconds, range of 0-59

Stephen A. Garbaty

A.P.T. for...

Lode Runner

Broderbund

Lately there have been a few articles published about "Enhancing" Lode Runner. The latest was by Paul Kippes in COMPUTIST #62. He describes how to get extra men, changing levels and changing your score by fixing the program code. There is a much easier way to do this, *while* playing the game. Try this:

⊖ **2** Gives you another guy everytime you press it.
⊖ **6** Advances you to the next level.
⊖ **A** Restarts current board and reduces man count by 1.
⊖ **R** Restarts game.
⊖ **S** Toggles sound On/Off.
esc Pauses game
⊖ **U** The right arrow will speed the game up.
⊖ **H** The left arrow will slow the game down.
⊖ **E** From title screen only, will put you into the editor, where you can play any level you like, or even create your own levels and store them to a data disk. It also contains a few other goodies.

I have been meaning to write to COMPUTIST for a long time, and seeing the challenge at the close of Paul's article gave me the motivation. He wanted to know if there is any way to stop the screen wipe between each level (The iris effect). Well I got to work and disassembled the code. After two days of work here's what I found:

The code starts at $88A2. It will be called after the level is loaded onto hi-res page 2. It will take the level from page 2 and transfer it to page 1, where you see it appear. It also updates the current level and number of men on the bottom of the screen, then returns to the game. The problems started when I tried to use the monitor move routine to take page 2 and place it on page 1. This wiped out the information (score,men,level) on the bottom of the screen. So I decided to use a monitor routine located at $F411. This routine will return the base address of any hi-res line you leave in the accumulator in address $26 & $27. By using a loop and transfering that line from page 2 to page 1, and then stopping before it overwrites the information on the bottom of the screen, you get an instant picture.

Here is the code you need. Enter the monitor, type it in, and save it.

```
CALL - 151                          to enter the monitor
88A2: A5 00 48 A5 01 48                            $6940
88A8: A9 40 85 E6 A9 00 8D EC                      $97B1
88B0: 88 A2 00 A0 00 AD EC 88                      $10D4
88B8: 20 11 F4 A5 26 85 00 A5                      $253A
88C0: 27 38 E9 20 85 01 A0 00                      $170D
88C8: B1 26 91 00 C8 C0 28 D0                      $35AA
88D0: F7 EE EC 88 AD EC 88 C9                      $FBD3
88D8: B0 D0 D6 A9 80 85 E6 20                      $3F0F
88E0: 70 7A 20 8C 7A 68 85 01                      $1B85
88E8: 68 85 00 60                                  $3E91
BSAVE INSTANTPIC, A$88A2, L$4C
```

To use it, BLOAD Lode Runner first, then this file, then enter the monitor and start Lode Runner with the command 5000G. If your copy starts someplace else in memory, substitute the proper starting address. You can also save the code directly into the program. First BLOAD Lode Runner and type in the code, then save it back to disk using the proper starting and ending addesses for your copy, this will make it permanent.

Thanks for the challenge Paul, and I hope everyone benefits from these tips. Incidentally, you can use InstantPic in your own programs with slight modifcations. Lines $88A2-$88A7 and $88DF-$88EA must be removed and you might want to add a time delay to slow it down (Located in monitor at $FCA8). Location $88DC holds the number of lines to transfer from page 2 to page 1.

Good luck and keep hacking.

C.E. "Chuck" Garrett

I really enjoy COMPUTIST and look forward each month to its arrival. I have not, as yet, derived much benefit, as I am still way down on the learning curve and have found no user group or other Apple users to learn by talking. I read most of the softkeys mainly to become acquainted with the lingo.

? Constant referance is made to "your DOS 3.3 System Disk". Is this the System Disk that came with earlier Apple Computers? Mine is the IIgs and the original System Disk was ProDOS, which was upgraded by Appple to a GS/OS System Disk. If this is so, how may I obtain one?

? Frequent reference is also made to "COPYA". Is this a commercial copy program such as Copy II Plus? I have never seen any ads for it.

So far the only success I have had, in making back up copies of my programs, is by using "Quick and Dirty" from Locksmith. This is only partially successful. For example:

A Q&D copy of Additions Logicians won't load completely and therefore won't run. It stops with a partial screen display and just shuts down, the boot drive stops reading.

A Q&D copy of Flight Simulator II does exactly the same thing.

A Q&D copy of Sticky Bear Reading copies but one of the three menu choices will not load. When selected, it just sits and the program is then locked and you have to re-boot to continue.

A Q&D copy of Circus Math runs OK, except one of the menu choices will not load. This does not lock the program and other selections can be made.

? If you can help, by providing answers/ solutions to any of the above, I will be grateful. Also, if any readers in the Hanford, CA area would like to establish a dialogue, I could use the help and will provide any limited assistance I am able.

James A. Hodge

Softkey for...

Calendar Crafter v1.1
MECC

To produce a broken version of Calendar Crafter perform the following block edit:

Block	Byte(s)	From	To
$567	BD	B0 26	80 20
	DF	00 D0 02	A9 27 00

The protection code is shown below. The program reads block 9, compares the first 20 bytes read, then expects an I/O error (#$27) when it tries to read block 8. The above edits change the "BCS $B1B5" to "BRA $B1AF" and puts a "LDA #$0027" at $B1AF, just ahead of the CLC instruction. This seems to satisfy the protection requirements.

```
B17D A9 09 00     LDA #0009
B180 8D BC B1     STA B1BC                 Block num.
B183 22 A8 00 E1  JSL E100A8              Read block 9
B187 22 00                  0022
B189 B6 B1 01 00            0001B1B6
B18D B0 26        BCS B1B5             shouldn't branch
B18F A2 13 00     LDX #0013
B192 BD C0 B3     LDA B3C0,X
B195 DD C0 B1     CMP B1C0,X          check what was read
B198 D0 1A        BNE B1B4             shouldn't branch
B19A CA           DEX
B19B CA           DEX
B19C 10 F4        BPL B192
B19E CE BC B1     DEC B1BC   set to read block 8 (bad block)
B1A1 22 A8 00 E1  JSL E100A8                try to read it
B1A5 22 00                  0022
B1A7 B6 B1 01 00            0001B1B6
B1AB 90 07        BCC B1B4             shouldn't branch
B1AD C9 27 00     CMP #0027        make sure it was I/O error
B1B0 D0 02        BNE B1B4         if not, set carry, return
B1B2 18           CLC
B1B3 60           RTS
B1B4 38           SEC
B1B5 60           RTS
```

Softkey for...

Chessmaster 2100 v1.01
Software Toolworks

This is an impressive game, with beautiful graphics and just about all the bells and whistles they could fit on a 3½" disk. It seems to play an excellent game of chess too, but I'm no Bobby Fischer.

The two annoying features are the slow boot time and the password protection.

The slow boot time can be sped up by adding a disk cache. The problem is that there are only two free blocks on the disk. Nine more blocks can be freed up by deleting FINDER files, but that still isn't enough space. The easiest, most effective way to gain over 120 blocks of free space is to delete all AppleTalk related files in the /CMASTER/SYSTEM/SYSTEM.SETUP directory.

1 In the /CMASTER/SYSTEM/SYSTEM.SETUP directory, delete the following files: ATSETUP, ATINIT, ATPATCH, ATSTART, PFILOAD, SPLOAD, ATROM, ATRESPONDER

2 Copy a disk cache program into the /CMASTER /SYSTEM/SYSTEM.SETUP subdirectory.

I used the AECACHE.SETUP cache program. With a 64K cache, boot time dropped from 2:13 to 1:25, a savings of 48 seconds.

The second problem took me a little more time to solve. Password protection is cute for about the first two times, and then it becomes annoying.

The first solution doesn't eliminate the password question, but it means you don't need to go leafing through the manual to find the password. You need to make sure that "Visit Monitor" is installed if you have a IIgs with the 01 ROMs, or install something like DiveriHack or the "Memory Mangler" desk accessory from the GSDebug disk if your IIgs has the 00 ROMs. (Get the upgrade!)

When the Chessmaster game gets to the password question, go into the monitor by pressing ⌘⌥ **esc** and selecting the "Visit Monitor" option. Dump memory from 1750.1780 and you'll see the answer to the question. Entering ⌥ **Y return** gets you back to the Control Panel, and an additional **return** gets you back to the game. Answer the question and continue normally.

The second solution to the password nuisance requires editing two blocks, but it gets you into the game faster because it eliminates the question-and-answer session. Perform the following edits:

Block	Byte(s)	From	To
$29E	$187	C9 02	82 0A
$5B2	$1DE	0B	6B

That will do it. Enjoy your chess game.

Softkey for...

Crossword Magic 4.0
Mindscape

This game had several layers of protection, which made it an interesting challenge to crack.

Using an Apple IIgs with its "Visit Monitor" option made the job a lot easier. When you enter the Control Panel, the registers and stack pointer are saved at E1/108.10F (2 bytes each for A, X, Y, and SP), the stack is saved at E0/300.3FF, page zero is saved at E0/1C00.1CFF, and the text screen ($400.7FF) is saved at E0/1400.17FF. Examining the stack will tell you the address of the instruction that was executing, and usually the stack will reveal a trail of return addresses. "Visit Monitor" is also handy when used in conjunction with a sector editor. You can disassemble 16 bit code (m=0 and x=0) and you can use the mini-assembler to apply large patches to sectors and blocks. (To find where your particular sector/block editor has its buffer, use the List disassembler command. Copy II Plus v8.3 has the 5¼" buffer at $E00-EFF and the 3½" buffer at $1100-12FF.)

Examining the disk with the Copy II Plus sector editor (with the patch option) showed that tracks $00 through $0E used FF epilog bytes, and that they had track numbers different from the actual track location. The Trax option in Bag of Tricks revealed that these tracks were all numbered 0, some of the sectors had bad checksums, and the disk had been formatted as volume 0. Tracks $0F to $11 were normal.

The next step was to create a controller that would copy the original disk to a disk with normal formatting. Lines 1015, 1017, 1027 and 1028 in the controller allowed DOS to ignore track and checksum errors. The controller needed two sections, one to copy tracks $00 through $0E, and the second section for tracks $0F through $11.

Once I had a normalized disk, I tried booting it. Needless to say, it didn't work. It hung in the disk controller code. (I found that out from "Visit Monitor".) The stage 1 boot code uses the sector read code in the disk controller ($C65C-C6FA). The stage 1 boot code had stepped the drive arm to track 1 but had not set location $41 for the disk controller, and it kept failing the track test. I was able to insert an instruction to increment location $41 so the boot process would continue.

At this point, stage 1 boot would work, but now the disk hung someplace else. The next two problem spots were in the stage 2 boot code, specifically the "read address" and "read sector data" routines. They were looking for FF epilogs, so I changed them to look for normal DE AA epilogs.

The disk would display the first and second title screens and die, so I had to find the next step in the protection scheme. Once again, "Visit Monitor" to the rescue. Stage 2 boot would read track $02, sector $0B to $600 and JSR to it. That code could be seen at E0/1600.16FF with "Visit Monitor". The protection code would look for the address header for sector $0E, then look for a D5. Within $100 bytes of the D5 it would expect to find three consecutive E7s, and within $10 bytes of the last E7 it expected to find a EE. Then it would copy the next 6 key bytes to $20-25. All I had to do was find the values for the key bytes.

I used what I call a "Trojan Horse". I changed the JSR $600 that executed the protection code to jump to code I supplied. My routine would wait for a keypress so I could insert the original disk, then (after a keypress) it would proceed to successfully read the key bytes and save them in bank 1 memory.

Once I had the key byte values I placed a little routine into the protection code that would read its own data and copy those values to $20-25. Then it would continue with the last piece of the protection code.

I now had a game that worked, but there were two last "gotcha's". The printer routine has its own protection. Presumably, part of the reason it's there is to identify whether the disk is a data disk or the original disk.

The printer protection routine is almost identical to the routine mentioned above, but it is called twice, looking for sectors $07 and $09. The key bytes are copied to $3A-41, but only 4 bytes are different the second time through. The patch to defeat the printer protection is similar in nature to the patch for the general disk protection, but it's larger and smarter. It took only 21 bytes to defeat the disk protection, but it takes 41 bytes to fix the printer routine.

The IOB included in this article will make all the changes need to produce a COPYA-able version of Crossword Magic 4.0. For those of you who would like to save some effort keying it in, if you don't mind applying patches with a sector editor, I've included hex dumps of the patches for track $02, sector $0B (11) and track $0E (14), sector $00. If you decide to manually apply the patches, you can omit the data values for (t2,s11) and (t14,s0), a total of 248 numbers.

For the really lazy, the patches for (t2,s11) and (t14,s0) will allow a bit copy to work reliably. Use the Patch option in Copy II Plus's sector editor. Set it to Custom with FF epilogs, ignore track, ignore data checksum, and then, manual sector copy tracks $00 to $0E and perform the necessary edits. Set the Patch option in the sector editor to DOS 3.3, and copy tracks $0F to $11 with manual sector copy. I've also included an AUTO PARM for Copy II Plus.

For the Super IOB sector edit option to work, line 330 needs to be changed. It's listed with the controller.

Track $02, sector $0B patch

```
14: A2 05 BD 23
18: 06 9D 20 00 CA 10 F7 A0
20: 06 D0 48 E7 FC EE E7 FC
28: EE
```

Track $0E (14), Sector $00 patch

```
25: C9 07 D0
28: 0C A2 08 BD 43 07 95 39
30: CA 10 F8 30 0A A2 03 BD
38: 4C 07 95 3C CA 10 F8 EA
40: EA 18 60 FD E7 FC E7 EE
48: FC EE EE FC EE E7 EE FC
```

Controller

```
330 IF A1 < T1 OR A1 > PEEK (TRK) THEN NEXT : RETURN
1000 REM Crossword Magic 4.0
1010 TK = 0:LT = 15:ST = 15:LS = 15:CD = WR:FAST = 1
1015 POKE 48628,208: POKE 48629,26: REM TRK ERR
1017 POKE 47405,24: POKE 47406,96: REM CHKSUM
1020 T1 = TK: GOSUB 170: GOSUB 490: GOSUB 610
1025 GOSUB 310: RESTORE
1027 POKE 48628,173: POKE 48629,120
1028 POKE 47405,208: POKE 47406,19
1030 GOSUB 220: GOSUB 490: GOSUB 610: IF PEEK (TRK) = LT
  THEN 1050
1040 TK = PEEK (TRK) :ST = PEEK (SCT): GOTO 1015
1050 TK = 15:LT = 18:ST = 15:LS = 15:CD = WR:FAST = 1
1060 T1 = TK: GOSUB 490: GOSUB 610
1070 GOSUB 490: GOSUB 610: IF PEEK (TRK) = LT THEN 1090
1080 TK = PEEK (TRK) :ST = PEEK (SCT): GOTO 1060
```

```
1090 HOME : PRINT "COPYDONE" : END
2000 DATA 255,255,255,255
2010 DATA 71CHANGES,0,0,97,254,0,0,98,8,0,0,254,230
     ,0,0,255,65
2020 DATA 1,1,206,222,1,10,77,222,1,10,87,170
2030 DATA 2,11,20,162,2,11,21,5,2,11,22,189,2,11,
     23,35,2,11,24,6
2040 DATA 2,11,25,157,2,11,26,32,2,11,27,0,2,11,28
     ,202,2,11,29,16
2050 DATA 2,11,30,247,2,11,31,160,2,11,32,6,2,11,
     33,208,2,11,34,72
2060 DATA 2,11,35,231,2,11,36,252,2,11,37,238,2,11
     ,38,231,2,11,39,252
2070 DATA 2,11,40,238,9,0,138,234,9,0,139,234
2080 DATA 14,0,37,201,14,0,38,7,14,0,39,208
2090 DATA 14,0,40,12,14,0,41,162,14,0,42,8
2100 DATA 14,0,43,189,14,0,44,65,14,0,45,7
2110 DATA 14,0,46,149,14,0,47,57,14,0,48,202
2120 DATA 14,0,49,16,14,0,50,248,14,0,51,48
2130 DATA 14,0,52,10,14,0,53,162,14,0,54,3
2140 DATA 14,0,55,189,14,0,56,74,14,0,57,7
2150 DATA 14,0,58,149,14,0,59,60,14,0,60,202
2160 DATA 14,0,61,16,14,0,62,248,14,0,63,24
2170 DATA 14,0,64,96,14,0,65,253,14,0,66,231
2180 DATA 14,0,67,252,14,0,68,231,14,0,69,238
2190 DATA 14,0,70,252,14,0,71,238,14,0,72,238
2200 DATA 14,0,73,252,14,0,74,238,14,0,75,231
2210 DATA 14,0,76,238,14,0,77,252
```

Checksums

330 - $4A8D	1070 - $605D	2100 - $9F92
1000 - $FC06	1080 - $AAAA	2110 - $9562
1010 - $9644	1090 - $7D80	2120 - $D99E
1015 - $803D	2000 - $D1A8	2130 - $493E
1017 - $8BE1	2010 - $DE44	2140 - $646B
1020 - $76B0	2020 - $8FC8	2150 - $9717
1025 - $3EC6	2030 - $8B27	2160 - $F187
1027 - $0384	2040 - $89A3	2170 - $2201
1028 - $C83B	2050 - $9CFC	2180 - $4C2E
1030 - $F5B7	2060 - $7B60	2190 - $73BC
1040 - $48AE	2070 - $0C6D	2200 - $1B92
1050 - $C9D8	2080 - $F5CF	2210 - $9500
1060 - $53B4	2090 - $4792	

Auto parms for Copy II Plus

```
T0-T1, SECTOR COPY, 5C=FF, 5D=FF, 66=FF, 67=FF
T2, SECTOR COPY, 5C=FF, 5D=FF, 65=00, 66=FF, 67=FF
SECTOR EDIT, TRACK 2, SECTOR 0B, DOS 3.3,
14:A2/05/BD/23/06/9D/20/00/CA/10/F7/A0/06/D0/48/
E7/FC/EE/E7/FC/EE
T3-TE, SECTOR COPY, 5C=FF, 5D=FF, 65=FF, 66=FF, 67=FF
SECTOR EDIT, TRACK E, SECTOR 00, DOS 3.3,
25:C9/07/D0/0C/A2/08/BD/43/07/95/39/CA/10/F8/30//
0A/A2/03/BD/
4C/07/95/3C/CA/10/F8/EA/EA/18/60/FD/E7/FC/E7/EE/
FC/EE/EE/FC/EE/E7/EE/FC
TF-T11, SECTOR COPY, 5C=DE, 5D=AA, 66=DE, 67=AA
```

Softkey for...

Dive Bomber

Epyx

This game has the "standard" Epyx protection scheme. They scan a track for a particular pattern of bytes and when they find them, they copy the key bytes that follow the pattern down to page zero. Then the key bytes are Exclusive OR'ed with another couple of pages of bytes to produce usable code. Then the boot process continues.

Just use the controller to create a COPYA-able copy of Dive Bomber.

Controller

```
330 IF A1 < T1 OR A1 > PEEK (TRK) - 1 THEN NEXT : RETURN
1000 REM EPYX DIVE BOMBER
1010 TK = 0:LT = 35:ST = 15:LS = 15:CD = WR:FAST = 1
1020 T1 = TK: GOSUB 170: GOSUB 490: GOSUB 610
1025 GOSUB 310: RESTORE
1030 GOSUB 230: GOSUB 490: GOSUB 610: IF PEEK (TRK) = LT
     THEN 1050
1040 TK = PEEK (TRK) :ST = PEEK (SCT): GOTO 1020
1050 HOME : PRINT "COPY DONE." : END
2000 DATA 255,255,255,255
2010 DATA 14 CHANGES,0,9,43,169,0,9,44,231,0,9,45
     ,133,0,9,46,248
2020 DATA 0,9,47,169,0,9,48,252,0,9,49,133,0,9,50
     ,249,0,9,51,169
2030 DATA 0,9,52,238,0,9,53,133,0,9,54,250,0,9,55
     ,208,0,9,56,67
```

Checksums

330 - $DD80	1025 - $D6BB	2000 - $B953
1000 - $7DD7	1030 - $E322	2010 - $6D9B
1010 - $394C	1040 - $6B0B	2020 - $58C3
1020 - $9E5D	1050 - $E347	2030 - $36C9

ProDOS EOR Disk Scanner

In Computist #57 Phil Goetz presented a program to scan a 5¼" floppy for data that had been encrypted by EOR'ing it with some constant value. The only drawback was that it didn't work on 3½" ProDOS disks.

I needed a program for ProDOS, so I adapted Phil's program. This should be able to work on any ProDOS volume. I know it works on 3½" and 5¼" disks.

Since ProDOS tends to store files in ascending blocks, I felt the option to scan down wasn't needed. Otherwise, this program behaves the same way as the original.

To use it, enter the length of the pattern you want to search for at $300 and the search pattern at $301. Then, BRUN PROSCAN, or, if the program is already in memory, execute it with a 2000G.

Matches are reported with the message 'BL$1234 01DF:C0', which would mean a match was found for your search string in block $1234 starting at byte $01DF, and the string had to be EOR'd with $C0 to match.

The device to be searched can be set in byte $2189. To access different drives set $2189 as follows:

slot 6, drive 1 - $60
slot 6, drive 2 - $E0
slot 5, drive 1 - $50
slot 5, drive 2 - $D0

Change bytes $2185-2186 for the maximum number of blocks. ProDOS 8 limits the maximum value to $FFFF. If you need to scan a device with more than 32 megabytes, simply convert the program to ProDOS 16. Use $118 or $640 for 5¼" and 3½", respectively, and enter them in the lo-byte hi-byte format (IE. 18 01 or 40 06).

I used the Merlin 8/16 Assembler to create the source code. The generated object code should make it easy to produce identical code even if you're using a different assembler. A hex dump of ProSCAN is also included.

```
                 *ProDOS EOR Disk scanner
                 *adapted by James A. Hodge from
                 *the DOS 3.3 version by Phil Goetz
                 *pub. Computist 57

       =0024     CH       =    $0024
       =0025     CV       =    $0025
       =0300     LEN      =    $0300
       =0301     STRING   =    $0301

       =2200     BUFFER   =    $2200
       =BF00     MLI      =    $BF00

       =C000     KEY      =    $C000
       =C010     STROBE   =    $C010

       =DB3A     STROUT   =    $DB3A
       =F941     PRNTAX   =    $F941
       =FB2F     INIT     =    $FB2F
       =FC22     VTAB     =    $FC22
       =FD8E     CROUT    =    $FD8E
       =FDDA     PRBYTE   =    $FDDA
       =FDF0     COUT1    =    $FDF0
       =FE84     SETNORM  =    $FE84
       =FE89     SETKBD   =    $FE89
       =FE93     SETVID   =    $FE93

                          TR   ADR
                          ORG  $2000

2000: 20 89 FE            JSR  SETKBD      SCAN start
2003: 20 93 FE            JSR  SETVID
2006: A9 01               LDA  #$01
2008: 8D 7D 21            STA  FLAG        tell getbyte to read sec.
200B: A9 00               LDA  #$00        make sure stuff is
200D: 8D 8C 21            STA  BLOCK       initialized properly
2010: 8D 8D 21            STA  BLOCK+1
2013: 8D 81 21            STA  PTR
2016: 8D 87 21            STA  Zor1
2019: 20 8E FD            JSR  CROUT
201C: 20 C8 20            JSR  PRN_BL

201F: A2 00          GO   LDX  #$00
2021: 8E 7F 21            STX  MIND        match index
2024: AD 00 03            LDA  LEN
2027: 8D 7E 21            STA  LEN2        countdown to match
202A: AD 81 21            LDA  PTR
202D: 8D 80 21            STA  MSTART
2030: AD 8C 21            LDA  BLOCK
2033: 8D 82 21            STA  BLK0
2036: AD 8D 21            LDA  BLOCK+1
2039: 8D 83 21            STA  BLK0+1
203C: 20 DC 20       L1   JSR  GETBYTE
203F: B0 7B =20BC         BCS  SoLong
2041: AE 7F 21            LDX  MIND        match index
2044: D0 08 =204E         BNE  L2
2046: 48                  PHA
2047: 4D 01 03            EOR  STRING
204A: 8D 7C 21            STA  EOR
204D: 68                  PLA
204E: 5D 01 03       L2   EOR  STRING,X
2051: CD 7C 21            CMP  EOR
2054: D0 35 =208B         BNE  L3
2056: EE 7F 21            INC  MIND        match index
2059: CE 7E 21            DEC  LEN2        countdown to match
205C: D0 DE =203C         BNE  L1          if no match

205E: AD 83 21            LDA  BLK0+1      Complete match
2061: AE 82 21            LDX  BLK0
2064: 20 D5 20            JSR  PRN_BLK
2067: A9 0A               LDA  #10
2069: 85 24               STA  CH
206B: AD 87 21            LDA  Zor1
206E: AE 80 21            LDX  MSTART
2071: 20 41 F9            JSR  PRNTAX      print 2 byte address
2074: A9 BA               LDA  #":"
2076: 20 F0 FD            JSR  COUT1
2079: AD 7C 21            LDA  EOR
207C: 20 DA FD            JSR  PRBYTE
207F: 20 C8 20            JSR  PRN_BL
2082: AD 8D 21            LDA  BLOCK+1
2085: AE 8C 21            LDX  BLOCK
2088: 20 D5 20            JSR  PRN_BLK

                 *Go back & start at byte after beginning of
                 *last match attempt, see if we need to go
                 *back a sector.

208B: AC 80 21       L3   LDY  MSTART
208E: C8                  INY
208F: F0 25 =20B6         BEQ  L5
2091: CC 81 21            CPY  PTR
2094: 90 20 =20B6         BCC  L5
2096: AD 81 21            LDA  PTR
2099: F0 1B =20B6         BEQ  L5
209B: AD 87 21            LDA  Zor1        if Zor1=0, just toggle
209E: F0 0F =20AF         BEQ  L4          2nd half of block
20A0: AD 82 21            LDA  BLK0
20A3: 8D 8C 21            STA  BLOCK
20A6: AD 83 21            LDA  BLK0+1
20A9: 8D 8D 21            STA  BLOCK+1
20AC: 20 63 21            JSR  GETBLOK
20AF: 20 52 21       L4   JSR  TOGGLE
20B2: AC 80 21            LDY  MSTART
20B5: C8                  INY
20B6: 8C 81 21       L5   STY  PTR
20B9: 4C 1F 20            JMP  GO

20BC: AD 10 C0       SoLong LDA STROBE
20BF: 20 84 FE            JSR  SETNORM     set normal video
20C2: 20 2F FB            JSR  INIT
20C5: 4C D0 03            JMP  $3D0        So long

20C8: A0 21          PRN_BL LDY #>BLK_HDR
20CA: A9 6D               LDA  #BLK_HDR
```

```
20CC: 20 3A DB          JSR STROUT      print string
20CF: A5 25             LDA CV
20D1: 8D 84 21          STA VERTTAB
20D4: 60                RTS

20D5: A0 03    PRN_BLK  LDY #$03
20D7: 84 24             STY CH
20D9: 4C 41 F9          JMP PRNTAX

20DC: AD 00 C0 GETBYTE  LDA KEY
20DF: C9 A0             CMP #$A0        pause
20E1: F0 F9 =20DC       BEQ GETBYTE
20E3: C9 9B             CMP #$9B        exit on escape
20E5: F0 69 =2150       BEQ BYE
20E7: AC 81 21          LDY PTR
20EA: D0 08 =20F4       BNE GETBYT1
20EC: AD 80 21          LDA MSTART
20EF: 0D 7D 21          ORA FLAG
20F2: D0 08 =20FC       BNE NEWSEC
20F4: B9 00 22 GETBYT1  LDA BUFFER,Y
20F7: EE 81 21          INC PTR
20FA: 18                CLC
20FB: 60                RTS

20FC: AD 7D 21 NEWSEC   LDA FLAG
20FF: D0 05 =2106       BNE NSEC
2101: 20 52 21          JSR TOGGLE
2104: D0 EE =20F4       BNE GETBYT1
2106: AD 84 21 NSEC     LDA VERTTAB
2109: 85 25             STA CV
210B: 20 22 FC          JSR VTAB
210E: A0 00             LDY #$00
2110: 8C 7D 21          STY FLAG
2113: EE 8C 21          INC BLOCK
2116: D0 03 =211B       BNE NSEC1
2118: EE 8D 21          INC BLOCK+1
211B: AD 8C 21 NSEC1    LDA BLOCK
211E: CD 85 21          CMP MAXBLOX
2121: D0 08 =212B       BNE NSEC2
2123: AD 8D 21          LDA BLOCK+1
2126: CD 86 21          CMP MAXBLOX+1
2129: F0 25 =2150       BEQ BYE         done
212B: AD 8D 21 NSEC2    LDA BLOCK+1
212E: AE 8C 21          LDX BLOCK
2131: 20 D5 20          JSR PRN_BLK
2134: 20 63 21          JSR GETBLOK
2137: A0 00             LDY #$00
2139: 90 B9 =20F4       BCC GETBYT1     if no error
213B: A0 21             LDY #>ERRMSG
213D: A9 72             LDA #ERRMSG
213F: 20 3A DB          JSR STROUT      print string
2142: AD 8E 21          LDA ERROR
2145: 20 DA FD          JSR PRBYTE
2148: 20 C8 20          JSR PRN_BL
214B: 4C FC 20          JMP NEWSEC

214E: 18                CLC
214F: 60                RTS

2150: 38       BYE      SEC
2151: 60                RTS

2152: AD F6 20 TOGGLE   LDA GETBYT1+2   toggle which half
2155: 49 01             EOR #1          of buffer
2157: 8D F6 20          STA GETBYT1+2   to access
215A: AD 87 21          LDA Zor1
215D: 49 01             EOR #1
215F: 8D 87 21          STA Zor1
2162: 60                RTS

2163: 20 00 BF GETBLOK  JSR MLI
2166: 80                HEX 80          READBLOCK
2167: 88 21             DA PARMS
2169: 8D 8E 21          STA ERROR
216C: 60                RTS

216D: 8D       BLK_HDR  HEX 8D
216E: C2 CC A4          ASC "BL$"
2171: 00                HEX 00

2172: A0 A0 A0 C5 ERRMSG ASC "   ERR #$"  three spaces
2176: D2 D2 A0 A3
217A: A4
217B: 00                HEX 00

217C: 00       EOR      HEX 00          EOR byte
217D: 00       FLAG     HEX 00
217E: 00       LEN2     HEX 00          countdown to match
217F: 00       MIND     HEX 00          match index
2180: 00       MSTART   HEX 00          match start index
2181: 00       PTR      HEX 00          pointer into buffer
2182: 00 00    BLK0     HEX 0000
2184: 00       VERTTAB  HEX 00
2185: 18 01    MAXBLOX  DW 280          $118 OR $640
2187: 00       Zor1     HEX 00

2188: 03       PARMS    HEX 03
2189: 60                HEX 60          unit ± = DSSS0000
218A: 00 22             DA BUFFER
218C: 00 00    BLOCK    DS 2
218E: 00       ERROR    HEX 00

End Merlin-16 assembly, 399 bytes, errors: 0
```

Proscan Hex Dump

```
2000: 20 89 FE 20 93 FE A9 01   $242F
2008: 8D 7D 21 A9 00 8D 8C 21   $D68B
2010: 8D 8D 21 8D 81 21 8D 87   $BCB4
2018: 21 20 8E FD 20 C8 20 A2   $B0B0
2020: 00 8E 7F 21 AD 00 03 8D   $C464
2028: 7E 21 AD 81 21 8D 80 21   $D7EE
2030: AD 8C 21 8D 82 21 AD 8D   $6B4B
2038: 21 8D 83 21 20 DC 20 B0   $9CAF
2040: 7B AE 7F 21 D0 08 48 4D   $F840
2048: 01 03 8D 7C 21 68 5D 01   $7C21

2050: 03 CD 7C 21 D0 35 EE 7F   $9E01
2058: 21 CE 7E 21 D0 DE AD 83   $EB6E
2060: 21 AE 82 21 20 D5 20 A9   $53F6
2068: 0A 85 24 AD 87 21 AE 80   $5575
2070: 21 20 41 F9 A9 BA 20 F0   $3E61
2078: FD AD 7C 21 20 DA FD 20   $E6A4
2080: C8 20 AD 8D 21 AE 8C 21   $70A9
2088: 20 D5 20 AC 80 21 C8 F0   $0CBD
2090: 25 CC 81 21 90 20 AD 81   $CCCA
2098: 21 F0 1B AD 87 21 F0 0F   $3F32

20A0: AD 82 21 8D 8C 21 AD 83   $E6DF
20A8: 21 8D 8D 21 20 63 21 20   $597D
20B0: 52 21 AC 80 21 C8 8C 81   $304E
20B8: 21 4C 1F 20 AD 10 C0 20   $7B4F
20C0: 84 FE 20 2F FB 4C D0 03   $8821
20C8: A0 21 A9 6D 20 3A DB A5   $C098
20D0: 25 8D 84 21 60 A0 03 84   $8780
20D8: 24 4C 41 F9 AD 00 C0 C9   $A5C6
20E0: A0 F0 F9 C9 9B F0 69 AC   $0D45
20E8: 81 21 D0 08 AD 80 21 0D   $D5E0

20F0: 7D 21 D0 08 B9 00 22 EE   $7537
20F8: 81 21 18 60 AD 7D 21 D0   $C934
2100: 05 20 52 21 D0 EE AD 84   $8F9B
2108: 21 85 25 20 22 FC A0 00   $EB75
2110: 8C 7D 21 EE 8C 21 D0 03   $A0C4
2118: EE 8D 21 AD 8C 21 CD 85   $BF3E
2120: 21 D0 08 AD 8D 21 CD 86   $EB60
2128: 21 F0 25 AD 8D 21 AE 8C   $D6D1
2130: 21 20 D5 20 20 63 21 A0   $4117
2138: 00 90 B9 A0 21 A9 72 20   $8F33

2140: 3A DB AD 8E 21 20 DA FD   $653F
2148: 20 C8 20 4C FC 20 18 60   $C3DD
2150: 38 60 AD F6 20 49 01 8D   $97D2
2158: F6 20 AD 87 21 49 01 8D   $B918
2160: 87 21 60 20 00 BF 80 88   $2E2E
2168: 21 8D 8E 21 60 8D C2 CC   $A1D5
2170: A4 00 A0 A0 A0 C5 D2 D2   $6926
2178: A0 A3 A4 00 00 00 00 00   $CA81
2180: 00 00 00 00 00 18 01 00   $08EC
2188: 03 60 00 22 00 00 00      $084B
```

Softkey for...

Scrabble

Electronic Arts

Here's a game that never should have made it past the beta test phase. Actually, it never should have made it to the beta test phase. This game demonstrates the level of British quality control that destroyed their motorcycle industry and did extensive damage to their auto industry. Apparently, Leisure Genius is a reference to the game's programmers.

While it plays a good game of Scrabble, it has the extremely annoying flaw of constantly recalibrating the disk drive. This game needs to be broken to be fixed.

The disk actually has two DOS systems on it. Both are modified DOS 3.3. The boot DOS looks like a fairly complete 48K DOS, but with most of its routines, and their entry points, moved around. It's modified enough that a controller using a "SWAP RWTS" isn't practical. The other DOS is loaded with the game, and it is relocated to $B400. It consists of most of the RWTS code, and a few routines to read and write the catalog, and load and save the dictionary and save game files. It also has a lot of dead, apparently unused, code, as does the rest of the game.

If I had sent COMPUTIST this article when I finished it, it probably would have been published before the Scrabble article by Edward Teach (COMPUTIST #63, pg. 30). He didn't mention the copyright date, but it appears that his version of Scrabble may have been different from mine, because the hello program is a different length and he didn't mention anything about the disk drive recalibration problem. His technique for capturing the files should work, with minor modifications, on a IIgs, but his Super IOB didn't work with my copy of Scrabble.

The specific version that I broke has copyright dates of 1984, 85, 86 and 87, but I suspect that any other versions will be very similar.

To produce a broken version of the game requires a 48K or 64K Apple that will allow you to reset into the monitor, an Apple IIe or IIc with 128K, a IIgs with the 01 ROMs and "Visit Monitor", or a IIgs with Diversi-Hack. (Let's see now; I guess that about covers it.)

Cracking Scrabble on a 48K or 64K Apple II+.

You need to be able to RESET into the monitor, either with an old Monitor ROM (pre-Autostart), a modified F8 ROM, or a non-maskable interrupt card (e.g., Wildcard or Snapshot).

$\boxed{1}$ Boot the Scrabble disk. When the title screen appears and the disk drive stops spinning, drop into the monitor.

$\boxed{2}$ Move the Applesoft hello program to a safe area of memory and note the program length.

4000<800.D00M
AF.B0 *note these values - 00AF- 25 00B0- 0B*

$\boxed{3}$ Boot a "HELLO-less" slave disk.

C600G

$\boxed{4}$ Get into the monitor and move the hello program back to $800.

CALL-151
800<4000.4400M
AF:25 0B *use the AF.B0 values noted in Step 2*
⊖ C *back to BASIC*

$\boxed{5}$ Save the hello program and the title screen.

SAVE HELLO
BSAVE SCRABTIT,A$2000,L$1FFF

$\boxed{6}$ Boot the Scrabble disk. When the Scrabble board appears, drop into the Monitor.

2000<B400.BEFFM

$\boxed{7}$ Boot the slave disk and capture the second part of the program.

C600G
BSAVE SCRABBLE.2, A$2000, L$63FF

$\boxed{8}$ Boot the Scrabble disk again. When the Scrabble board appears, drop into the Monitor again.

2000<800.1FFFM

$\boxed{9}$ Boot the slave disk, re-assemble, and BSAVE the program.

C600G
CALL-151
800<2000.37FFM
BLOAD SCRABBLE.2
BSAVE SCRABBLE, A$800, L$7BFF

$\boxed{10}$ Copy the four data (dictionary) files from the original disk to the new disk with FID or Copy II Plus.

The files are:
D-USA-1
D-USA-2
D-USA-3
D-USA-4

If the file "D-USA-3" won't copy, then it can be captured by the following method:

CALL -151
BLOAD SCRABBLE

Put the Scrabble disk into the drive.

300:A9 03 20 00 B4 60 N 7E44G N 300G

When the drive stops, you may have to hit RESET. Get back into the Monitor, and move the data to a safe place:

4000<8A00.A2FFM

Boot the slave disk, and:

C600G
BSAVE D-USA-3,A$4000,L$18DB

The HELLO file needs some editing. I'll explain that later.

Cracking Scrabble on a 128K IIe, IIc or IIgs with rev. 0 ROMs.

There are a couple of methods for booting a disk into auxiliary memory. One was originally published in COMPUTIST #16, in the article "Secret Weapon: Ramcard", by Ken Greenlaw. An elegant refinement of the XFER.BOOT and RESTORE techniques was published in COMPUTIST #58, in the RDEX article "Cracking on the IIe", by Zorro. That is the method I'll demonstrate.

[1] Boot a slave disk. Then enter the monitor and set up to boot the Scrabble disk into auxiliary memory.

PR#3 *switch in the 80 column firmware*
CALL-151
0: 8D 03 C0 8D 05 C0 4C 00 C6

[2] Insert the Scrabble disk in the drive and enter.

0G

[3] The drive will run for a while as the hello program and title screen are loaded. When the drive stops, hit RESET. Now, setup to move aux. memory to main memory.

CALL-151
300:18 4C 11 C3
3F8: 4C 00 03 *set up control-Y vector*
800<800.3FFF ⊖Y
AF<AF.B0 ⊖Y
⊖C
SAVE HELLO
BSAVE SCRABTIT, A$2000, L$1FFF

[4] Enter the monitor and set up to boot the Scrabble disk into auxiliary memory again.

PR#3
CALL-151
0: 8D 03 C0 8D 05 C0 4C 00 C6

[5] Insert the Scrabble disk in the drive and enter.

0G

[6] The drive will run for awhile as the hello program and title screen are loaded, then it will run again to load the Scrabble program. When the drive stops the second time, hit RESET. Now, move auxmem to main memory:

800<800.83FF ⊖Y
2000<B400.BEFF ⊖Y
BSAVE SCRABBLE,A$800,L$7BFF

[7] See step 10 from the II+ crack to capture the data files.

All the files you'll need are now on an unprotected disk, but a little more work is needed. I'll explain that after I detail how to deprotect Scrabble on a IIgs.

Cracking Scrabble on a IIgs with 01 ROMs or 00 ROMs and DiversiHack.

This is the easiest way to crack Scrabble, and probably the best.

If you have a IIgs with the 01 ROMs, there is a desk accesssory called "Visit Monitor", which is installed by getting into the monitor and entering: **# return**. If you then get into the control panel, you'll see that two new desk accessories have been installed.

If you have the 00 ROMs in your IIgs, you'll need DiversiHack or a similar desk accessory that will allow you to get into the monitor from the Control Panel.

[1] Boot Scrabble. When the drive stops, get into the control panel, and then into "Visit Monitor", and move main memory to a safe location.

1/800<0/800.4000M
0/AF.B0 *note these values*

[2] Put a slave disk in the drive and reboot. Get into the monitor and set up to save the hello program and title screen.

C600G
CALL-151
0/800<1/800.4000M
0/AF:25 0B *use the values noted in step 1*
⊖C
SAVE HELLO
BSAVE SCRABTIT, A$2000, L$1FFF

[3] Boot the Scrabble disk again. When the drive stops after displaying the title screen, get into "Visit Monitor" again.

800.B80 *display memory, look for BRUN, note its start address*
7F=F *set system mask to clear hi-bit*
B0F:"BLOAD" *change BRUN to BLOAD, if $B0F was start address of BRUN*
⊖Y *exit Control Panel to program*

[4] The HELLO program will continue executing, but it will BLOAD, instead of BRUN, Scrabble. When the drive stops, Scrabble is loaded. You may get a "Syntax Error" message, but it can be ignored. Go into "Visit Monitor" again and move Scrabble to bank 1.

1/800<0/800.83FFM

[5] Boot the slave disk, and then move Scrabble back to bank 0 memory.

C600G
CALL-151
0/800<1/800.83FFM
BSAVE SCRABBLE, A$800, L$7BFF

[6] Follow step 10 from the II+ section to get the data files.

Modifying the HELLO program

By now you should have seven files on your disk; HELLO, SCRABTIT, SCRABBLE, D-USA-1, D-USA-2, D-USA-3, D-USA-4.

The HELLO program needs to be changed to work with the deprotected files.

The files SCRABTIT and USA-SCRABBLE originally had ⊖C's as the second characters in their names. Copying over lines 30 and 80 in the HELLO program will remove the invisible control characters. If you have called the game file "SCRABBLE" instead of its original name "U ⊖C SA-SCRABBLE", change the name in line 80.

There are several delay loops in the HELLO program, in lines 46, 56, 67, and 73. Loading will go much faster if you shorten them or eliminate them altogether.

Don't forget to save HELLO after you're finished.

Fixing Scrabble's DOS

When this game was programmed, there must have been several people working on it. Unfortunatly, they didn't talk to one another to find out who was using what area in memory. As a result, there is a conflict (more than one, but only one that really counts) over the use of address $AD02. That is supposed to be the last track accessed, but it gets stepped on by a routine in the game.

To fix the DOS so it doesn't constantly recalibrate the disk drive, all you have to do is change the six references to $AD02 to a safe address, like $B302. That means only one byte needs to be changed in six locations. The disk will still recalibrate itself once at the start of the game unless you make an additional change.

To fix it:

BLOAD SCRABBLE
CALL-151
2258:B3
2417:B3
2422:B3
2430:B3
2437:B3
2463:B3
7FD:20 00 2A
2A00:AD EC B7 0A 8D 02 B3 60
BSAVE SCRABBLE, A$7FD, L$7C02

That's all there is to it. You should now have a deprotected version of Scrabble that runs quietly.

Converting Scrabble to ProDOS

The best way to deal with Scrabble's DOS is to get rid of it altogether. The only advantage to the original DOS is that you can run it on a 48K machine. (Who cares?!)

The advantage in converting to ProDOS is mainly speed. ProDOS can use the /RAM volume for additional speed and silence, and decreased wear of the disk drive. The following chart shows the relative clockings for Scrabble, when it is set so the computer will play against itself.

DOS 3.3 - approx. 57 min.
ProDOS (disk based) - approx. 26 min.
ProDOS (ram disk) - approx. 15 min.

Times are approximate because they vary due to the random selection of letters. If a "player" (the computer) gets a good selection of letters, "his" turn will go fast. The times still indicate a clear speed advantage for the ProDOS version. With a speedup card, or a IIgs set at fast speed, the ProDOS ram disk time drops to less than 5 minutes.

The conversion is pretty straightforward. First, transfer the following files to a ProDOS disk; D-USA-1, D-USA-2, D-USA-3, D-USA-4, HELLO, SCRABTIT, SCRABBLE.

The disk should be named /SCRABBL. During file transfer, the dashes (-) in the dictionary file names will be converted to periods (.). Copying the dictionary files over to the ProDOS disk first should make it easier for the game to access them. You can use a 5¼" or 3½" disk.

The Scrabble game needs a couple of modifications to work in the ProDOS environment. The following code needs to be installed.

The first listing is the ProDOS interface ("SDOS"), and the second listing ("SPATCH") is code to replace the original DOS relocate code and a bitmap clearing routine.

You can either enter the assembly listing and assemble the code, or enter the hex dump directly into memory. If you choose to enter the hex dump, save both files, the first as "**SDOS, A$2000, L$26C**", and the second as "**SPATCH, A$7E44, L$30**".

To install the new code:

BLOAD SCRABBLE
BLOAD SDOS,A$2000
BLOAD SPATCH
BSAVE SCRABBLE, A$800, L$7BFF

At this point you should have a working, disk based, ProDOS version of Scrabble. You may have to add a line to the "original" HELLO program to set the prefix to /SCRABBL.

To create a ram disk version, enter the Applesoft listing, and use it to start up the game. You need at least a 128K IIe or IIc for the ram disk version.

```
      1    * Scrabble ProDOS by James A. Hodge
      2
=00C03     Create   =   $C0
=00C84     Open     =   $C8
=00CA5     Read     =   $CA
=00CB6     Write    =   $CB
=00CC7     Close    =   $CC
=BF008     MLI      =   $BF00
      9
=080010    H0800    =   $0800
=080C11    H080C    =   $080C
     12
=403613    Prnt_mes=    $4036
=5C0614    Get_key  =   $5C06
=790C15    Save_D0  =   $790C
=790F16    Rest_D0  =   $790F
```

```
                     TR        ADR
                     ORG       $B400

          * The following 3 JMPs are ref'd in main pgm

B400:4C 17 B4                  JMP Get_Dict       Like the name?
B403:4C 42 B5                  JMP Saver          save-a-game
B406:4C 60 B4                  JMP ReStore        Restore-a-game

B409:00              OoopS     HEX 00
B40A:00              OoopS1    HEX 00
B40B:00              OoopS2    HEX 00
B40C:00              Rammit1   HEX 00
B40D:00              Rammit2   HEX 00
B40E:00              Rammit3   HEX 00
B40F:00              Rammit4   HEX 00
B410:00              Rammit5   HEX 00
B411:00              Nut       HEX 00
B412:00              Nutz      HEX 00
B413:00              Nutz1     HEX 00
B414:00              Nutz2     HEX 00
B415:00              Nutz3     HEX 00
B416:00              Nutz4     HEX 00

B417:18              Get_Dict  CLC
B418:69 30                     ADC #$30
B41A:8D 5F B4                  STA Dict_num
B41D:20 00 BF                  JSR MLI            Open the requested dict.
B420:C8                        DFB Open
B421:48 B4                     DA  D_OPEN
B423:90 03 =B428               BCC GoOn
B425:8D 09 B4                  STA OoopS
B428:AD 4D B4        GoOn      LDA D_OPEN+5
B42B:8D 4F B4                  STA D_READ+1
B42E:8D 57 B4                  STA D_CLOSE+1
B431:20 00 BF                  JSR MLI            Read it (pretty obvious)
B434:CA                        DFB Read
B435:4E B4                     DA  D_READ
B437:90 03 =B43C               BCC GoOn1
B439:8D 0A B4                  STA OoopS1
B43C:20 00 BF        GoOn1     JSR MLI            Guess what this does.
B43F:CC                        DFB Close
B440:56 B4                     DA  D_CLOSE
B442:90 03 =B447               BCC Gone
B444:8D 0B B4                  STA OoopS2
B447:60              Gone      RTS                Bye. Back to Scrabble

B448:03              D_OPEN    HEX 03             Parm lists for Get-Dict
B449:58 B4                     DA  Diclen
B44B:00 B7                     DA  $B700
B44D:00                        DS  1

B44E:04              D_READ    HEX 04
B44F:00                        DS  1
B450:00 8A                     DA  $8A00          dict. bufr
B452:00 1D                     DA  $1D00          requested len.
B454:00 00                     DS  2

B456:01              D_CLOSE   HEX 01
B457:00                        DS  1

B458:07              Diclen    DFB Dict_num-Diclen
B459:44 2E 55 53               ASC 'D.USA.'
B45D:41 2E
B45F:31              Dict_num  ASC '1'            changes to 1,2,3 or 4



B460:20 0C 79        ReStore   JSR Save_D0
B463:20 64 7E                  JSR $7E64          My clear.bitmap routine
B466:20 D3 B4                  JSR Restore1
B469:20 FC B4                  JSR Restore2
B46C:20 08 B5                  JSR Restore3
B46F:20 14 B5                  JSR Restore4
B472:20 00 BF                  JSR MLI
B475:CC                        DFB Close
B476:40 B5                     DA  R_CLOSE

B478:AD BF 8A                  LDA $8ABF          Restore pg. 0 values
B47B:85 BF                     STA $BF
B47D:AD EA 8A                  LDA $8AEA
B480:85 EA                     STA $EA
B482:AD 7B 8A                  LDA $8A7B
B485:85 7B                     STA $7B
B487:AD 7F 8A                  LDA $8A7F
B48A:85 7F                     STA $7F
B48C:AD 7C 8A                  LDA $8A7C
B48F:85 7C                     STA $7C
B491:AD CE 8A                  LDA $8ACE
B494:85 CE                     STA $CE
B496:AD BC 8A                  LDA $8ABC
B499:85 BC                     STA $BC
B49B:AD CF 8A                  LDA $8ACF
B49E:85 CF                     STA $CF
B4A0:AD 7A 8A                  LDA $8A7A
B4A3:85 7A                     STA $7A

B4A5:A5 BF                     LDA $BF
B4A7:C9 01                     CMP #$01
B4A9:F0 0F =B4BA               BEQ R1
B4AB:C9 02                     CMP #$02
B4AD:F0 0B =B4BA               BEQ R1
B4AF:C9 03                     CMP #$03
B4B1:F0 07 =B4BA               BEQ R1
B4B3:C9 04                     CMP #$04
B4B5:F0 03 =B4BA               BEQ R1
B4B7:4C 00 08                  JMP H0800

B4BA:A5 7B           R1        LDA $7B
B4BC:C9 00                     CMP #$00
B4BE:F0 0F =B4CF               BEQ R2
B4C0:C9 01                     CMP #$01
B4C2:F0 0B =B4CF               BEQ R2
B4C4:C9 02                     CMP #$02
B4C6:F0 07 =B4CF               BEQ R2
B4C8:C9 03                     CMP #$03
B4CA:F0 03 =B4CF               BEQ R2
B4CC:4C 00 08                  JMP H0800

B4CF:20 0F 79        R2        JSR Rest_D0
B4D2:60                        RTS

                     Restore1
B4D3:20 00 BF                  JSR MLI            Open the save file
B4D6:C8                        DFB Open
B4D7:5C B6                     DA  S_OPEN         Use the SAVE OPEN
                                                  parms
B4D9:90 03 =B4DE               BCC R3
B4DB:8D 0C B4                  STA Rammit1
B4DE:AD 61 B6        R3        LDA S_OPEN+5
B4E1:8D 21 B5                  STA R_READ1+1
B4E4:8D 29 B5                  STA R_READ2+1
B4E7:8D 31 B5                  STA R_READ3+1
B4EA:8D 39 B5                  STA R_READ4+1
B4ED:8D 41 B5                  STA R_CLOSE+1
B4F0:20 00 BF                  JSR MLI            Read $D00 bytes to $6E00
B4F3:CA                        DFB Read
B4F4:20 B5                     DA  R_READ1
B4F6:90 03 =B4FB               BCC R4
B4F8:8D 0D B4                  STA Rammit2
B4FB:60              R4        RTS

                     Restore2
B4FC:20 00 BF                  JSR MLI            Read $200 bytes to $5400
B4FF:CA                        DFB Read
B500:28 B5                     DA  R_READ2
B502:90 03 =B507               BCC R5
B504:8D 0E B4                  STA Rammit3
B507:60              R5        RTS

                     Restore3
B508:20 00 BF                  JSR MLI            Read $200 bytes to $5000
B50B:CA                        DFB Read
B50C:30 B5                     DA  R_READ3
B50E:90 03 =B513               BCC R6
B510:8D 0F B4                  STA Rammit4
B513:60              R6        RTS

                     Restore4
B514:20 00 BF                  JSR MLI            Read $200 bytes to $8A00
B517:CA                        DFB Read           contains pg. 0 values
B518:38 B5                     DA  R_READ4
B51A:90 03 =B51F               BCC R7
B51C:8D 10 B4                  STA Rammit5
B51F:60              R7        RTS

B520:04              R_READ1   HEX 04
B521:00                        DS  1
B522:00 6E                     DA  $6E00
B524:00 0D                     DA  $0D00
B526:00 00                     DS  2

B528:04              R_READ2   HEX 04
B529:00                        DS  1
B52A:00 54                     DA  $5400
B52C:00 02                     DA  $0200
B52E:00 00                     DS  2

B530:04              R_READ3   HEX 04
B531:00                        DS  1
B532:00 50                     DA  $5000
B534:00 02                     DA  $0200
B536:00 00                     DS  2

B538:04              R_READ4   HEX 04
B539:00                        DS  1
B53A:00 8A                     DA  $8A00
B53C:00 02                     DA  $0200
B53E:00 00                     DS  2

B540:01              R_CLOSE   HEX 01
B541:00                        DS  1

                     *

B542:20 0C 79        Saver     JSR Save_D0        Save a game
B545:A9 0A                     LDA #$0A
B547:85 80                     STA $80
B549:A9 00                     LDA #$00
B54B:85 81                     STA $81
B54D:20 0C 08                  JSR H080C
B550:20 36 40                  JSR Prnt_mes
B553:C9 CE D3 C5               ASC "INSERT BLANK DISK
                                   (PRODOS) AND HIT A KEY"
B557:D2 D4 A0 C2
B55B:CC C1 CE CB
B55F:A0 C4 C9 D3
B563:CB A0 A8 D0
B567:D2 CF C4 CF
B56B:D3 A9 A0 C1
B56F:CE C4 A0 C8
B573:C9 D4 A0 C1
B577:A0 CB C5 D9
B57B:00                        HEX 00

B57C:20 06 5C                  JSR Get_key
B57F:A9 0A                     LDA #$0A
B581:85 80                     STA $80
B583:A9 00                     LDA #$00
B585:85 81                     STA $81
B587:20 0C 08                  JSR H080C
B58A:20 36 40                  JSR Prnt_mes
B58D:C7 C1 CD C5               ASC "GAME SAVING TO S.GAME
                                   FILE"
B591:A0 D3 C1 D6
B595:C9 CE C7 A0
B599:D4 CF A0 D3
B59D:AE C7 C1 CD
B5A1:C5 A0 C6 C9
B5A5:CC C5
B5A7:00                        HEX 00

B5A8:20 00 BF        TryAgin   JSR MLI            Open the save file
B5AB:C8                        DFB Open
B5AC:5C B6                     DA  S_OPEN
B5AE:90 0E =B5BE               BCC GetItOn
B5B0:8D 11 B4                  STA Nut
B5B3:20 00 BF                  JSR MLI            If OPEN failed,
B5B6:C0                        DFB Create         file didn't exist
B5B7:50 B6                     DA  S_CREATE
B5B9:90 ED =B5A8               BCC TryAgin        Now try to open it
B5BB:8D 12 B4                  STA Nutz           If err, save won't work
B5BE:AD 61 B6        GetItOn   LDA S_OPEN+5
B5C1:8D 21 B5                  STA R_READ1+1
B5C4:8D 29 B5                  STA R_READ2+1
B5C7:8D 31 B5                  STA R_READ3+1
B5CA:8D 63 B6                  STA S_WRITE+1
B5CD:8D 6B B6                  STA S_CLOSE+1

B5D0:20 00 BF                  JSR MLI            Write $D00 bytes from $6E00
B5D3:CB                        DFB Write
```

```
B5D4: 20 B5                  DA  R_READ1
B5D6: 90 03 =B5DB            BCC S1
B5D8: 8D 13 B4               STA Nutz1

B5DB: 20 00 BF      S1       JSR MLI  Write $200 byts from $5400
B5DE: CB                     DFB Write
B5DF: 28 B5                  DA  R_READ2
B5E1: 90 03 =B5E6            BCC S2
B5E3: 8D 14 B4               STA Nutz2

B5E6: 20 00 BF      S2       JSR MLI  Write $200 bytes from $5000
B5E9: CB                     DFB Write
B5EA: 30 B5                  DA  R_READ3
B5EC: 90 03 =B5F1            BCC S3
B5EE: 8D 15 B4               STA Nutz3

B5F1: 20 00 BF      S3       JSR MLI  Write $100 bytes from $0000
B5F4: CB                     DFB Write
B5F5: 62 B6                  DA  S_WRITE
B5F7: 90 03 =B5FC            BCC S4
B5F9: 8D 16 B4               STA Nutz4

B5FC: 20 00 BF      S4       JSR MLI
B5FF: CC                     DFB Close
B600: 6A B6                  DA  S_CLOSE
B602: A9 0A                  LDA #$0A
B604: 85 80                  STA $80
B606: A9 00                  LDA #$00
B608: 85 81                  STA $81
B60A: 20 0C 08               JSR H080C
B60D: 20 36 40               JSR Prnt_mes
B610: D2 C5 AD C9            ASC "RE-INSERT SCRABBLE
                                  DISK AND HIT A KEY"
B614: CE D3 C5 D2
B618: D4 A0 D3 C3
B61C: D2 C1 C2 C2
B620: CC C5 A0 C4
B624: C9 D3 CB A0
B628: C1 CE C4 A0
B62C: C8 C9 D4 A0
B630: C1 A0 CB C5
B634: D9
B635: 00                     HEX 00

B636: 20 06 5C               JSR Get_key
B639: 20 0C 08               JSR H080C
B63C: 20 0F 79               JSR Rest_D0
B63F: 60                     RTS

B640: 0F            Savlen   DFB S5-Savlen
B641: 2F 53 43 52            ASC '/SCRABBL/S.GAME'
B645: 41 42 42 4C
B649: 2F 53 2E 47
B64D: 41 4D 45
          =B64F     S5       EQU *-1

B650: 07            S_CREATE HEX 07
B651: 40 B6                  DA  Savlen
B653: C3                     HEX C3      Access
B654: 06                     HEX 06      Ftype (bin.)
B655: 00 10                  DA  $1000   Aux. data (addr.)
B657: 01                     HEX 01      Store. type
B658: 00 00                  DS  2       Date
B65A: 00 00                  DS  2       Time

B65C: 03            S_OPEN   HEX 03
B65D: 40 B6                  DA  Savlen
B65F: 00 B7                  DA  $B700
B661: 00                     DS  1

B662: 04            S_WRITE  HEX 04
B663: 00                     DS  1
B664: 00 00                  DA  $0000
B666: 00 02                  DA  $0200
B668: 00 00                  DS  2

B66A: 01            S_CLOSE  HEX 01
B66B: 00                     DS  1
```

```
                    * Scrabble ProDOS relocate code and
                    * Bitmap clearing routine
                    * by James A. Hodge

                    TR       ADR
                    ORG      $7E44

7E44: A9 00                  LDA #$00      DOS relocater
7E46: 85 00                  STA $00
7E48: 85 02                  STA $02
7E4A: A9 B4                  LDA #$B4
7E4C: 85 03                  STA $03
7E4E: A9 20                  LDA #$20
7E50: 85 01                  STA $01
7E52: A2 04                  LDX #$04
7E54: A0 00                  LDY #$00
7E56: B1 00         L1       LDA ($00),Y
7E58: 91 02                  STA ($02),Y
7E5A: 88                     DEY
7E5B: D0 F9 =7E56            BNE L1
7E5D: E6 01                  INC $01
7E5F: E6 03                  INC $03
7E61: CA                     DEX
7E62: D0 F2 =7E56            BNE L1

7E64: A9 00         CLEARMAP LDA #$00      Bitmap clearing routine
7E66: A2 10                  LDX #$10
7E68: 9D 5F BF      L2       STA $BF5F,X
7E6B: CA                     DEX
7E6C: D0 FA =7E68            BNE L2
7E6E: A9 01                  LDA #$01
7E70: 8D 6F BF               STA $BF6F
7E73: 60                     RTS
```

Hexdump of Scrabble-ProDOS interface

```
2000: 4C 17 B4 4C 42 B5 4C 60   $5F17
2008: B4 00 00 00 00 00 00 00   $9527
2010: 00 00 00 00 00 00 00 18   $1D1B
2018: 69 30 8D 5F B4 20 00 BF   $7BCF
2020: C8 48 B4 90 03 8D 09 B4   $CF05
2028: AD 4D B4 8D 4F B4 8D 57   $234B
2030: B4 20 00 BF CA 4E B4 90   $591F
2038: 03 8D 0A B4 20 00 BF CC   $D59B
2040: 56 B4 90 03 8D 0B B4 60   $A8BE
2048: 03 58 B4 00 B7 00 04 00   $06CE
2050: 00 8A 00 1D 00 00 01 00   $AD24
2058: 07 44 2E 55 53 41 2E 31   $828A
2060: 20 0C 79 20 64 7E 20 D3   $6DEB
2068: B4 20 FC B4 20 08 B5 20   $8D94
2070: 14 B5 20 00 BF CC 40 B5   $8835
2078: AD BF 8A 85 BF AD EA 8A   $AE2F
2080: 85 EA AD 7B 8A 85 7B AD   $850E
2088: 7F 8A 85 7F AD 7C 8A 85   $933B
2090: 7C AD CE 8A 85 CE AD BC   $FB6D
2098: 8A 85 BC AD CF 8A 85 CF   $64A9
20A0: AD 7A 8A 85 7A A5 BF C9   $F2E6
20A8: 01 F0 0F C9 02 F0 0B C9   $1517
20B0: 03 F0 07 C9 04 F0 03 4C   $796C
20B8: 00 08 A5 7B C9 00 F0 0F   $9D5A
20C0: C9 01 F0 0B C9 02 F0 07   $94E3
20C8: C9 03 F0 03 4C 00 08 20   $04F4
20D0: 0F 79 60 20 00 BF C8 5C   $674C
20D8: B6 90 03 8D 0C B4 AD 61   $15DA
20E0: B6 8D 21 B5 8D 29 B5 8D   $8251
20E8: 31 B5 8D 39 B5 8D 41 B5   $7A67
20F0: 20 00 BF CA 20 B5 90 03   $031C
20F8: 8D 0D B4 60 20 00 BF CA   $2597
2100: 28 B5 90 03 8D 0E B4 60   $13D8
2108: 20 00 BF CA 30 B5 90 03   $9253
2110: 8D 0F B4 60 20 00 BF CA   $E439
2118: 38 B5 90 03 8D 10 B4 60   $72A5
2120: 04 00 00 6E 00 0D 00 00   $4B32
2128: 04 00 00 54 00 02 00 00   $8535
2130: 04 00 00 50 00 02 00 00   $EFA8
2138: 04 00 00 8A 00 02 00 00   $5FAC
2140: 01 00 20 0C 79 A9 0A 85   $0648
2148: 80 A9 00 85 81 20 0C 08   $F8C1
2150: 20 36 40 C9 CE D3 C5 D2   $E95D
2158: D4 A0 C2 CC C1 CE CB A0   $4D04
2160: C4 C9 D3 CB A0 A8 D0 D2   $00B1
2168: CF C4 CF D3 A9 A0 C1 CE   $258F
2170: C4 A0 C8 C9 D4 A0 C1 A0   $AE4A
2178: CB C5 D9 00 20 06 5C A9   $9391
2180: 0A 85 80 A9 00 85 81 20   $AD14
2188: 0C 08 20 36 40 C7 C1 CD   $0F3D
2190: C5 A0 D3 C1 D6 C9 CE C7   $DD20
2198: A0 D4 CF A0 D3 AE C7 C1   $8920
21A0: CD C5 A0 C6 C9 CC C5 00   $9C70
21A8: 20 00 BF C8 5C B6 90 0E   $815F
21B0: 8D 11 B4 20 00 BF C0 50   $3179
21B8: B6 90 ED 8D 12 B4 AD 61   $2789
21C0: B6 8D 21 B5 8D 29 B5 8D   $90F2
21C8: 31 B5 8D 63 B6 8D 6B B6   $ED5B
21D0: 20 00 BF CB 20 B5 90 03   $3168
21D8: 8D 13 B4 20 00 BF CB 28   $EA98
21E0: B5 90 03 8D 14 B4 20 00   $18C2
21E8: BF CB 30 B5 90 03 8D 15   $738E
21F0: B4 20 00 BF CB 62 B6 90   $3483
21F8: 03 8D 16 B4 20 00 BF CC   $469B
2200: 6A B6 A9 0A 85 80 A9 00   $7BC0
2208: 85 81 20 0C 08 20 36 40   $074B
2210: D2 C5 AD C9 CE D3 C5 D2   $EA45
2218: D4 A0 D3 C3 D2 C1 C2 C2   $1316
2220: CC C5 A0 C4 C9 D3 CB A0   $428D
2228: C1 CE C4 A0 C8 C9 D4 A0   $935D
2230: C1 A0 CB C5 D9 00 20 06   $87B7
2238: 5C 20 0C 08 20 0F 79 60   $54DF
2240: 0F 2F 53 43 52 41 42 42   $21E8
2248: 4C 2F 53 2E 47 41 4D 45   $18E1
2250: 07 40 B6 C3 06 00 10 01   $A163
2258: 00 00 00 00 03 40 B6 00   $5B08
2260: B7 00 04 00 00 00 00 02   $209C
2268: 00 00 01 00               $42ED
```

Hex dump of ProDOS relocate code

```
7E44: A9 00 85 00               $E086
7E48: 85 02 A9 B4 85 03 A9 20   $8962
7E50: 85 01 A2 04 A0 00 B1 00   $05A8
7E58: 91 02 88 D0 F9 E6 01 E6   $31C0
7E60: 03 CA D0 F2 A9 00 A2 10   $4336
7E68: 9D 5F BF CA D0 FA A9 01   $4CB0
7E70: 8D 6F BF 60               $F643
```

Hello program to run Scrabble with the /RAM disk

```
100 TEXT : HOME : HGR
110 PRINT CHR$ (4) "PREFIX^/SCRABBL"
115 PRINT CHR$ (4) "BLOAD^SCRABTIT"
120 PRINT : VTAB 22: PRINT TAB( 8) "'ALF^A^MO,^THERE,^
    FELLA"
130 PRINT CHR$ (4) "BLOAD^D.USA.1,A$4000"
140 PRINT CHR$ (4) "BSAVE^/RAM/D.USA.1,A$4000,L$17AE"
150 PRINT CHR$ (4) "BLOAD^D.USA.2,A$4000"
160 PRINT CHR$ (4) "BSAVE^/RAM/D.USA.2,A$4000,L$1614"
170 PRINT CHR$ (4) "BLOAD^D.USA.3,A$4000"
180 PRINT CHR$ (4) "BSAVE^/RAM/D.USA.3,A$4000,L$18DB"
190 PRINT CHR$ (4) "BLOAD^D.USA.4,A$4000"
200 PRINT CHR$ (4) "BSAVE^/RAM/D.USA.4,A$4000,L$18A1"
210 PRINT CHR$ (4) "PREFIX^/RAM"
220 TEXT : HOME : PRINT
230 VTAB 12: PRINT TAB( 4) "GET^READY.^HERE^COMES^
    SCRABBLE!"
240 PRINT CHR$ (4) "BRUN^/SCRABBL/SCRABBLE"
```

Checksums

100 - $0930	150 - $E54B	210 - $1FC6
110 - $48C6	160 - $4DCD	220 - $8C1B
115 - $10BC	170 - $5432	230 - $7AC1
120 - $3723	180 - $9B19	240 - $5F15
130 - $FCC5	190 - $5F4D	
140 - $4E5F	200 - $D7C6	

Softkey for...

Scruples

Leisure Genius

This game will be easy to de-protect if you've got an Apple IIgs with the 01 revision ROMs, otherwise, you're on your own. The softkey I wrote for Scrabble may give you some guidance for cracking Scruples on a non-IIgs.

There are four files needed from the Scruples disk:

```
A  3 HELLO
B 34 LOADER ; A$2000, L$200D
B 87 SCRUPLES ; A$1FA7, L$55E4
A  2 RUN
```

Capturing these files involves using the "Visit Monitor" IIgs desk accessory.

[1] Boot Scruples. When you see the message, "A P P L E S C R U P L E S", hit ⌘ esc and get into "Visit Monitor".

[2] Convert a CALL token to a STOP token.

932:B3

[3] Enter a ⊖Y and hit **return** 3 times. That takes you back to the game.

[4] When the game STOPs, get into the monitor through "Visit Monitor" and move the HELLO program to a safe location.

5000<800.A00M

[5] Boot a HELLOless slave disk with the command "C600G". Then, get into the monitor using "CALL -151" and move the HELLO program back down to $800.

800<5000.5200M
AF:78 9

[6] Save the Hello program and the binary Loader.

SAVE HELLO.THERE
BSAVE LOADER, A$2000, L$200D

[7] Boot Scruples again. When you see the message, "A P P L E S C R U P L E S", hit ⊖ ⊖esc and get into "Visit Monitor".

[8] Now convert the CALL token to a REM token and edit the "RUN RUN" command.

932:B2
7F=F
953:"BLOAD SCRUPLES,A$1FA7"
968:22 3A B3

[9] Press ⊖Y and hit **return** 3 times. That takes you back to the game. When the disk drive stops the SECOND time, Scruples should be loaded.

[10] Boot the slave disk again.

BSAVE SCRUPLES, A$1FA7, L$55E4

There are 2 ways to get the "RUN" file, the hard way and the easy way. The hard way is to use the technique used above. The easy way is to just enter the program.

10 TEXT : HOME
20 PRINT CHR$(4)"BRUN SCRUPLES,A$1FA7"
SAVE RUN

And that's all there is to it!

New routines for Super IOB

These routines won't make Super IOB any smarter or work any faster, but they make working with it a little more convenient.

Line 65 will save the controller as a text file. While I'm trying to develop a controller, I save the Super IOB as an Applesoft file, and it usually takes 17 or 18 sectors of disk space. When the controller finally works, I "RUN 65", save it as a text file, and it takes (on average) only 3 to 5 sectors of disk space.

The second routine automates the chore of making a lot of data statements for large disk patches.

It requires that the disk patch be in memory. I usually read the sector that I've patched in with the Copy II Plus sector editor, get into the monitor through the IIgs "Visit Monitor" desk accessory, and move the sector buffer somewhere safe. After I get out of the sector editor, I load in Super IOB, move the sector data into main memory, and "RUN 700".

There are seven variables that need to be set.

NL - Number of Lines of DATA statements
VL - number of Values on a Line
FD - First Data stmnt line number
AD - Address where the patch is located. The equation format in line 710 will convert a hex address to decimal if the 0s are replaced with the corresponding digit from a 4 digit hex address. For example, if the patch code is located at $234F, the equation would be "AD = 4096 * 2 + 256 * 3 + 16 * 4 + 15".
TR - Track number of sector to patch
S - Sector number to be patched
BA - Byte Address (within the sector) that patch starts at

You need to supply the appropriate values for NL and VL. For example, if you have 25 values to put in data statements and you want 4 values on a line, VL will equal 4 and NL will equal 7. When you copy over the data statements just don't copy over the last three sets of values on the last line.

This routine will print the data statements on the screen. You must copy over them to actually add them into the Super IOB. A line could be added to open a text file for writing. Then the statements could be EXECed into the program.

```
65 D$ = CHR$ (4): INPUT "SAVE FILENAME? ";F$: POKE 33,33:
   PRINT D$"OPEN "F$: PRINT D$"WRITE"F$: LIST
   1000,9999: PRINT D$"CLOSE": TEXT : END
700 REM MAKE DATA STMTS
705 NL = 0 : VL = 4
710 FD = 2000 : AD = 4096 * 0 + 256 * 0 + 16 * 0 + 0
715 TR = 0 : S = 0 : BA = 0
720 FOR I = 1 TO NL : PRINT FD + (I - 1) * 10 "DATA";
730 FOR J = 1 TO VL : PRINT TR","S","BA"," PEEK (AD)
   CHR$(44 - (VL = J) * 12);
735 BA = BA + 1 : AD = AD + 1
750 NEXT : PRINT : NEXT : END
```

B. Dudley Brett

Here are softkeys for two old clinkers, Magic Spells and Moptown of Special Delivery Software. These are 1981 vintage programs, probably not found in too many people's libraries.

However, the method I used to softkey may have application to some other programs. The method can be used with a DOS 3.3 disk with a catalog, but having a rewritten DOS so that the method of using a captured RWTS and swap controller or DeMuffin Plus will not work.

I cannot claim to have originated the approach; William Clarke in COMPUTIST #30 furnished the concept. I have made some changes in his method so that use of an Apple IIe is not completely necessary, and have tried to explain how to remove the frustration of necessary monitor move operations. The latter especially gave me some problems initially. I thought that my computer, an Apple IIe clone, was just not working properly. However, using monitor moves with two other computers, an Apple II+ and a true Apple IIe, indicated the problem was general.

Though I can attest to the validity of my approach to deprotecting programs such as these, I must admit to the fact that the method is time consuming, fraught with frustration (delightful for a masochist!) and to be attempted only as a last resort. There is a bit of instructional value in my accompanying article, though.

Softkey for...

Magic Spells
Moptown
Special Delivery Software

Recently I obtained these ancient releases of 1981 and decided to try to remove the copy-protection from them. Whereas both could be bit-copied, neither could be easily copied. Altered marks, changing from sector to sector, altered checksums, etc. suggested that the easy way of just fixing the read marks would be a tedious job.

I next tried to capture the RWTS, but quickly found that it was an extensively rewritten DOS 3.3 RWTS, with different entry point addresses. In short, I was unable to use a swap controller with this RWTS, as it simply was not compatible with the rest of DOS 3.3. Perhaps another approach was required! Fortunately for the cause of disk deprotection, (though, considering the extensive amount of fiddling required, perhaps not so elegant), I remembered an article by William Clarke (Deprotecting Millionaire, COMPUTIST #30, pp.12-15). Clarke's approach was to use auxilliary memory to trap images of both the copy-protected DOS and normal DOS, shuffling each into the correct place ($9600 to $BFFF) when required. Thus one could read programs from the protected disk with its own DOS, and then write the same program to a normal disk.

I promptly decided to try Clarke's method, not realizing the amount of time that would be required, nor the frustration ahead as something would not work as expected. After this experience, I have come up with a workable method, with only a few frustrations left, but still requiring more than the usual time of a normal COMPUTIST softkey. This, therefore, is an approach to be tried with some trepidation, and only as a last resort.

First let me relate some of the frustrations encountered. William Clarke suggested storing both the protected DOS and normal DOS in auxilliary memory, then making a memory move through the control-Y vector to a move routine in page $03. Unfortunately, I found that sometimes it would work and sometimes my Apple IIe would just "hang". If I pressed ⊖reset to stop the nonsense, I usually had a devil of a time restoring things, so that I could complete the operation. I was continually turning the computer on and off, and getting more frustrated. But, I then realized that both "DOS's" did not have to be stored in auxiliary memory, as any file on the protected disk could simply be loaded low in memory, allowing the 2 DOS routines to reside from $4000 to $93FF without mishap. The computer still would "hang", but not beyond recovery. With this major source of frustration out of the way, I could and did deprotect the two disks. Here now is the method.

Strategy

The entire transfer of all files from the protected disk to a normal DOS 3.3 disk (preferably with a fast DOS) involves a number of steps. These are:

- Initiallize a DOS 3.3 disk (for each of the 2 programs).
- Capture the DOS from one of the disks (I used MAGIC SPELLS) and BSAVE it on a normal disk.
- Transfer Applesoft files to one of the initiallized disks (This requires both DOS files).
- Using SDS.DOS (the captured DOS), BLOAD each binary file into memory, and find address and length (a memory search is needed as the normal DOS 3.3 pointers won't work).
- Transfer binary files to an initiallized disk (using the addresses and lengths previously found).
- Using a special Applesoft transfer program and binary memory move routine, transfer all text files to the initiallized disk.
- Make one small edit to the HELLO program to remove the reset trap.

Trap Magic Spell DOS

To capture the DOS from Magic Spells or Moptown a method of bypassing the reset trap used in both these programs must be found. If one has some method of resetting into the monitor, the procedure is simple. If not, Ken Greenlaw's (COMPUTIST #16) method requiring an Apple IIe and extended memory can be used. This follows:

[1] Initiallize, with a fast DOS 3.3, one disk for each program which is being deprotected.

[2] Insert the disk containing your COMPUTIST cracking tools.

BLOAD XFER.BOOT *Greenlaw's routine*
PR#3

[3] Put the original disk in drive #1.

CALL768
⊖reset *after the drive stops*

[4] Put your cracking disk in drive 1.

BLOAD RESTORE *memory move routine*
CALL-151
3F8:4C 00 03 *set control-Y vector*
4000<9600.BFFF⊖Y *Move mem from aux to main*
BSAVE SDS.DOS, A$4000, L$2A00

Save the Applesoft Files

To transfer all Applesoft files from the protected disk, the technique is to move the SDS.DOS in place of normal DOS 3.3, load a file, replace normal DOS and write to the initialized disk. The process is repeated for each file. I did experience one minor problem making memory moves in the monitor while trying to exchange the operating DOS. Whenever I first attempted a move, such as `9600<4000.69FFM`, my computer would hang. I found that pressing **reset** and repeating the move was successful. I am not sure if this is just a quirk of my computer, and will include this double move in my instructions just in case.

[1] Put in your cracking disk.

```
BLOAD SDS.DOS, A$4000
CALL-151
6A00<9600.BFFFM          move DOS 3.3 copy into low memory
```

[2] Set things up to read the protected disk.

```
9600<4000.69FFM          move SDS.DOS up into command space
⊖ reset                  do this only if computer hangs
CALL-151
9600<4000.69FFM          repeat the move
⊖ C                      get back to BASIC
```

[3] Place original disk in drive 1.

```
LOAD program,D1          Either CATALOG first or consult the list
                         included in this article
CALL-151
9600<6A00.93FFM          move DOS 3.3 up into command space
⊖ reset                  if computer hangs
CALL-151
9600<6A00.93FFM          repeated move
```

[4] Place initialized disk in drive (preferably drive 2).

```
⊖ C
SAVE program,D2          use the same name as in step 3
```

Repeat steps 2-4 for each Applesoft file. I would suggest write protecting the original disk so as to avoid tragic mistakes.

The Applesoft files to save are:

Magic Spells
- HELLO
- MENU2
- MSHR4
- WRITER3

Moptown
- HELLO
- MENU
- MAKE MY TWIN
- WHO'S DIFFERENT?
- WHAT'S THE SAME?
- WHO COMES NEXT?
- MOPTOWN PARADE
- WHO'S NEXT DOOR?
- SECRET PAL
- CHANGE ME
- CLUBHOUSE
- MOPTOWN MAP
- MOPTOWN HOTEL

Finding the Binary Files

In order to transfer protected binary files to your initialized disk, their beginning address and length must be known prior to any BSAVE. If the SDS.DOS were more similar to DOS 3.3, this would be an easy operation, as one could simply BLOAD the file, and read locations \$AA72.AA73 for the address and \$AA60.AA61 for length. These pointers, however, could not be found in SDS.DOS, and thus a more involved technique is required. After SDS.DOS has been moved into operation, memory on page \$03 and from \$800 to \$9500 must be cleared to zeros before any file is BLOADed. Then a scan through memory is made using ⊖ S to stop the scroll of memory so as to read file start and end addresses. This process can be shown by a simple example.

[1] Boot DOS 3.3 and put your cracking disk in drive 1.

```
BLOAD SDS.DOS, A$4000
CALL-151
9600<4000.69FFM          move SDS.DOS into control
⊖ reset                  if computer hangs only!
CALL-151                 reenter monitor after reset
9600<4000.69FFM          repeat move if necessary
CATALOG                  write down binary file names
```

[2] Clear memory.

```
300:00 N 301<300.3C0M    write 00's in page 3
800:00 N 801<800.9500M   write 00's to rest of RAM
BLOAD binary program
800.9500                 start a scroll down memory lane
⊖ S                      when 00's change to file start
```

Start the scroll again, after writing down file start address, stopping it when 00's reappear to find file end. Repeat step 2 to locate all other binary files.

Sometimes it is safer to use an odd byte value to fill memory, such as \$11. If a binary file is BLOADed, instead of BRUN, it may contain leading or trailing \$00s. RDEXed

What if the binary file cannot be found from \$800 to \$9500? First search page \$03 (300LL etc.). Some files are also BLOADed into Low Res graphics (\$400.7FF). If the monitor screen fills with garbage, you know that its address is A\$400 and its length is \$400 bytes. Fortunately, it doesn't matter too much where you eventually BSAVE this picture file, as the controlling Applesoft program states the BLOAD address.

BSAVEing the Binary Files

The binary files may be saved using a similar technique to that employed to capture Applesoft files. An example follows.

[1] Put in your cracking disk.

```
BLOAD SDS.DOS,A$4000
CALL-151
6A00<9600.BFFFM          snatch a copy of DOS 3.3
```

[2] Put original disk in drive 1 and initialized disk in drive 2.

```
9600<4000.69FFM          put SDS.DOS into control
⊖ reset                  if computer hangs
CALL-151                 to reenter monitor
9600<4000.69FFM          again

BLOAD binary file, A$xxxx, D1  use another address if necessary
                         See note
9600<6A00.93FFM          DOS 3.3 back into control
⊖ reset                  here we go again!
CALL-151
9600<6A00.93FFM
zzzz<xxxx.yyyyM          move any relocated files to proper place, if
                         necessary - see note
BSAVE binary file, A$xxxx, L$yyyy, D2
```

Repeat step 2 for all other binary files. If a file must be saved above \$4000, a reboot (step 1) will be necessary.

Note: *For any file loading to pages \$04 to \$7FF, or a file loading above \$6A00, the file must be loaded somewhere between \$800 and \$6A00, so that normal DOS 3.3 may be used to BSAVE the file. After DOS 3.3 has been relocated into control, the file can be moved back by a monitor move. For any file loaded (A\$400, L\$400) to the low res area, it is not necessary for a move back, as I have found out that a proper address, already present in the running Applesoft program, will ensure that it is BLOADed properly. One additional file, RAMLOADER, does not have to be copied, as it is simply a DOS relocation program and never is called upon by any file, nor is it really needed.*

The binary files to BSAVE are:

Magic Spells
- SPELLSPICS, A\$800, L\$1800
- PRINTER6+PICS+ERR, A\$4400, L\$1963
- MUSICRESETMOVE, A\$300, L\$48
- MEADOW, A\$2000, L\$2000
- KINGDOM OF SPELLS.LPIC, A\$800, L\$400

Moptown
- HOME SWEET HOME.LPIC, A\$1400, 400
- MOPTOWN PARADE.LPIC, A\$1400, L\$400
- PIC.MAPCRAM, A\$4000, L\$12F0
- PIC.HOTELCRAM, A\$4000, L\$1B48
- PIC.MOPTOWNCRAM, A\$4000, L\$17D8
- PRINTER12\$800CRAM, A\$4000, L\$E68
- MUSICRESETMOVESHAPE, A\$300, L\$B0
- UNSCRAM.ERR, A\$9240, L\$3C0

Reading and Writing Text Files

As all text files (found only in Magic Spells) are sequential, rather than random, no complicated approach as in William Clarke's softkey is necessary. Each text file, when read, inputs as the first entry, the number of records. Thus, one can simply determine the record number and then input the following records as strings. Upon moving normal DOS 3.3 back into control, the records can be written to our initialized disk. In order to facilitate this operation, the following Applesoft program and accompanying binary move routine is the simplest approach. Just type both programs in and store them on your cracking disk where the SDS.DOS file is already found.

Note that the Applesoft program will first BLOAD the SDS.DOS file and a relocation routine, MVE. Then prompts are made to place original and copy disk (your initiallized disk) in the correct drives and the name of the text file is requested. Using the accompanying list of text files or names from the Magic Spells catalog, enter an appropriate file name. The program then will, using the MVE subroutine, place the correct SDS.DOS in place of DOS 3.3 (line 160) and read in the text file. Replacing DOS 3.3 (line 200) is followed by writing the file to drive 2. The program then exits (line 260). One can then type RUN and repeat the procedure for another text file.

Text Exchanger

```
10 REM TEXT.EXCHANGER
20 TEXT : HOME : IF PEEK (810) < > 0 THEN 90
30 D$ = CHR$ (4)
40 VTAB 4: PRINT "PLACE^CRACKER^DISK^IN^DRIVE^1"
50 PRINT "AND^PRESS^<RETURN>^:^" ;: GET Z$: PRINT
60 PRINT D$; "BLOAD^ MVE" : PRINT D$; "BLOAD^
   SDS.DOS,A$3000"
70 POKE 810,1
80 POKE 773,150: POKE 781,191: POKE 789,90: CALL 768:
   REM MOVE DOS 3.3 DOWN
90 CLEAR
100 DIM FI$(30):D$ = CHR$ (4)
110 TEXT : HOME
120 VTAB 4: PRINT "PLACE^MAGIC.SPELLS^IN^DRIVE^1^AND"
130 PRINT "DOS^3.3^COPY^DISK^IN^DRIVE^2"
140 PRINT : PRINT "PRESS^RETURN^WHEN^DONE^:^" ;: GET Z$:
    PRINT
150 TEXT : HOME : VTAB 4: INPUT "FILE^NAME^:^" ;FF$
160 POKE 773,48: POKE 781,89: POKE 789,150: CALL 768:
    REM MOVE SDS.DOS UP
170 PRINT : PRINT D$; "OPEN^" ;FF$;; ",D1" : PRINT D$;
    "READ^" ;FF$
180 INPUT NF: FOR I = 1 TO NF: INPUT FI$(I): NEXT
190 PRINT D$; "CLOSE^" ;FF$
200 POKE 773,90: POKE 781,131: CALL 768: REM MOVE DOS 3.3
    UP
210 PRINT : PRINT D$; "OPEN^" ;FF$; ",D2" : PRINT D$;
    "WRITE^" ;FF$
220 PRINT NF: FOR I = 1 TO NF
230 PRINT FI$(I): NEXT I
240 PRINT D$; "CLOSE^" FF$
250 TEXT : HOME
260 PRINT "FINISHED" : END
```

Checksums

10 - \$BADD	100 - \$E72B	190 - \$F495
20 - \$5EEC	110 - \$4CEF	200 - \$105A
30 - \$30D6	120 - \$BD0F	210 - \$2B7D
40 - \$D611	130 - \$958D	220 - \$4480
50 - \$73B6	140 - \$EE78	230 - \$312C
60 - \$2AC3	150 - \$2AE6	240 - \$DAFF
70 - \$90BB	160 - \$B995	250 - \$8694
80 - \$02EC	170 - \$A292	260 - \$B2E6
90 - \$9995	180 - \$C24E	

Hexdump of MVE

```
0300: A9 00 85 3C A9 40 85 3D        $116B
0308: A9 FF 85 3E A9 69 85 3F        $043C
0310: A9 00 85 42 A9 B6 85 43        $EDFF
0318: 20 4A FF A9 00 85 47 20        $A522
0320: 3F FF 20 2C FF 60              $3651
```

MVE Source code

```
0300- A9 00     LDA #$00      (low byte origin)
0302- 85 3C     STA $3C
0304- A9 40     LDA #$40      high byte origin
0306- 85 3D     STA $3D
0308- A9 FF     LDA #$FF      low end byte origin
030A- 85 3E     STA $3E
030C- A9 69     LDA #$69      high end byte origin
030E- 85 3F     STA $3F
0310- A9 00     LDA #$00      low byte destination
0312- 85 42     STA $42
0314- A9 B6     LDA #$B6      high byte destination
0316- 85 43     STA $43
0318- 20 4A FF  JSR $FF4A     start the move
031B- A9 00     LDA #$00
031D- 85 47     STA $47
```

```
031F- 20 3F FF  JSR $FF3F
0322- 20 2C FF  JSR $FF2C
0325- 60        RTS          return to applesoft prog.
```

Here are the text files to be transferred:

Magic Spells

LOG
DEMONS 1.MS
DEMONS 2.MS
DEMONS 3.MS
DEMONS 4.MS
DEMONS 5.MS
DEMONS 6.MS
DEMONS 7.MS
DEMONS 8.MS
DEMONS 9.MS
DEMONS 10.MS
DEMONS 11.MS
SAMPLE.MS
SAMPLE 1.MS
Q.MS
C1982.MS
O1982.MS
W1982.MS

Last Step - Loosen the Reset Trap

Now that all files, Applesoft, binary and text have been transferred, there is one small job to complete. I have never liked the commonly used reset trap in protected programs. Let's turn it off! Here's how:

1 Boot normal DOS 3.3. Place either deprotected disk in drive 1.

LOAD HELLO
LIST10

2 Edit this line, removing the commands: **POKE 1010,23: POKE 1011,3:** (keep everything else, though).

4 Save the changed program.

UNLOCK HELLO *if necessary*
SAVE HELLO

MOST WANTED
Softkeys

Airheart	*Broderbund*
Alcon	*Taito*
Alien Mind	*PBI Software*
Ancient Art of War at Sea	*Broderbund*
Arkanoid	*Taito*
Bad Street Brawler	*Mindscape*
Beyond Zork	*Infocom*
Bilestoad	*Datamost*
Border Zone	*Infocom*
Borg	*Sirius*
Bubble Bobble	*Taito*
Bubble Ghost IIgs	*Accolade*
Bureaucracy	*Infocom*
California Games (IIgs)	*Epyx*
Chuck Yeager's Advanced Flight Simulator	*?*
Cosmic Relief	*Datasoft*
Crime & Punishment	*Imagic*
Darklord	*Datasoft*
Design Your Own Train	*Abracadata*
Dondra	*Spectrum Holobyte*
Dungeon Masters Assistant vol. I:Encounter	*SSI*
DROL	*Broderbund*
Eliminator	*Adventure International*
Explore-Australia	*Dataflow Computer Service*
Frogger	*Main Street*
The Games: Winter Edition	*Epyx*
Gladiator	*Taito*
Gutenbery Sr.	*Micromation LTD.*
Halls of Montezuma	*Electronic Arts*
Ice Demons	*Morningstar*
Impossible Mission II	*Epyx*
Indoor Sports	*Mindscape*
Joker Poker	*Mindscape*
King of Chicago	*Cineware*
L.A. Crackdown	*EPYX*
Lost Tomb	*Datasoft*
Manhunter New York IIgs	*Sierra On Line.*
Mavis Beacon Teaches Typing	*Software Toolworks*
Modem MGR	*MGR software*
National Inspirer	*Tom Snyder Productions*
Observatory (The)	*Mindscape*
Operation Wolf	*Taito*
Pirates!	*Microprose*
Platoon	*Data East*
Pool of Radiance	*SSI*
Quadratic Equations II	*Olympus Educational Software*
Questron II	*Electronic Arts*
Rastan	*Taito*
Renegade	*Taito*
Rocket Ranger (IIgs)	*Cinemaware*
S.D.I. (IIgs)	*Cinemaware*
Sea Stalker	*Broderbund*
Sky Shark	*Taito*
Soko-Ban	*Spectrum Holobyte*
Sound Song & Vision	*Advanced Software*
Spare Change	*Broderbund*
Speedy Spides	*Readers Digest*
StickyBear Math: Add & Subtract	*?*
Strike Fleet	*Electronic Arts*
Superstar Ice Hockey	*Mindscape*
Superstar Indoor Sports	*Mindscape*
Teleport	*Cavalier*
Test Drive	*Accolade*
The Three Stooges (IIgs)	*Cinemaware*
Thunder Chopper	*?*
Ticket to Washington D.C.	*Blue Lion Software*
Tomahawk	*Electronic Arts*
Tomahawk (IIgs)	*Datasoft*
VCR Companion	*Broderbund*
Wasteland	*Electronic Arts*
Wings of Fury	*Broderbund*
Wizardry:Return of Werda	*Sir-Tech.*
Works (the)	*First Star Software*
ZorkQuest	*Infocom*

Mike Basford

Softkey for...

Reader Rabbit
Learning Company

Using PCTools or a similar program, change "HELLO.EXE". Search for CD 13 72 09 and change it to 90 90 EB 24. What this does is bypass the disk check (CD 13) and replaces the JB 2592 with JMP 25AD.

Irwin Roth

In COMPUTIST #61, pg.44, a contributing author, Mike Basford, mentions a program called "CRACKER" which I have been unable to find. Please put me in contact with Mr. Basford or advise where or how I may attain this program.

Say Mike, could you handle this one for me? RDEXed

Ann Onymous

Sorry to be so confused. Am I supposed to be submitting this on an Apple or IBM formatted textfile?

Until we get an IBM of our own, an Apple disk would be nice. We send the IBM formatted disks to a local reader who translates the files for us. . . . . RDEXed

Softkey for...

Print Shop
Broderbund

The Printshop program lets you install it on a hard disk, and will run thereafter without needing the floppy. The installation program, however, keeps track of its use, and will only allow four hard disk installations.

Here's a way to trick it.

Make a subdirectory on your hard disk.

MD C:\PS

Run the installation program.

PSHD

Follow the prompts, and tell it to install Printshop in the C:\PS subdirectory. The program will terminate and leave you at the A> prompt.

Run the installation program again

PSHD

and select the "Remove from Hard Disk" option. The program will terminate by letting you know that it has increased your quota for hard disk installations.

Now run PCTools or some similar program and UNDELETE all the files in the C:\PS subdirectory. With PCTools you should choose automatic undeletion. IBM filenames lose their first character when they are deleted. You will have to supply the lost letter of each filename - make them match the names on the floppy.

Finally, use PCTools to undelete the hidden file C:\HPSDISK.SYS in the root directory of your hard disk.

The whole process takes less than 10 minutes. You now have a working copy of Printshop on your hard disk, and have not "used up" one of the installations from your floppy.

The A.S.P.

Softkey for...

Xenocopy Plus v1.09
?

If you have a hard disk or want to create a backup copy that is not tied into the Xenocopy diskette, this patch will partially remove the copy protection.

1 Place the Xenocopy original disk in the 'A' drive.

```
C:DEBUG XENOCOPY.COM
E 2EE B0 77 0C 02 90 90
E 312 33 16 0E 03 81 F2 33 96 B0 77 30 C2 81 F2 E4 21 31
    CA 31 E2 31 FA 30 F2 30 D3 88 D8 88 E3 53 AA 42 E2
    D3 CC
G
```

2 When DEBUG stops at CS:335, you next enter:

```
E 337 90
E 339 90 90
F 308 335 90
F 3DE 3E6 90          This seems to negate copy protect checks
E 308 B8 FC 78 BB 78 00 B9 00 00 BE 35 03 BF 35 03
R AX 0
R BX 0
R CX 0
R DX 0
R CX A96
N C:XENO.COM
W CS:100
Q
```

'XENO.COM' may now be copied to XENOCOPY .COM or left on the hard disk. Repeat steps 1 and 2 for 'XENOWRIT.COM', substituting for 'XENOCOPY .COM' where necessary, also 'XENO.COM' needs to be a new name such as 'WRIT.COM'.

Notes:

1) The February version of "Copywrit" will make a good complete backup copy of the "Xenocopy Plus" disk.

2) Version 1.16 of "CopyIIPC" will make a backup copy but loses some of the copy protection in the process. The above mods will allow for the lost copy protection tracks.

3) The main reason for this work was that "Xenocopy" would not come up using a "Teac" half-height drive, but will (seems to) after the mods.

4). These mods make all the code decrypted and available for inspection, so some sharp person can pursue the complete removal of the copy protection.

(4a) The two ".EXE" files are the programs loaded from "XENOCOPY.COM" and "XENOWRIT.COM". They seem to be written as 512 byte sectors, with sector IDs "119 to 126", so if they could be read and copied to 512 byte "1 to 8" sectors and the "Xeno" programs modified to read the new format files, all the copy protection would (just guessing) be removed.

5). When you do the backup option, there is a hidden file "BACKUP.COM" that is used.

6). As always, these unprotect procedures are tested as much as time will permit, an extensive test of modified versions are left up to the user.

7). The public domain "Ultra" utilities help in analyzing the disks.

Enjoy your new found freedom, hard disks forever!

UNK

Softkey for...

Managing Your Money
Andrew Tobias

Those people who have Andrew Tobias' "Managing Your Money" (MYM) and a hard disk find it very inconvenient to put disk 1 in drive A to start up MYM. Below is an update to MYM version 1.5, for people with Hard Disks, to bypass this problem. If you have only floppy disks drives, this procedure won't help much.

1 Make sure you run INSTALL and specify MYM is to be run using a Hard Disk.

2 Make a copy of MYM.EXE using the following DOS command making sure that MYM.EXE is on your Hard Disk.

COPY MYM.EXE MYM.PGM /V

3 Run DEBUG against MYM.PGM as follows with your input to DEBUG shown in bold.

```
DEBUG MYM.PGM
D 4709 470D
XXXX:4709 B8 01 02 CD 13
D 4744 4745
XXXX:4744 75 CF
D EFD0 EFDF
XXXX:EFD0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00
    00
R CX
CX EFD0
:                                        press return
```

4 If the above 4 displays do not show the same output, you are probability at a different level of MYM and will have to figure out a corresponding update using data from above and below.

```
A 4709
XXXX:4709 JMP EFD0
XXXX:470C XOR AX,AX
XXXX:470E                                press return
A 4744
XXXX:4744 NOP
XXXX:4745 NOP
XXXX:4746                                press return
A EFD0
XXXX:EFD0 PUSH BX
XXXX:EFD1 MOV AX,1453
XXXX:EFD4 MOV [BX],AL
XXXX:EFD6 INC BX
XXXX:EFD7 DEC AH
XXXX:EFD9 JNZ EFD4
XXXX:EFDB POP BX
XXXX:EFDC JMP 470C
XXXX:EFDF                                press return
R CX
CX EFD0
:EFE0
W                                        Writing EFE0 bytes
Q
RENAME MYM.EXE MYM.BAK
RENAME MYM.PGM MYM.EXE
```

MYM.BAK is now your unmodified, original MYM program. Save it, in case the new version does not work. You should no longer need to put disk 1 in drive A.

Note: If you run the updated version of MYM.EXE from a floppy disk, you will be prompted to put Disk 1, 2 or 3 in other drives. This new version must be installed using INSTALL specifying a Hard Disk and must executed from a Hard Disk to work.

Richard Cole

Softkey for...

Managing Your Money v1.5
Andrew Tobias

These instructions will direct you how to change Manage Your Money Version 1.5 so that it will run on your hard disk without requiring a system disk in your floppy drive. Naturally these changes are meant to be used by licensed users only.

This file represents several hours of stepping through compiled BASIC (yechhh a fate worse than death) in order to remove the offending code. If you find this useful please send $10.00 to:

Richard Cole
1071 Wilson Drive
Simi Valley, CA 93065

Run setup first to set up MYM for your system. Then run all three disks onto your hard disk. Then use Debug as instructed on MYM.EXE. From then on you can run MYM directly from your hard disk with no system disk.

You can also use backup or copy to save MYM and its files to a floppy disk. This allows you to make extra backups etc. You must remember that you have to use SETUP.EXE on the original system disk to change the system setup as to number or drives, where to look for data files, screen type, etc.

This was tested on DOS 3.0 using DEBUG 3.0. For earlier versions of DOS and DEBUG you might have to look up or down in memory one or two sectors to find the proper locations.

Make sure to operate on a backup file and not the original. I suggest you set up MYM for your system first from the original disks and then run them on your hard disk. Then make a copy of MYM.EXE and use that to operate on. When you have it working copy it back to your hard disk and you're in business.

At location $04F0 you will find CALL 44F2 or byte string E8 FF 3F. Change these three bytes to NOP NOP NOP or byte string 90 90 90. This removes the call to the floppy disk.

At location $72DE you will find CALL 4848 or byte string E8 67 D5. Change these three bytes to NOP NOP NOP or byte string 90 90 90. This removes those nasty little checks for the system disk.

the COMPUTIST shopper

Software Package	IIe	IIgs
816 Paint	$45 ☐	$45 ☐
Alternate Reality: The Dungeon	$28 ☐	
Alternate Reality: The City	$20 ☐	
Appleworks	$190 ☐	
Artic Fox	$28 ☐	
Bank Street Writer 64K	$46 ☐	
Bank Street Writer Plus (128K)	$46 ☐	
Bard's Tale	$30 ☐	$35 ☐
Bard's Tale II	$35 ☐	
Bard's Tale III	$35 ☐	
California Games	$26 ☐	$26 ☐
Certificate Maker	$25 ☐	
Championship Karate	$14 ☐	
Chessmaster 2000	$28 ☐	
Clip Art Library (for Paintworks Plus)		$20 ☐
Copy II Plus v8.0	$23 ☐	$23 ☐
Create with Garfield	$20 ☐	
Create w/ Garfield Deluxe	$28 ☐	
Crossword Magic	$32 ☐	
Dark Lord	$15 ☐	
Dazzle Draw	$40 ☐	
Draw Plus	$52 ☐	
F-15 Strike Eagle	$24 ☐	
Fantavision	$34 ☐	$40 ☐
Flight Simulator II	$36 ☐	
Force 7	$15 ☐	
Halls of Montezuma	$28 ☐	$28 ☐
Hardball	$22 ☐	$28 ☐
Hitchhiker's Guide	$20 ☐	
Jet	$29 ☐	
Karateka	$24 ☐	
King's Quest	$32 ☐	$32 ☐
King's Quest II	$32 ☐	$32 ☐
King's Quest III	$32 ☐	$32 ☐
King's Quest IV		$32 ☐
Knight of Diamonds	$23 ☐	
Legacy of the Ancients	$28 ☐	
Leisure Suit Larry	$26 ☐	$26 ☐
Lode Runner	$24 ☐	
Macroworks	$20 ☐	
Math Blaster	$28 ☐	
Math Blaster Plus	$29 ☐	$29 ☐
Math Rabbit	$27 ☐	
Math Talk		$35 ☐
Marble Madness	$25 ☐	$25 ☐
Mean 18, Ultimate Golf	$28 ☐	
Merlin 8/16	$80 ☐	
Millionaire II	$40 ☐	
Mousewrite	$99 ☐	$99 ☐
Multiscribe 3.0	$48 ☐	$66 ☐
Music Studio 2.0	$52 ☐	
Newsroom	$38 ☐	
Paint Write Draw		$120 ☐
Paintworks Plus		$47 ☐
Paintworks Gold		$65 ☐
Pegasus	$23 ☐	
The Print Shop	$34 ☐	$40 ☐
Print Shop Companion	$27 ☐	
Print Shop Graphics Library:		
Disk One	$16 ☐	
Disk Two	$16 ☐	
Disk Three	$16 ☐	
Pro-Byter	$32 ☐	
Reader Rabbit	$27 ☐	$34 ☐
Silent Service	$23 ☐	
Space Quest	$32 ☐	$32 ☐
Space Quest II	$32 ☐	
Star Fleet	$35 ☐	
Star Trek- The Kobayashi Alternative	$27 ☐	
Star Trek II- The Promethean Prophecy	$27 ☐	
Stickybear Series:		
ABC's	$24 ☐	
Math I	$24 ☐	
Math II	$24 ☐	
Numbers	$24 ☐	
Reading	$24 ☐	
Shapes	$24 ☐	
Typing	$24 ☐	
Opposites	$24 ☐	
Thexder	$23 ☐	$23 ☐
TimeOut Superfonts	$42 ☐	
Tomahawk	$23 ☐	
Topdraw	$66 ☐	
Type	$30 ☐	
Typing Tutor IV	$32 ☐	$32 ☐
Ultima V	$40 ☐	
Visualizer	$53 ☐	$59 ☐
Where in USA is Carmen San Diego	$30 ☐	
Where in World is Carmen San Diego	$30 ☐	
Where in Europe is Carmen San Diego	$30 ☐	
Wings of Fury	$25 ☐	
Winter Games	$25 ☐	$27 ☐
Wizardy	$32 ☐	
Word Attack	$28 ☐	
Wordperfect w/ Spelling Checker-	$95 ☐	$95 ☐
World Games	$25 ☐	$27 ☐
Writer's Choice Elite	$60 ☐	
Writer Rabbit	$24 ☐	
Yeager's Advanced Flight Trainer	$28 ☐	

Name ______________________ ID# ________

Address ______________________

City ______________ State ________ Zip ________

Country ______________ Phone ______________

VISA MC ____ - ____ - ____ - ____ Exp. ________

Signature ______________________ CP66

Softkey Publishing PO Box 110816-T Tacoma WA 98411

- Software SUBTOTAL ________
- Plus SHIPPING (see right) ________
- WA residents Sales Tax 7.8% ________
- ■ TOTAL Enclosed ________

How To Order

- **US orders:** Check the box for your selection. For Apple IIgs software, check the box in the right-hand column.
- Please add $3 per order for shipping & handling. Orders over $200 receive **free** shipping.
- Most orders shipped UPS, so use your **street address**.
- **Washington** state residents, please add 7.8% sales tax.
- **Foreign Orders:** Please inquire as to appropriate shipping fees.

* Prices subject to change without notice.

The COMPUTIST Shopper is offered as a service to our readers. Because we do not 'stock' but order only the software needed, we are able to keep operating costs to a minimum and can pass the savings on to you.

In most cases, your order is shipped within two weeks. The COMPUTIST Shopper will NOT cash your check nor charge your credit card until your software is ready to be shipped. If the software you ordered is not available, you will have the option to cancel your order, or make an alternate selection.

How to place an UnClassified Ad

If possible, send text on 5.25" Apple formated disk, include a typed sample copy with appropriate instructions. Use up to 40 characters per line, we will adjust word wrap. The Computist club member charge is $4 (for processing) plus 50 cents per line. For non-members, the charge is $4 plus $1 per line. Multiple insertions of the same ad are charged only for the line rate, unless changes are made to the copy.

Special Graphics Instructions: The first three words of the first line are printed in bold for free. If you want other words bolded, use 5 characters less per line. Use 10 characters less per line if you have a lot of uppercase letters. Bold letters are wider than normal. Circle the words you want bolded. If you want a line centered, write **CENTER** next to that line. There is no charge for centering any line.

You must check your ad, for errors, the first time it runs. Errors on our part will be corrected, then, for free. Errors or changes on your part will be charged the $4 processing fee.

Our liability for errors or omissions is limited to the cost of the ad.

We reserve the right to refuse any ad.

Washington state residents add 7.8% sales tax.

Send this form and a check or money order (funds drawn on US bank only) for the entire amount to:

COMPUTIST unCLASSIFIEDS
PO Box 110846
Tacoma, WA 98411