COMPUTIST

Canada & Mexico $7 **U.S. $3.75**

79

Table of Contents

Subscribe to COMPUTIST

We give you More!

Only $24 for 8 issues

I am:

- ○ Renewing my current subscription
- ○ Changing my address (Please include last label)
- ○ A new subscriber.

Subscription rates:

☐ **$26** **For new subscribers** and late renewals that missed 1 issue. (U.S. regular/3rd Class Mail) Includes an extra $2 for postage and handling. We will send your Starter Kit and 1st issue by 1st Class mail.

☐ **$24** U.S. regular renewal (3rd Class mail)

☐ **$34** 1st Class (U.S. / Canada / Mexico)

☐ **$54** All other Foreign (Air mail)

☐ **$68** COMBO (1st Class plus Disk)

☐ **$95** Foreign COMBO (Air mail plus Disk)

VISA Charge It! MC

Name _____

Address _____

_____

City _____ State _____ Zip _____

Country _____ Phone _____

Visa/MC _____ - _____ - _____ - _____ Exp. ____ - ____

Signature _____ #79

•Send US funds drawn on US bank. • For regular subscriptions, please allow 4-8 weeks for 1st issue or add $2 for postage and we will send your 1st issue by 1st Class mail. • Send check/money order to:

COMPUTIST 33821 E Orville Rd Eatonville WA 98328-9590 (206) 832-3055

COMPUTIST
33821 Orville Road East
Eatonville WA 98328-9590

ADDRESS CORRECTION REQUESTED

BULK RATE
U.S Postage
PAID
Kapowsin WA
Permit No. 6

COMPUTIST

Charles R. Haight — **Editor**
Jeff Hurlburt — **Reviews**
Dave Goforth — **BBS**

COMPUTIST is published by SoftKey Publishing. Address all inquiries to:

COMPUTIST
33821 East Orville Road
Eatonville, WA 98328-9590
(206) 832-3055

• 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.
• Entire contents copyright 1990 by SoftKey Publishing. All rights reserved. Copying done for other than personal or internal reference (without express written permission from the publisher) is prohibited.
• The SoftKey Publishing assumes no liability or responsibility for the products advertised in this newsletter. Although we are usually pretty much in agreement, any opinions expressed by the authors are not necessarily those of COMPUTIST magazine or SoftKey Publishing.

SUBSCRIPTIONS: Rates (for 8 issues):

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

• Subscriptions are sold by number of issues and not by month or year. An 8 issue subscription means that you will recieve 8 issues before you need to renew. It's when you will receive each issue that we're a little erratic about.

• Domestic Dealer rates: Call (206) 832-3055 for more information.
• Change Of Address: Let the U.S. Postal Service know that you are moving. Tell them that you want your mail forwarded. If your issue **does not** come to you in an envelope then you have a regular subscription and you must tell the USPS to forward your third class mail. Notify us as soon as you know your new address. When we receive your notice of change of address, we will send you an acknowledgement card. If you do not receive the acknowledgement card after 2 weeks, send another notice or call us direct.

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.

Apple® is a trademark of Apple Computers. IBM® is the IBM trademark.

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 "**ctrl**". An example of both is:

6 ctrl P

Type **6**. Next, place one finger on the **ctrl** key and then press **P**. Don't forget to press the return key.

Other special combination keypresses include **ctrl reset** and **open-apple ctrl reset**. In the former, press and hold down the **ctrl** key then press the **reset** key. In the latter, press and hold down both **ctrl** and **open-apple** 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 acquire the following:

• Applesoft program editor such as "Global Program Line Editor (GPLE)".
•Assembler such as "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.

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.

Laser 128: Your ROM includes a forced jump to the monitor. Press **ctrl return reset**.

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 that changes the open-apple ctrl reset vector to point to the monitor. *(This will void an Apple //c warranty since you must open the case to install it.)*

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 (**CALL -151**) before running any protected programs and press **# return** . This will turn on two hidden CDAs, Memory Peeker and Visit Monitor. Thereafter press **openapple ctrl esc** to go to the Desk Accessories menu. Select Visit Monitor and there you are. Use **ctrl Y** to exit.

Recommended literature

•Apple II Reference Manual (or IIe, IIc, etc.)
•DOS 3.3 & 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
```

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 special 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 (◊) charactors. **SAVE** the program at periodic intervals using the name given in the article. All characters after a REM are not checked by the checksum program so typing them is optional.

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 in older issues 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 first 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 **ctrl Y** .

SSSS.EEEE ctrl Y

Correct the lines where the checksums differ.

Writing to the RDEX editor

RDEX (are-decks) stands for: **R**eader's **D**ata **EX**change. 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 so that we may inform our other readers.

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 volunteer 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.

How to get a free library disk

Whenever possible, send everything on Apple format (5.25" - DOS/ProDOS or 3.5" - ProDOS) or IBM format (3.5") disks. Other formats are acceptable but there may be some delay as we look for someone to translate it for us. *(If you use a 5.25" disk, when we print your letter, we will return your disk with the current library disk copied onto it.)* 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 or include them in the text of your letter. 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, and you are not using the S-C Assembler, send them as normal text files.

When to include a printed letter

Don't include hardcopy (printout) unless:

a. You are writing about a bug or other printing error.
b. You are writing to ask for help.
c. You are answering another readers help request.
d. You are writing about your subscription or sending an order for back issues or software.

Bugs, requests for help and answers to requests for help are bumped to the head of the line and go in the very next issue. All other letters are printed in the order that we receive them.

Writing to get help

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. There's an old saying that goes "A properly framed question includes 90% of the answer".

How to get mail

If you are interested in receiving mail from other readers, be sure that we have a current address. 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.

How to write to RDEX authors

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 for you. Check to the right of the authors name to see if the author is writing from a foreign country and include the proper postage.

Help Line

These readers have volunteered their time to help you. Please call only within the given time frames (corrected for your time zone). No collect calls.

Jack Nissel (Disk Protection, 7-10PM EST)
(215) 365-8160

The BBS
(Bulletin Board System)

Dave Goforth is the sysop for the Computist BBS. The number is: (206) 581-9292. If you already have a User ID# and password, sign-on using the User ID#. If you are a new user, it may take a day or so to validate your new ID# and password.

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

COMPUTIST

U.S. $3.75

Canada & Mexico $7

79

Table of Contents

Features, Notes & such:

Jeff Hurlburt **TX**

The PRODUCT MONITOR

RATINGS

Superb ★★★★★
Excellent ★★★★
Very Good ★★★
Good ★★
Fair ★
Poor ☹
Bad ☹☹
Defective 💣

Home Again

Change-over

Last fall the lone remaining advertiser-supported Apple II-only monthly announced the intention to "include Mac coverage". At the time, there seemed little reason for comment. Unlike, say, a TI-99 bulletin board I've called, a computer magazine can not be content with discussions of summer vacations and fishing trips. If a publication can't find enough II products 'action' to pay the bills, it has to find something else to talk about.

Re-discovery

My reason for mentioning the II-to-Mac shift now is that inCider's move is symptomatic of maneuvering we must expect and be wary of in the post-Computer Wars I world. Regular viewers of the weekly PBS computer-stuff show "Computer Chronicles" have already heard the new 'party line'. Basically, it goes like this: "For years the home computing market has been in the doldrums. Recently, however, Apple and IBM have re-discovered the individual user! They are coming to the rescue with powerful, low-priced products like the Mac LC and PS/1."

Okay, so what is the pay-off in being "re-discovered"? First, the PS/1: It is a compact, attractive, AT-compatible '286 machine which requires an optional box to accommodate standard PC/AT peripheral cards. At $2000 for the basic color version, PS/1 is priced near the limit of what most home buyers seem to be willing to 'go for' in an initial purchase. It is also priced above faster '386 no-name (a.k.a. "grud") AT's with more RAM and larger hard disks and far above equivalent grud '286 systems.

Mac LC is an attractive, compact, Mac-compatible 68020 machine which, with the addition of a low-cost IIe card, can run IIe software. At, roughly, $3000 for the basic color version it is priced far beyond the typical home buyer's initial investment limit. However, as inCider noted in it's "Meet the Mac LC" face-off with an equivalent hard disk II system, the IIgs can end up costing as much as the base 'LC plus IIe card (assuming the IIgs purchaser makes a series of remarkably poor buying decisions). Same-price grud competition includes a new crop of much faster '486 AT's with more RAM and much larger hard disks.

It was, I believe, Abraham Lincoln who once observed: "You can re-discover some of the people all of the time and all of the people some of the time..." At least "Chronicles" avoided references to the "little people" and "unwashed masses"; but the meaning is clear enough. Technological trickle-down has proved out, we have been noticed by the big name manufacturers! The "doldrums", of course, refers to THEIR home markets— understandable, when you consider that no major manufacturer has paid any real attention to home users for the last five years. THE home market has been flourishing since 1989, when home buyers began to snap up no-name VGA+AdLib PC/AT's like they were going out of style.

They were (going out of style). First came the '286 wave; and now, as of spring '91, higher speed '386 systems are selling for well below $2000. A good barometer of what's hot (and what's not) is the computer advertising in your newspaper's Sunday "Business" section. This, typically, is where all computer stuff advertisements (with prices!) appear. I checked ours; and, believe it or not, in five or six pages plastered with computer ads, neither the PS/1 nor the Mac LC were listed. The word "Apple" did not appear even once! (Yes; I have, in the past, found an 'LC ad. Prices were NOT listed.)

Today's home programmer/ game-player/ composer/ author/ educator... is learning to shop for speed, power, and upgradability (i.e. slots!) regardless of brand name. Any suggestion that he or she is willing to settle for PS/2-1's, "Low Cost" Macs, or other sub-business-class machines is not merely off-target, it is the reverse of the actual situation. Typical office applications have little need for quality sound, large color palettes, or exceptional speed— all areas under continual pressure from designers of entertainment products. The home computer MUST be a relatively 'hot', versatile performer; and, there are all sorts of reasons why the home purchaser, in particular, aims for the 'most machine' he or she can reasonably afford. First, of course, he or she is buyer AND user. Shopping for five or ten word processor/office machines someone else will use is one thing; buying the one YOU and family members will be using is quite another matter. Other home user motivators include an interest in a wide range of steadily more demanding software, peer pressure, and concern that younger family members truly have 'the power to be their best'.

In the same broadcast, "'Chronicles" notes that home markets are becoming more attractive because "business markets are becoming saturated". Again, we are dealing with THEIR business markets. One can expect to sell just so many $4000-$6000 name brand units when more powerful machines are available at half the price. Eventually, buyers for oil corporations, universities, etc. were bound to wise-up. (Does anyone still blow $49.95 on a box of ten For-Sure-Certified diskettes?) I do not doubt that IBM, Commodore, Apple, Compaq, etc. WANT to sell piles of machinery to home users. I do doubt that any of them knows what this market looks like. If the big guys and their media placidly presume home computists to be both less demanding AND less informed, it does not augur well for their home market showdown with the gruds.

Where Are You?

You are here! Should "here" mean "primarily a **II+ (IIe, IIc, II clone) user**", then you are acutely aware of being out of the mainstream of personal computing. (Either that, or you've been 'out' for so long that you're starting to think you're 'in'!) Not only is very little new software coming from the major vendors; but nothing looks as good as the super-res and VGA stuff you've seen on other machines. You CAN upgrade the II, even to the point of adding a VGA display; but the biggest problem isn't YOUR hardware. It's the thousands of other 'old II' users who must be persuaded to make the same changes— that is, if you wish to create a recognizable 'super II' user base, develop and trade programs, attract vendors, etc., etc..

Recommendations: Keep your II, use it, enjoy it; and, when opportunities arise, improve it if the costs are not too steep. Hardware experimentation is a valuable, time-honored II owner activity. Given the rapid pace of microprocessor and component advances, there really is no telling what you might be able to achieve. Should you decide to sample the era of modern store-bought personal computing, go for the best, most II-like machine you can afford. As of Spring '91, this probably means either 1. take a risk on the IIgs OR 2. grab a PC-owner friend and shop the local grud establishments for a '386 PC/AT.

"Here" may be the joyful realm of **PC-ville**. Your 'big problems' are deciding whether to

1. add another 2MB of RAM (to handle "Windows 3.0" stuff), and/or
2. fill that little vertical panel slot with a 1.44MB 3.5" drive, and/or
3. swap out your old 40MB drive for a 120MB unit, and/or
4. dump your old VGA card plus the non-multi-sync monitor and replace with extended VGA equipment.

Recommendations: Yes, Yes, Maybe, Not yet. It may also be a good idea to keep your weekends open and your car gassed-up, just in case someone calls about doing some shopping.

If "here" is **IIgs-ville** then you already know the 'old place' isn't what it used to be. I've lost track of the number of IIgs projects "cancelled for lack of market interest", deceased hardware suppliers, and major vendor PR persons who (politely) barely refrain from laughing when I ask about "availability in IIgs format". As to national/international publications which actually devote hundreds of column inches to II coverage on a monthly basis...; suffice it to say you won't need base ten numerals to count them.

A sampling of local bulletin board listings pretty well sums up what has happened. In a printout from 1986, of 70 boards, 17 (24.3%) are listed as "Apple" BB systems, which ties with PC for the lead. By December 1990, of 298 boards, 8 (2.7%) are "Apple" BB's. Amiga and Atari shares are even smaller; C-64/128 (4.4%) and Mac (3%) come in a bit higher. PC's share is 81.5%.

You (we) were entirely justified in expecting Apple to make a major II series effort long before now— if only to prevent nearly complete dominance of unit sales, peripherals development, and software releases by a platform with which no Apple product is compatible. Think back to the late '80's and you can see that the threat of a strong, improving IIgs was the last barrier to a no-name PC/AT sweep. When, by mid-'89, the "threat" evaporated, Amiga, Atari, Mac, and even IBM each had good reason to be very very concerned. If they weren't then, you can bet they are now. Mac's big watchword used to be "Friendliness"; today it's "Connectivity". IBM, who used to believe IT decided PC standards, dares not market the PS/1 without offering an optional expansion box to hold AT-compatible cards!

So much for spilt milk. As they say in the beer commercials: "Well, Pard, (slurp) it don't get no worsen this!" 'It' could; but, evidently, it won't. Several bright spots on the horizon point to, if anything, the beginnings of a IIgs upturn. First, there's the Mac LC. Last Fall, according to "industry watchers", 'LC was destined to displace IIgs and, thus, signal the inevitable demise of the II series. Instead, as we now know, 'LC positions color Macs, more or less permanently, OUT of IIgs territory. Big Green's Mac cards are on the table. When Apple makes a serious low-end market play, it will be the 'IIgs card'.

Every IIgs user is aware that most major software vendors are not releasing 'all of that great PC stuff' in IIgs format. Too little attention is given to the continuing strong support from sources like Beagle Bros, Roger Wagner, Byte Works, and MECC. Nibble and SoftDisk-GS regularly release quality software and individual programmers continue to produce useful, innovative shareware.

Two recent product releases are especially encouraging. Apple's GS/OS 5.04 may come on as "just another revision of old, familiar GS/OS" to IIgs owners preoccupied with hardware needs. No problem; the 'Rule Book' says that if you use a machine, you're supposed to carp about the operating system. Meanwhile, PC/AT users are falling all over themselves in glee at the thought that they may soon have something like GS/OS. The other release is "Platinum Paint" from Beagle Bros. It's the kind of product that could have "mainstream users" wondering where the mainstream is. If the IIgs is dead, at least it's attracting some very classy flies. If it's not, what might we look forward to when the upturn REALLY gathers steam?!

Though InCider's "Meet the Mac LC'" piece made no recommendations and was hardly enthusiastic— well, actually, it reads like something one might come up with in a Mac prisoner of war camp— even so, Roger Wagner responded with a full-page rebuttal. One comment was especially thought provoking: "The IIgs is the best platform with which to enter the '90's." My first reaction was something along the lines of "Poor RW. He's finally blown a 'higher functions' LSI chip. How can IIgs be the 'best platform' if it's not supported?" But that, of course is RW's point. Viewed 'in itself', instead of "Will it be around next year?", "Is it smart buy?", etc. the IIgs has remarkable potential. For starters, it is the ONLY platform to offer both an abundance of expansion slots AND sophisticated firmware. It is also a compact machine widely regarded as the best looking computer ever produced by anyone. (Well, it never hurts to be good looking.)

Granting that IIgs is in the "Best Platform" running; what's the problem? Why isn't the Best Platform doing BP-type stuff? This one's easy. Just imagine that you've switched-in a bigger power supply and crammed a 1MB model IIgs with the best available performance enhancers. What is missing? Exactly! Until we can either swap-out motherboards or plug in a card to obtain 'state of the world' graphics capabilities, non of the other add-ons will be enough to spark a full-scale IIgs swarming. Conversely, once super graphics ARE in place, all of the other add-ons and the IIgs itself will immediately become vastly more attractive.

Recommendations: Keep, use, enjoy, and learn about your IIgs. It could wind up as one of the big winners in Computer Wars II. Speed-up, math co-processor, and similar enhancements are worth a serious look, so long as you are willing to accept the risks (i.e. future compatibility) that come with 'leading the pack'. As to user hardware experimentation, why not? Your Apple club's IIgs VGA card project could be THE way to crack the graphics logjam. ("What about the CRT monitor and 'old IIgs' super-res?" Easy. We buy multi-syncs, plug them into your new super IIgs VGA card and 'standardize' IIgs as a dual color monitor machine! Now, what sort of programming, flight-sim, CAD, and adventure game software do you suppose THAT would attract!!)

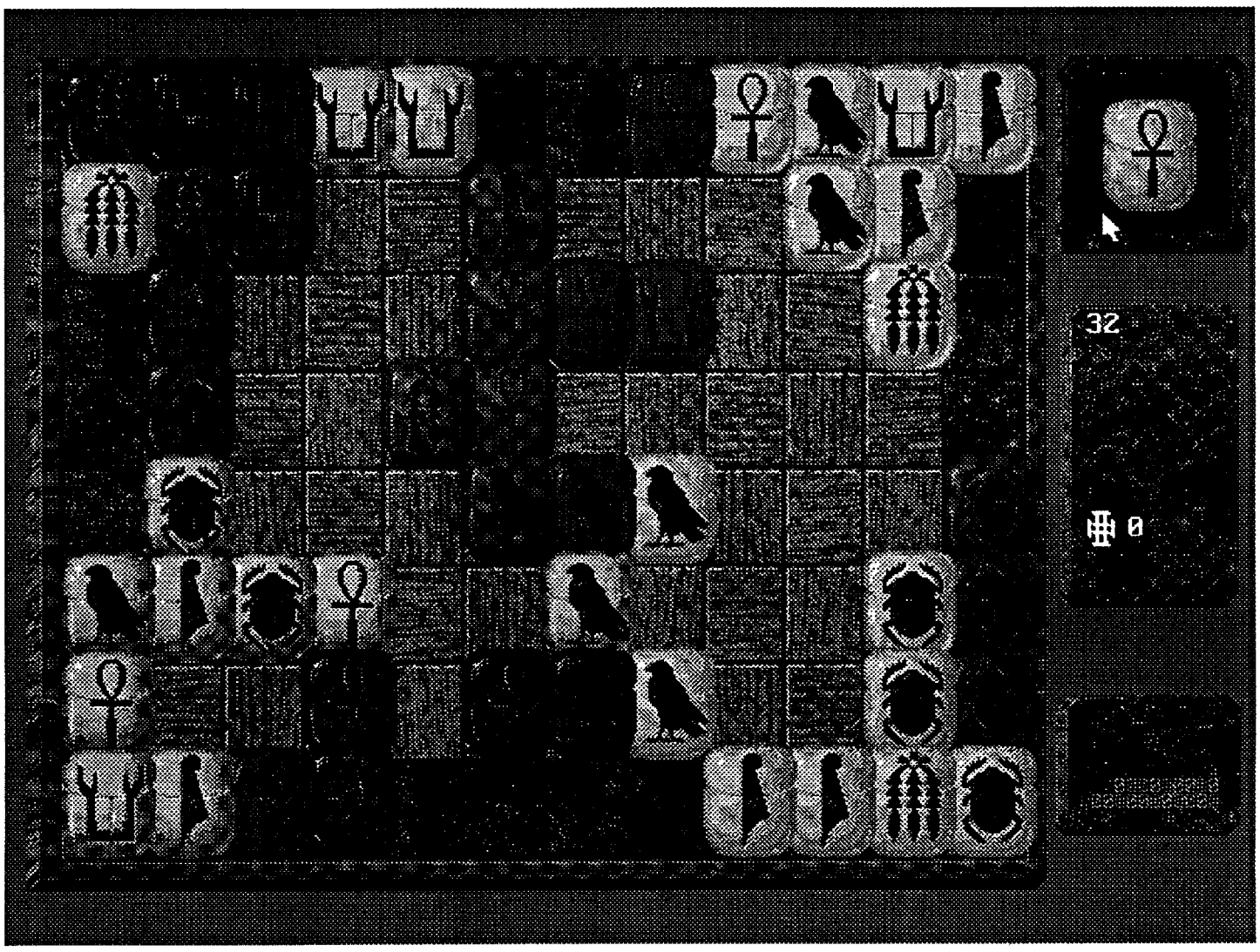
Mickey's ABC's: A Day at the Fair plus The Sound Source (Combo Pak)

★★★★

$69.95 for CGA-EGA 512K PC
Walt Disney

Mickey's ABC's' is part of a Disney "Combo Pak" which includes **The Sound Source** hardware for direct reproduction of digitized voices, sound effects, and music. Housed in a small off-white modem-size case, '**Sound Source** includes speaker, battery-powered amplifier with volume control, and cable for plugging into your PC's parallel port. The plug contains an interface and output socket so that printer's, etc. can remain connected. There is no ON/OFF switch. Instead, to conserve battery power, '**Sound Source** remains OFF unless switched ON by a program which uses it.

It goes without saying that, when **Mickey's ABC's** arrived, I could hardly wait to plug in the '**Sound Source** and start the program. After a brief musical introduction and a protection quiz, there he was, snoring peacefully (in typical, loud, clear, cartoon



style) in his second floor bedroom. Pressing "W" places a large "W" in the upper left screen; Mickey wakes up, stretches, and opens the window. "Window" appears next to "W"; and a teacher voice says "W...Window". Pressing "A" works the same, but this time Mickey slides down a pole to his kitchen (the screen scrolls smoothly to follow), opens the refrigerator, takes out an Apple, and starts munching: "Ummm Good". Evidently, most letters match two items placed somewhere in the four-room house. "P" may send Mickey into his living room to water a wilting potted Plant: "There you go!" (the plant perks up); or it may bring Pluto bounding in through the front door.

To go to the Fair, you press "F". Mickey walks out the door and soon he's there, a place just packed with alphabet stuff like Trees, Ice Cream, Sheep, Owls, ..., AND Mickey's friends. Here, "P" gets Mickey into a pie-eating contest with Donald. (Goofy is the referee.) "I" sends him over to Daisy to buy an Ice Cream cone. Great fun; and, of course, excellent artwork and animation. When you've finished exploring the fair, "H" gets Mickey Home again.

Fine, but, how would the package do with the "target audience"? Katy, an age 5 niece, paid a visit; so, I invited her into the 'Computer Room' and started the program. The plan was simple and logical: Katy, sitting in a nearby chair, would watch and listen to the presentation. I would run things and watch Katy. About 30 seconds into my demo Katy was standing up, crowding the keyboard and asking about Mickey. Then, she was scooching into my chair. Then, maybe a minute into MY demo, I was standing, and SHE was running the program! Scientific conclusion: **Mickey's ABC's'** is HOT stuff.

There are all sorts of 'learning theory ways' to look at the new Disney product. Some would emphasize the value of the "multi-modality" approach (i.e. vision, hearing, touch, kinesthetics all get 'inputs'). Some would especially approve of accommodating both Visual and Auditory learners. Others might zero-in on the "learning behavior-reward" linkages. Basically, the package appears to make "learning reading skills" fun; that is, **Mickey's ABC's'** grabs the learner's attention and teaches phonics! Like I said, "HOT stuff!!

(Supplied with manuals on both 3.5" and 5.25" media)

Ishido

★★★★

$54.95 for CGA-VGA 640K PC
Accolade

As related in the "The Way of Stones" (a booklet included in the package), Ishido's roots are in an ancient "Rule of Four" system for divination. The game evolved as a means to preserve the 72-stone set and encourage mental discipline. The booklet, which mentions recent names and dates for the game's re-discovery, goes on to relate an ancient tale about two great Ishido masters. However authentic any of this may be, it certainly sets the right mood. **Ishido**, much like Shanghai, is an elegant challenge.

Ishido is played on a 12 x 8 board. The 72 stones (held in a pouch) are colored and marked with patterns. With six patterns and six colors there are two of each (e.g. two blue beetles, two red beetles, two red birds, etc.). To begin a game, the first six stones are drawn at random from a "pouch" and placed in the standard starting squares: one in each corner and two on diagonally adjacent squares in the center. You draw stones to play, one at a time, at random from the pouch. In the computer version, score information and the "pouch" appear to the right of the board. The currently drawn stone appears in the upper right corner where it may be 'clicked' and mouse-dragged into position.

The objective is to make plays which contact the maximum number of adjacent stones (horizontally and vertically). For instance, you score most for a "four-way" (a played stone contacts four stones). A legal "four-way" play must match two of the adjacent stones for color and two for pattern. A "three-way" is a one and two or two and one match; a "two-way" is one and one; a "one-way" must match the stone it's next to for color or pattern. The game ends when you run out of stones or there is no legal play for a drawn stone.

Sound like Chinese puzzle box directions? Not to worry. First, **Ishido** IS something of a puzzle box. Setting up plays, especially high-point four-ways, takes practice and a 'feel' for the pattern on the board. Second, the well organized **Ishido** manual covers everything and even includes strategy hints. Third, thanks to in-program error messages, an illegal play produces a message explaining the matching requirements for the attempted play. In fact, you can easily learn to play the game without ever looking at the manual! Finally, should you feel the need to tap some 'inner strength' for assistance, **Ishido** offers an "Oracle" option!

Accolade's **Ishido** is presented in beautiful 16-color 640 x 480 VGA with AdLib output for CLICK! stone placement sound effects. You have a choice of five boards (eight in EGA mode) and six stone sets (which can be edited using **Ishido**'s Stone-set Editor utility or "Deluxe Paint II".) Selection is via handy pull-down menus which also handle Game Save/Load, Ancient/Modern Scoring, and **Ishido**'s comprehensive play options. The latter include Solitaire, Two-player (Cooperative or Challenge), Computer Player, and Tournaments. A large High Scores roster is maintained on disk.

Some entertainment products let you know, from the start, that "THIS was good buy; this is going to be fun." **Ishido** is one of these: subtly instructive, super addictive, and endlessly challenging elegance in stone.

(Supplied with booklet, manual, and code wheel on 3.5" and 5.25" media)

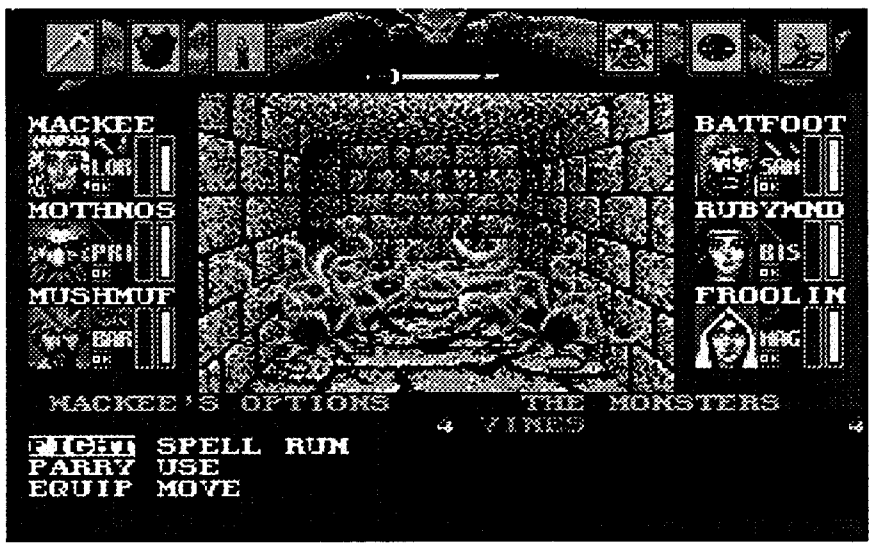
Pen, a fortress of dust, dark secrets, and dread called "Bane of the Cosmic Forge"

Exactly how your band of six inexperienced questers comes to enter the fortress is unclear. Maybe you had a little too much ale. Maybe, perfectly sober, you volunteered to remove the Pen in order to free a blighted land of its baleful influence. (Probably, that's what happened.) In any case, the gate behind you has slammed and locked itself. You're on the way to Glory— one way or the other!

Sir-Tech's press releases make a point of insisting that **Bane'** is NOT "Wizardry VI". True. The new "Wizardry" hits you with a 16-color VGA 3-D perspective forward view showing stone walls, oaken doors, fountains, flickering torches, etc.. Encounters produce beautifully drawn and animated maze personages and monsters PLUS juicy weapons effects, flying fireballs, and greasy explosions, ALL to the accompaniment of excellent digitized sound effects. (The latter, played through your PC speaker, are somewhat muted; but an AdLib-output upgrade is promised soon along with a mouse interface.) Evidently, the game's artists were just 'too pooped to paint' after finishing the animated monster pics (no wonder), because you are stuck with stone maze 'walls' (albeit, nicely detailed 'walls') throughout, even in the "forest".

You soon discover that the **Bane** "fortress" is less A maze than a collection of mazes sprinkled amongst widely disparate regions. Your band will be tested in the multi-level castle, the moors, sprawling tomb mazes, labyrinthine mines, dark enchanted forests, and much more beyond. "Wizardry V'", delivers more actual maze; **Bane'** offers a larger total gamescape and plenty of maze space to justify diligent mapping. Naturally, there's a good collection of weird contrivances and other puzzle-type obstacles. A **Hint Book** is expected; but, in-program clues are generally adequate, especially given quick Game Save/Restore to recover from lethal miscues. Still, if you can rope-in a second player to help (e.g. with mapping, sage advice, etc.), do so. The adventure will be more fun; and two players are more likely to make optimal choices at the scenario's various 'branching points'.

Smarts are important; but so is hard-hitting toughness. With eleven races and fourteen professions (or "classes"), putting together a party tailored to your style of play is easier than ever. Developing powerful, survival-prone characters is not difficult either; but you will need to master a few new



Wizardry: Bane of the Cosmic Forge

★★★ R

$59.95 for CGA-VGA 640K PC
Sir-Tech

"As you write, so shall it be!" Thus it is said of the Cosmic Forge, that who wields this Pen possesses power transcending the bounds of mere magic. Yet, when a king and his magician ally manage to secure the pen, there is no Golden Age, not even for the possessors. They disappear mysteriously and a once magnificent castle-crowned estate becomes the monster-ridden guardian of the

tricks. Aside from the standard Strength, Dexterity, etc. attributes, **Bane'** characters must also develop at least some of the twenty-four new Skills. Including "Sword", "Bow", "Skulduggery", "Theology", "Thaumaturgy", ... these affect weapons use, self-defense, scouting, magical abilities, etc..

The "tricks" have to do with getting the most out of the loosened constraints on profession changes. You can now, for example, change to some new class, boost and/or acquire skills, spells, attributes, etc. with each Level advance, and, then, change back and repeat the process! (A class change, remember, resets Level and Experience to zero; where Level advances are most rapid.) As always, the character must meet the race

and attribute requirements of the target profession. There are other limitations; but, clearly, the designers intended to shake up things a bit. Good idea! The challenge of planning **which** profession a party member should move to and **when** adds a new, very entertaining dimension to character development.

Combat remains semi-tactical (i.e. you meet the monsters and trade blows), but is better managed and more flexible than in earlier "Wizardry"'s. Magic, however, has undergone a complete overhaul. Tiltowait is gone— "GASP! They wouldn't!" they would— and the new spells, castable at several power levels, are organized into six "Spell Books" (e.g. Air, Fire, Earth, ...). Among professions which can learn spells you will now find Psionics and Alchemists along with some new "mixed classes". One new magic user class, Bard, does not learn any spells. Instead he or she invokes the powers of magical instruments. Using the lute, for example, casts a potent SLEEP. (Hint: A Bard is a VERY valuable asset to any party.)

On the whole, **Bane**' magic is less varied (e.g. no teleporting or coordinate finding) and more focused upon offensive combat. Indeed, your (or your opponent's) 'first strike' will often determine the outcome. (It doesn't take too many Fire Storms, Psionic Blasts, or Deadly Airs to put a pretty good crimp in the other side's capabilities.) Obviously, this tends to limit opportunities for "Wiz' V"-style dueling. It also produces more lost and 'unacceptable result' battles which you must sit through to the bitter end. The game supplies no 'Break-off' option to handle being plastered on the first exchange with some new, untried adversary. There's no dearth of these; and clashes with Nightgaunts, King Crabs, Stone Guardians, Drow Elves, high-level Psionics, etc. do supply much of the fun. "Wizardry" combat remains entertaining; but, it COULD be improved. By **Bane II**, it probably will be.

That the largest of the "Wizardry"'s should offer the most developed of any scenario in the series is no surprise. That the scenario incorporates scripts detailing a background of corruption, tragedy, and dark (rotten, bad) evil is a mild shock. "Wizardry," the writer's seem to be saying, "has a serious side, too." Fair enough. I enjoyed the story; but **Bane** deserves its "R". The game is not for pre-teen children. Supplied with an attractive well-written manual on both 3.5" and 5.25" media, **Bane of the Cosmic Forge** begins an entertaining, engaging experiment in LONG-play fantasy adventuring.

Altered Destiny

★★

$59.95 for CGA-VGA 640K PC
Accolade
(+ $12.95 for required Clue Book)

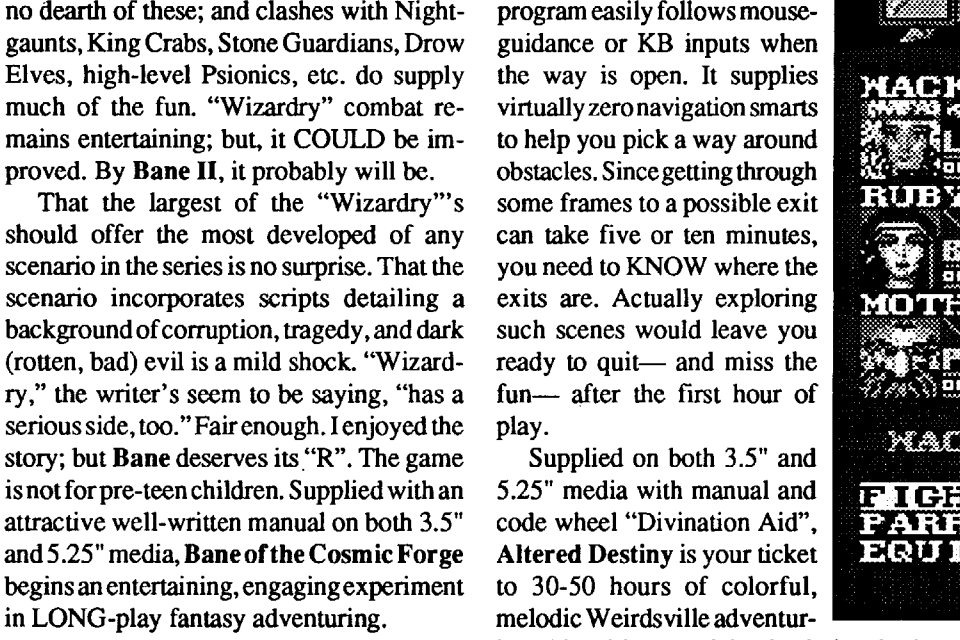
Your friendly TV repairman may be a whiz at separating the horizontal from the vertical; but he's never been much good with pesky business details, like claim checks. Some guy who looks like a Barbarian Self-Defense Academy instructor got your walnut-cased clunker; you're stuck with his new hi-def model. Tough luck. The poor devil is in for a long night of lo-res 50's sci-fi and 1-900-LUV-FONE spots; you, on the other hand— plug, Click!, "What th..." thoop!— are simply in for it!

Being sucked through a TV screen didn't hurt; mainly, it was just surprising. Your mission in Outer Limitsville, the one Conan-face was supposed to get, COULD hurt, a lot, permanently! According to your alien summoner, someone named Helmar has been corrupted by the power of The Jewel, taken it, and now threatens their world with destruction and chaos. Your job is to separate Helmar from The Jewel.

Through one richly illustrated full-color VGA frame after another, the quest takes you from shops and caves on vine-tethered sky islands, into Weird Woods, a Forest of Dreams, the Canyons of Fear, and more, all with AdLib effects and a fine musical score to enhance the mood of adventure. As in "King's Quest" and "Loom", the usual way to find out about things is to mouse-guide your figure around in the scenery and try commands like "Look", "Open", "Take", "Ask About", etc.. While some of the creatures you encounter can pose real difficulties (such as a Hoppa plant that wants to turn you into a tree), there are few combat sequences. Mainly, the challenge is to unravel clues, collect needed items, and solve problems.

Offering quick, multi-position Save and a gamescape relatively free of lethal surprises, '**Destiny**, nevertheless, requires that you keep the **Clue Book** at hand in order to assure a decent level of playability. The problem is that a few scenes are so crammed with nifty vines, trees, boulders, etc. that getting around in them becomes a genuine chore. The program easily follows mouse-guidance or KB inputs when the way is open. It supplies virtually zero navigation smarts to help you pick a way around obstacles. Since getting through some frames to a possible exit can take five or ten minutes, you need to KNOW where the exits are. Actually exploring such scenes would leave you ready to quit— and miss the fun— after the first hour of play.

Supplied on both 3.5" and 5.25" media with manual and code wheel "Divination Aid", **Altered Destiny** is your ticket to 30-50 hours of colorful, melodic Weirdsville adventuring. Also, it's your ticket back, just in time for a promised sequel! ("Does that mean I can keep the Hi-Def TV?" Beats me, kid.)



Stormovik

★★★

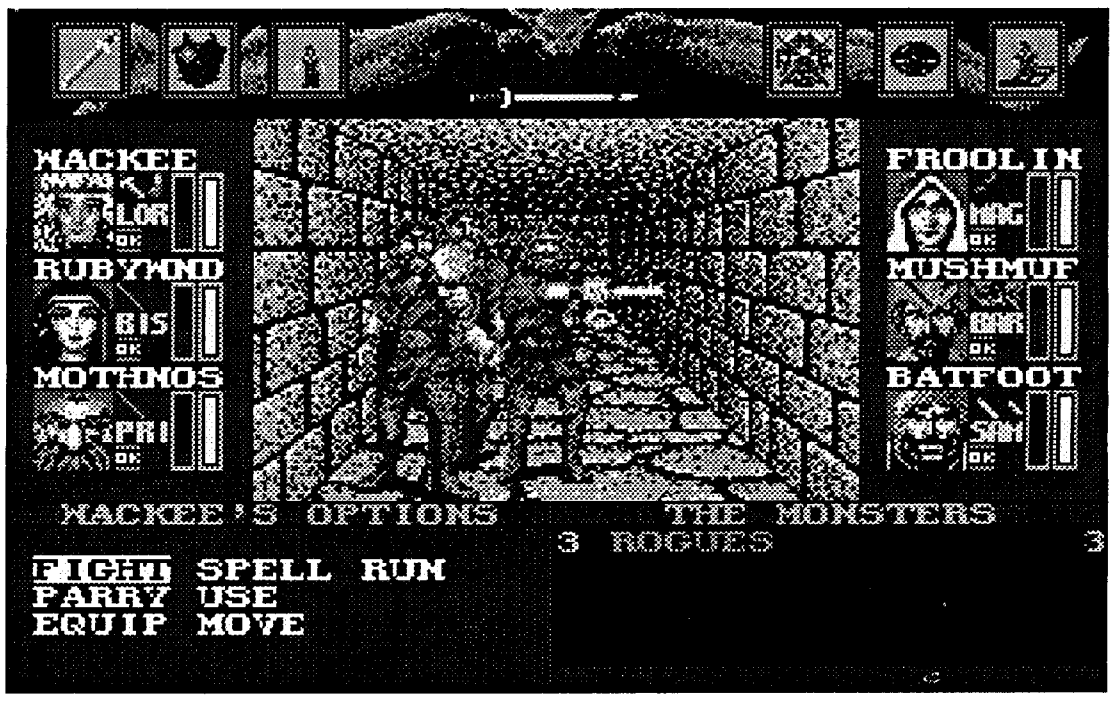
$49.95 for CGA-VGA 512K PC
Electronic Arts

It is the mid-'90's and Eastern Europe is going to 'hell in a hand basket'. The USSR isn't in great shape either, which may explain why heavily armed and armored terrorist bands range, practically at will, through large tracts of old Warsaw Pact lands. An outright invasion is out of the question; but, covert air strikes flown by officers too young to be traitors just might, eventually, crack terrorist power and turn the tide. And THAT explains how you, a mere jr. lieutenant, come to be skimming over the sovereign territory of another nation in one of the Soviet Army's premier attack fighters: SU-25 "Stormovik".

You go in under 100 meters— low enough to require turns to avoid hills and to see farm buildings all too clearly— to stay in the "grass", beneath radar detection. You go in fast, a tad below 1000 kpH, because, sooner or later, you'll be spotted anyway. The longer you take to find and knock out a bunker, zap a column of tanks, punch out a weapons barge, or whatever, the hotter it's going to get. Mother Russia wants her pilots AND her precious Stormoviks back in one piece!

I have flown faster, higher-tech aircraft in other simulations; yet, EA's **Stormovik** continues to deliver, by far, the most entertaining, addictive challenge. It's not that **Stormovik** offers any spectacular options or instrumentation advantages. Viewpoint options, instrumentation/controls (including radar and IR counter-measures), etc. are all advanced 'bat sim 'standard stuff'. In fact, **Stormovik**'s radar display is weak on ranging and targeting; and, to view the simulator's mission map, you must put up with a brief disk access delay.

What **Stormovik** does do, via smooth joystick response, convincing full-color VGA, and AdLib sound, is put you IN the cockpit flying real day, night, fog missions against real targets. (AND, thanks to an exceptionally well-organized manual and explicit point-and-click menus, **Stormovik** gets you off the ground in record time.) Each mission begins with a clever, well-written briefing and ends with a report guaranteed to make you feel like a hero (or, if you fail, like a bumbling clod). Mission successes add to your score (maintained in an on-disk Pilot Roster) which, in turn, leads to promotions. The higher your rank, the better your weapons selection and the juicier your mission options. As to 'ranking out' or 'using up' the missions? Don't worry. Perks, pride, and all the action you can handle, **Stormovik** is a simulation experience that doesn't let go 'til you press RESET!



Fast Frames, Updates, etc.

Note to PR Persons

Here's a piece of information you may wish to pass on to your PC software designers. While, for some obscure reason, it may seem like a good idea to disable, re-route, or otherwise mess up normal keyboard I/O, there is at least one major drawback. It can, for all practical purposes, make it impossible to employ graphics capture software to obtain a "screen shot" for publication.

Quest for Gold ☹

Drawn from the popular TV series, Disney's **Duck Tales: The Quest for Gold** ($44.95 for CGA-EGA 512K PC) pits you (Scrooge) against Flintheart Glomgold in a world-spanning competition to decide who is the best money maker. Stock investing or a dive for rare coins in the old Money Bin are your more mundane options. For big bucks, you, with your brave nephews, must fly off to distant sites in search of Lost Treasure!

Climbing mountains, exploring mummy-infested caves, swinging through jungles, and more, '**Quest for Gold** certainly promises loads of colorful, nicely animated arcade adventuring. THEN you run into SLOW disk accesses, weak old-PC sound, no Game Save, and (the killer) incredibly clumsy controls. A pretty box and clever manual are plusses; they aren't miracle workers.

The Three Stooges ★

Aimed squarely at the stooge in each of us, Cinemaware's '**Three Stooges** ($49.95, for 1.25MB IIgs) puts you in charge of the dauntless trio as they work to save an orphanage from foreclosure. Starting, literally, at 'square one' of 180 on a one-way boardgame-type "street", you have one month (about thirty turns) to earn the necessary $5000. $10000 saves the orphanage AND supplies needed repairs. REALLY rake it in and the boys get to marry Ma's three beautiful daughters!

Each turn begins when you stop a hand which is moving rapidly over the next six squares. (From time to time Moe needs to 'slap some sense' into Larry and Curly to reduce hand speed and get better control over square picking.) Possible selections include one of four arcades, Money for Nothing (you may find a few hundred $$$), Evil Banker (lose $ and/or a turn), and Stooges Trivia. The four mini-game money-makers include a Pie Fight (as new waiters you get to plaster the "swells"), Boxing (actually, a race through the streets to find a radio to play "Pop Goes the Weasel"), entering a soup & Cracker-eating contest (the oysters in the soup are competing for the crackers), and Hospital work (a typical Stooge race to the operating room).

All of which turns out to be a fair challenge, but only mildly entertaining. Despite high quality artwork and animation, good sound, and decent joystick response, the game is less an arcade bonanza than an 'event'. Including digitized Stooge images and voice tracks PLUS a very nice twenty-page "'Three Stooges" history booklet, this is a slick, cute collector's item for dedicated Stoogeophiles.

Baneful Tales

Your bard pulls on an ear and frowns at the maps you've been keeping. Your sorceress just sighs and follows the others into the tavern. "Well, darn it," you protest, "this place isn't supposed to be here! You guys think I'm some sort of klutz?!"

Be that as it may, Ye Bane du Computist appears to be a substantial enough establishment and a busy one as well. It's barely late afternoon, yet every adventurer on the Cosmic Forge quest must be here, swilling ale and swapping tales. Scarcely have you

joined your companions at a corner table and downed a slug of frothy dark liquid, when the first juicy snatches of conversation surface:

"Turns out the catapult needs a new heavy-duty rubber band, besides having a busted sprocket. Once you merge four strands from Rubber Beasts and get the smith to fix the sprocket, you're in business. Just drop in a boulder, wind, and..."

"... then Narf answers: 'Madness makes us free' and the Sirens are all sweetness and light! 'Oh noble travelers', says the chief Siren, 'please take these here magic boots and ...'"

"... used the chisel. The catch is the diamond wall has four faces, each approached by a different route! You have to crack each face at least once..."

"... discovered by the Turdnil party. See, they went up the West Slope and, somehow, made it back down and to the pyramid. Too bad there's no explanation of these two red dots. Whatever they are, one's next to a 'Guardian' and the other's near a walkway leading from the top of the pyramid."

"... when you blow the Horn. Talk about a sound to chill your bones! Charron wants ashes for his ferryin' service; but, if you talk to the guy, his real interest is the cylinders. Seems like he's supposed to return them to the Isle of the Dead. Nice payoff, too, if I do say so— 'ceptin' the one he said was 'too hot' for him to handle! ..."

"Got a pencil? Okay, here's a list of some items that have a 'special power' to boost attributes:

1. Ruby Talisman- Intelligence +1
2. Heraldic Shield- Strength +1
3. Parrot- Personality +1
4. Ankh of Purity- Karma +1
5. Bone Necklace- Vitality +1
6. Mantis Gloves- Dexterity +1 (two times)
7. Mantis Boots- Speed +1 (two times)
8. Ankh of Sanctity- Piety +1

Remember, most of these also enhance armor class when worn; so you gotta think twice about using the 'special power'. Except for the first use of 'mantis' stuff, the attribute boost uses up the item."

"... had us running around in circles. Fortunately, the splotch was sticky enough to hold the chest in place. As for the 'Mu Mu idol..."

"... and found some really arcane stuff from Gorbash. See, here he talks about reducing 'Level' at 'byte $24 in an experienced character's block in Savegame.dbs', whatever that is. Then there's something about 'restarting play' and 'cleaning up on Level advancement after the next battle'? Like I said, weird!

"The racket from the bell, twistin' and turnin' on the rope, and bat swarms were bad enough; but the real problem is things look the same from either side. Finally, Mothnose happened to check his compass. We must uv swung across four or five times without knowin' we'd made it!"

"... make you a sketch of the pyramid. Not counting the basement and basement pits, it has four levels: 9x9, 7x7, 5x5, and 3x3. The tricky part is keeping track of these stairways and finding the buttons that engage the...

"... sitting on a giant toadstool. Guess what he wants us to find— right. Anyway, someone finally mentions 'Claim check' and he gets all enthusiastic and gives us a question for the Bottle Oracle. Funny thing is, the clerk at the reclamation center turns out to be a pretty good hint for the claim number!"

"Mindless though they are said to be, Nightgaunts can, indeed, be tough. Ol' Goo-goo here found a cure, though. He just pops 'em with a quick Astral Gate."

"... until Rubywand took a look at it. She claims 'The Hand of Destiny' means 'Cosmic Forge'. She says it's mage-talk, sort uv a code word...

"Saeren sells some nice stuff, especially if you've got a pixie in your party. Mainly, she knows about the Delphi oracle..."

"... called it 'Igmo's Egress' because he was the only one with enough nerve to just pick up the red stick and strike it. KA-POOM!, a door appears; and you've got a direct route from the Wizard's Lair to the Mines!"

"... of Truth'? The thing was polished to a mirror finish! One well-placed clip with our pick on the base was all it took. We had some reflecting stones without messing up the..."

"Sure, you buy a Zwiehander from the Smith and you have a very nice weapon. The Dragonslayer is a tad better; but the biggie is the Avenger! Batfoot wears Ebony armor, packs Avenger in one hand and a Skull Dagger in the other. The guy is a walking siege tower!!"

"... so banged up that we were beginning to think we'd have to retrace our path through the tombs and head back to the fountains on the Isle of the Damned. That's when we rounded a bend near the Queen's chamber and saw THE Fountain! ..."

"Ha! That's because there ain't no 'right answer'! You meet up with Rebecca and you're headed for the temple basement 'Guest Suite'; that's all there is to it. Lucky we had them red mushrooms from the 'pillar. S'posedly, though, there's another way out."

Next?

For sure, Terrapin's **Logo Plus** for Apple II series and Beagle Bros's **Platinum Paint** for Apple IIgs. Likely, too, are **Countdown** from Access, and KYE's **Genius Mouse** for PC. Plus ...whatever. As always, comments, suggestions, etc. are welcome. (See PM listing below.)

Vendors

ACCOLADE
attn: Melinda Mongelluzzo
550 S. Winchester Blvd., Suite 200
San Jose, CA 95128
(408-985-1700)

AD LIB
attn: Jill Carette
220 Grand-Allee East, Suite 960
Quebec QC G1R 2J1
Canada
(800-463-2686)

APPLE COMPUTER
attn: mailstop 36I
20525 Mariani Avenue
Cupertino, CA 95014
(405-996-1010)

CENTER FOR GIFTED AND TALENTED
attn: Theresa Monaco
University of Houston
University Park/Farrish Hall #123
Houston, TX 77004

CINAMAWARE
attn: PR/Marketing
4165 Thousand Oaks Blvd.
West Lake Village, CA 91362
(805-495-6515)

ELECTRONIC ARTS
attn: Lisa Higgins
1820 Gateway Drive
San Mateo, CA 94404
(415-571-7171 orders: 800-245-4525))

JANKLOW BENDER
attn: Kim Adamo
257 Park Avenue South
New York, NY 10010

PRODUCT MONITOR
attn: Jeff Hurlburt
7814 Santa Elena
Houston, TX 77061
(713-645-8680)

SIR-TECH
attn: Sheri Mitchell
P.O. Box 245
Charlestown Mall
Ogdensburg, NY 13669
(800-447-1230/315-393-6633)

WALT DISNEY COMPUTER SOFTWARE
attn: Kirk Green
500 South Buena Vista
Burbank, CA 91521
(818-567-5340) ref: Janklow-Bender

B. Dudley Brett — Canada

Other Notes

I would like to add my words of encouragement to the many who have written in support. Virtually the only fine Apple II publications now extant are COMPUTIST and A-2 Central. Nibble I find increasingly non-readable as I do not have a IIgs and have no inclination to purchase one. I have been a subscriber of COMPUTIST since issue #8, and will continue despite any rising cost.

I rather like the tabloid format. The information enclosed is so vast that any return to the previous format would be a come-down (It would necessitate a sharp drop in material). I also note, particularly in volume #72, much improvement in softkey articles. More and more readers are writing explanatory articles, rather than the change this byte to that byte note. I learned assembly language more from the study of COMPUTIST articles than from any other source. But! I sure don't learn anything from the one byte patches type of article (and neither can anybody else, especially when the patches don't work).

As I now also have a Mac, I would appreciate information on Mac softkeys. What utilities are required and what techniques can be applied in this noble endeavor? So far, I have confined my efforts with the Mac to using word-processor, database, etc. and have only recently started programming in QuickBasic and Pascal.

James A. Hodge — CT

Object Module Format (OMF)

I started using the ORCA/Disassembler recently, and I found that I needed the information in the relocation dictionary to do an efficient disassembly. The ORCA/Disassembler is great for dismembering P16 type files, but, since it doesn't automatically generate labels or ENTRY directives for inter-segment references, you need a list of address references to aid in defining data areas and to create a correct disassembly. First, I'll show you where the information is, and then I'll present some utilities to acquire and manipulate it.

OMF File Structure

With ProDOS 16, Apple created a disk and memory management system able to deal with large programs and files. As with previous advances in their operating systems, understanding the system involves many details. The key to dealing with (and patching) OMF (Object Module Format) files is a knowledge of the file structure. I want to discuss the most common executable file structure, OMF Version 2.

All relocatable executable files follow the same file structure. This includes EXE, TOL, CDA, NDA, S16, P16 and a few others. They are composed of one or more segments, and each segment has a segment header, a body (the code/data image), and a relocation dictionary. Where an old-fashioned BIN file was usually a simple image of what was in memory, an OMF file is a collection of sequential records telling the loader how to build the program.

The header has 16 fields. They are:
(format is: # of bytes used - purpose):

4 - Byte count: # of bytes seg. occupies on disk
4 - Reserved: # bytes to add at segment end in memory
4 - Length: # of bytes segment occupies in memory
2 - unused
1 - Number Len: length of numbers (usual value is 4)
1 - Version: OMF format version (should be 1 or 2)
4 - ank size: usual value is $10000
2 - Kind: see following discussion
2 - unused
4 - Org: forces loading at a specific location
4 - Alignment: forces loading on page or seg. boundary
1 - Number sex: flag for numbers (usually lsb first)
1 - unused
2 - Seg No.: segment number in file
4 - Segment entry: where to start execution in segment
2 - Disp. to names: offset from header start to load name
2 - Disp. to body: offset from header start to "program"
10 - Load name
Varies - Segment name: 1 byte minimum

The KIND field is one byte long in OMF 1.0 and two bytes in OMF 2.0. It informs the loader about the type of segment so that appropriate action will be taken. The most common KINDs are code ($00), data ($01) and direct page/stack ($12) segments. Initialization segments ($10) are executed as soon as they are loaded. The KIND value is set by the START, PRIVATE, DATA, PRIVDATA or KIND assembler directives. The KIND directive allows fine control over the type of a segment. Now I'm going to "cop out" on any further discussion and refer you to your assembler manual, Apple IIgs Programmer's Workshop Ref., Apple IIgs ProDOS 16 Ref., Call A.P.P.L.E. October 1988 "Segmentation, Part 2" by Mike Westerfield, or the GS/OS reference if you want to know more about KIND.

According to the P16 reference there are only eight types of records allowed in executable files; RELOC ($E2), INTERSEG ($E3), DS ($F1), LCONST ($F2), cRELOC ($F5), cINTRSEG ($F6), SUPER ($F7) and END ($00) records.

The body is defined with two records, the DS and LCONST records. DS records are like their assembler counterparts that Define Storage; they start with $F1 and use the following 4 bytes to reserve space. The LCONST records start with $F2 then have a 4 byte number telling how many bytes of constant data are contained in the record, followed (surprise, surprise!) by the appropriate amount of code or data.

If the segment doesn't reference any labels in other segments, and it doesn't contain any internal absolute references, it won't have any RELOC, INTERSEG, cRELOC, cINTERSEG or SUPER records. Following the LCONST and/or DS records will be an END ($00) record, followed by either end-of-file or the segment header for the next segment.

Usually there are internal and intersegment references, and each reference requires a relocation record of some sort. RELOC, cRELOC and some SUPER records are used for internal absolute references, while INTERSEG, cINTERSEG and some SUPER records are used for references to addresses in other segments.

If, for instance, a segment has a JMP to another location within the same segment, that operand of the JMP statement would have a RELOC, cRELOC or SUPER record associated with it. (I'll explain SUPER records later.) If you had the following piece of code:

```
1234:4C 25 A0    JMP  far_away far_away = $A025
```

The associated record would look like one of these:

```
RELOC : E2 02 00 35 12 00 00 25 A0
00 00
cRELOC: F5 02 00 35 12 25 A0
```

Both records start with their opcodes (E2 and F5) and a count of the number of bytes to patch (2 in this example) and a bit-shift operator. They vary in the use of 4 byte (RELOC) or 2 byte (cRELOC) addresses, but the information is contained in the same order. First is the offset from the start of the segment to apply the patch at ($1235), followed by the offset of the address reference ($A025). As you can see, the "c" in cRELOC (and in cINTERSEG) stands for Compressed.

The interseg records reference addresses in other segments. The loader can't relocate intersegment references until it knows where in memory the other segments are loaded. If segment 2 contains the following instruction:

```
1234:22 00 00 00 JSL  S3_label label at $8765 in seg. 3
```

The linker would create one of the following records:

```
INTERSEG : E3 03 00 35 12 00 00 01
00 03 00 65 87 00 00
cINTERSEG: F6 03 00 35 12 03 65 87
```

The first bytes are the op-codes (E3 and F6), next is the number of bytes to patch, followed by the bit-shift operator. The INTERSEG record has a 4 byte patch address while the cINTERSEG has a 2 byte patch address. The INTERSEG record then has a 2 byte file number that the cINTERSEG record lacks (the file number should be 1 unless it's a reference to a run-time-library). The INTERSEG record has a 2 byte value for the segment reference (03 00, or segment 3, in this example) while the cINTERSEG record makes do with 1 byte. Last is the offset of the address reference within the segment ($8765); 4 bytes for the INTERSEG record and only 2 bytes for the cINTERSEG version.

The last record type is the SUPER ($F7) record. The P16 reference describes it as a "super-compressed relocation record (the equivalent of many cRELOC or cINTERSEG records)". The GS/OS and APW references provide full descriptions, however.

For a start, SUPER records are composed of the low bytes of addresses that need to be patched. The address reference (to be adjusted) is stored "in place" in the LCONST record. There is a considerable space saving.

There are 38 different types of SUPER records, but they all have the same structure. They start with the opcode F7, followed by a 4 byte LENGTH field (indicating the number of bytes that follow), a 1 byte TYPE field, and then the body, containing the (LENGTH - 1) number of bytes. (I capitalized LENGTH and TYPE so they stand out. It is not an Apple convention.)

The TYPE field can have a value of 0 through 37 ($00 - $25). The chart in figure 1 will relate the TYPE value to the SUPER record:

There are additional nuances related to the type. For INTERSEG1 relocations, the segment byte is the third byte at the "to be patched" location, while for INTERSEG2 through INTERSEG36 the segment is implicit in the TYPE. The chart in figure 2 relates TYPE, INTERSEG#, and the referenced segment number:

The last element of the SUPER record is the SUBRECORD. Subrecords are 1 byte skip counts or 1 byte offset counts followed by (count + 1) 1 byte offsets. Skip counts are single bytes with their high bit set (value > $80). The (skip count - $80) indicates the number of 256 byte pages to skip. For example, a skip count of $93 would mean that the next $13 (19 decimal) pages should be skipped. The offset count (plus 1) indicates how many low order address bytes follow it. The page (high order address byte) is implicit in the subrecord's position in the body of the SUPER record. A couple of examples should help clarify SUPER records. (These are from a real program.)

```
F7 1C000000 12 83 00B0 84 0061 81
002C 0118D8 00DF 030345A4CD
   0010 0194D6 0119BE
F7 1C000000 1E 83 00AD 84 005E 81
002F 011BDB 00E2 030648A1D0
   0013 0197D9 011CC1
```

I broke these up to illustrate the various parts. There is the opcode (F7) followed by the 4 byte length (with least significant byte first) followed by the TYPE ($12 and $1E in these examples) followed by 12 subrecords in each example. The TYPE of $12 and $1E both mean that the addresses are related to segment 5, but the TYPE $12 addresses are not bit shifted and the TYPE $1E are shifted $F0 (-$10) bits. The first subrecord in each example is 83, which means to skip 3 pages, pages 0, 1, and 2 in this case. The second subrecord in the first example is 00B0, which tells the loader that at $03B0 in this segment is an address that needs to be adjusted to reflect where segment 5 was loaded into memory. The loader then adjusts the 2 byte address it finds at $03B0. The third subrecord in both examples is another skip count indicating pages 4, 5, 6, and 7 are to be skipped. The next patch will be made to page 8 (addresses of $08xx). The rest of the records are, as they say, left as exercises for the reader.

(Trivia: I've noticed a difference between files assembled and linked with ORCA and other assembler/linker combinations. ORCA seems to put all the cRELOC records first, then cINTERSEG, then SUPER records, with no discernible pattern to the patch or address info. Merlin puts out a relocation dictionary where the records are in patch address order, but the cRELOC and cINTERSEG records are integrated with one another. This information is of no particular significance, but I thought it was interesting. Do you suppose one form might have an advantage over the other? I wonder what other assembler/linker combos do?)

(More useless information for those of you who have wondered why people don't provide object code with "modern" assembly listings. As you can see, there's a lot more to an OMF file than there is to an old-fashioned BIN file. It WOULD be possible to re-create an EXE, S16 or CDA by punching in a hex dump, but there is still the segment header and relocation dictionary to be considered. If an author provided a dump of the complete file then you could CREATE a file entry and enter and BSAVE the hex dump. It would work, but there's so much information that it would be a MAJOR hassle. A tiny little piece of machine language might need a file several times the size of the code.)

As you can see, decoding the SUPER records is perhaps the most difficult part of understanding OMF files, but a program can ease the task considerably. Read on.

ORCA/Disassembler utilities

Reading an OMF file

I created OMFR (for OMF file Reader) to read the relocation dictionaries of OMF files I'm going to disassemble, and present me with data I can use in the disassembly process. OMFR is not as flexible as the ORCA/APW Dumpobj utility, but Dumpobj doesn't detail the SUPER records, and OMFR does. The information I get is invaluable for defining data areas and finding all the locations that are affected (patched) by the loader.

OMFR was designed to work on OMF version 2.0 files, and it will not completely understand lesser versions without some minor revision (a project for some other day).

OMFR puts out files containing relocation, interseg, and ds and constant info. The output files are called RELOC, INTERSEG, and CONST.DS, respectively. The RELOC and INTERSEG files are intended to be manipulated with the Appleworks database, while CONST.DS is input for the word processor. RELOC and INTERSEG can be sorted and combined in several different ways to produce files for input to the other programs (SCRIPT.ENTRY, SCRIPT.LABELS, and XREFFER) in this article.

There's not much to say about using OMFR. The name and type of the OMF file you want to examine should be in line 80. Line 95 has the switches that determine whether output files are created. Set DS, RL and/or IS to a non-zero value if you want disk output. The variable ND (meaning No Detail) determines whether SUPER records are, or are not, detailed. The program was written in a modular fashion, so if you don't like the output it should be easy to customize to suit your taste. For instance, if you want "position in file" information, print the value of the variable "B".

The screen output of OMFR is, frankly, lousy when compared to Dumpobj, but the disk output is much more usable, which is what I was after in the first place. You might want to use Dumpobj to get an initial feel for the structure of a file, but OMFR and it's related programs will give you the ability to collect and organize the relocation dictionary data and gain some very useful insight into the relationships between data areas and subroutines.

It's possible to relate what's on disk to what's in memory by using the "Memory Peeker" CDA or, even better, the "IIgs Internals" CDA by Ken Kashmarek (it comes with Merlin 16+). It seems that segments rarely load into memory in the same way that they are ordered in their file. The CONST.DS output file contains the segment numbers and their lengths, so all you need to do is match the lengths with the info provided by one of the CDAs. Knowing where things are can make it easier to find the exact spot to apply a patch.

If you use OMFR, I ** STRONGLY ** recommend putting the file you want to examine on a RAM or ROM disk. This program would make a good disk drive torture test.

Figure 1

TYPE	SUPER record	bit-shift count	relocate # bytes
00	RELOC2	0	2
01	RELOC3	0	3
02	INTERSEG1	0	3
$03 - $0D	INTERSEG2 - INTERSEG12	0	3
$0E - $19	INTERSEG13 - INTERSEG24	0	2
$1A - $25	INTERSEG25 - INTERSEG36	$F0 (-16)	2

Figure 2

TYPE	SUPER INTERSEG #	referenced segment #
$03 - $0D	2 - 12	2 - 12
$0E - $19	13 - 24	1 - 12
$1A - $25	25 - 36	1 - 12

OMFR

```
10 REM OMF File Reader
20 REM by James A. Hodge - 10/8/
   90#
30 GOTO 80
40 REM hex converter
50 X = INT (N / 16): PRINT CHR$
   (48 + X + 7 * (X > 9));
60 X = N - X * 16: PRINT CHR$ (48
   + X + 7 * (X > 9));
70 RETURN
80 FI$ = "/ram5/target.program,
   t$b3"
90 F1$ = "/ram5/const.ds" :F2$ =
   "/ram5/reloc" :F3$ = "/ram5/
   interseg"
95 CD = 0:RL = 0:IS = 0:ND = 0:
   REM $F7 NoDetail
100 DIM X(99),HDR$(19),LC(30,1),
   PC(30)
110 D$ = CHR$ (4)
120 FOR I = 1 TO 19: READ HDR$(I):
   NEXT
130 IF CD THEN PRINT D$ "open◊"
   F1$
135 IF RL THEN PRINT D$ "open◊"
   F2$
140 IF IS THEN PRINT D$ "open◊"
   F3$
145 ONERR GOTO 290
150 GOSUB 2000: REM seg header
160 L = 1
170 PRINT D$ "BLOAD◊" FI$ ",A$2000
   ,L" L ",B" B:B = B + L
180 N = PEEK (8192)
190 IF N = 226 THEN GOSUB 500:
   GOTO 270: REM reloc
200 IF N = 227 THEN GOSUB 600:
   GOTO 270: REM interseg
210 IF N = 241 THEN GOSUB 700:
   GOTO 270: REM ds
220 IF N = 242 THEN GOSUB 800:
   GOTO 270: REM lconst
230 IF N = 245 THEN GOSUB 900:
   GOTO 270: REM cRELOC
240 IF N = 246 THEN GOSUB 1000:
   GOTO 270: REM cINTERSEG
250 IF N = 247 THEN GOSUB 1100:
   GOTO 270: REM super
260 IF N = 0 THEN GOSUB 50: PRINT
   : PRINT : GOTO 150
270 REM
280 GOTO 160
290 CALL - 3288: PRINT D$ "close"
300 END
500 REM RELOC = $E2 = 226
510 L = 10: PRINT D$ "BLOAD◊" FI$
   ",A$2000,L" L ",B" B:B = B + L
520 IF RL THEN PRINT D$ "write"
   F2$:N = X(34): GOSUB 50:N =
   PEEK (8195): GOSUB 50:N = PEEK
   (8194): GOSUB 50: PRINT :N =
   X(34): GOSUB 50:N = PEEK
   (8199): GOSUB 50:N = PEEK
   (8198): GOSUB 50: PRINT :
   RETURN
530 GOSUB 50: PRINT "◊" ;: FOR I =
   8192 TO 8201:N = PEEK (I):
   GOSUB 50: NEXT : PRINT
540 RETURN
600 REM INTERSEG = $E3 = 227
610 L = 14: PRINT D$ "BLOAD◊" FI$
   ",A$2000,L" L ",B" B:B = B + L
620 IF IS THEN PRINT D$ "write"
   F3$:N = X(34): GOSUB 50:N =
   PEEK (8195): GOSUB 50:N = PEEK
   (8194): GOSUB 50: PRINT :N =
   PEEK (8200): GOSUB 50:N = PEEK
   (8803): GOSUB 50:N = PEEK
   (8202): GOSUB 50: PRINT :
   RETURN
630 GOSUB 50: PRINT "◊" ;: FOR I =
   8192 TO 8205:N = PEEK (I):
   GOSUB 50: NEXT : PRINT
640 RETURN
700 REM DS = $F1 = 241
710 L = 4: PRINT D$ "BLOAD◊" FI$
   ",A$2000,L" L ",B" B:B = B + L
715 FOR I = 8195 TO 8192 STEP -
   1:PC(SS) = PC(SS) + PEEK (I) *
   16 ^ (2 * ABS (8192 - I)): NEXT
720 IF CD THEN PRINT D$ "write"
   F1$: GOSUB 50:N = X(34): GOSUB
   50: FOR I = 8195 TO 8192 STEP -
   1:N = PEEK (I): GOSUB 50: NEXT
   : PRINT : RETURN
730 GOSUB 50: PRINT "◊" ;: FOR I =
   8195 TO 8192 STEP - 1:N = PEEK
   (I): GOSUB 50: NEXT : PRINT
740 RETURN
800 REM LCONST = $F2 = 242
810 L = 4: PRINT D$ "BLOAD◊" FI$
   ",A$2000,L" L ",B" B:B = B + L
820 IF CD THEN PRINT D$ "write"
   F1$: GOSUB 50:N = X(34): GOSUB
   50: FOR I = 8195 TO 8192 STEP -
   1:N = PEEK (I): GOSUB 50: NEXT
   : PRINT : GOTO 840
830 GOSUB 50: PRINT "◊" ;: FOR I =
   8195 TO 8192 STEP - 1:N = PEEK
   (I): GOSUB 50: NEXT : PRINT
840 M = 0: FOR I = 8195 TO 8192
   STEP - 1:M = M + PEEK (I) * 16
   ^ (2 * ABS (8192 - I)): NEXT :B
   = B + M
850 LC(SS,0) = B - M:LC(SS,1) =
   M:SS = SS + 1:PC(SS) = PC(SS -
   1) + M
860 RETURN
900 REM cRELOC = $F5 = 245
910 L = 6: PRINT D$ "BLOAD◊" FI$
   ",A$2000,L" L ",B" B:B = B + L
920 IF RL THEN PRINT D$ "write"
   F2$:N = X(34): GOSUB 50:N =
   PEEK (8195): GOSUB 50:N = PEEK
```

```
   (8194): GOSUB 50: PRINT :N =
   X(34): GOSUB 50:N = PEEK
   (8197): GOSUB 50:N = PEEK
   (8196): GOSUB 50: PRINT :
   RETURN
930 GOSUB 50: PRINT "◊" ;: FOR I =
   8192 TO 8197:N = PEEK (I):
   GOSUB 50: NEXT : PRINT
940 RETURN
1000 REM cINTERSEG = $F6 = 246
1010 L = 7: PRINT D$ "BLOAD◊" FI$
   ",A$2000,L" L ",B" B:B = B + L
1020 IF IS THEN PRINT D$ "write"
   F3$:N = X(34): GOSUB 50:N =
   PEEK (8195): GOSUB 50:N = PEEK
   (8194): GOSUB 50: PRINT :N =
   PEEK (8196): GOSUB 50:N = PEEK
   (8198): GOSUB 50:N = PEEK
   (8197): GOSUB 50: PRINT :
   RETURN
1030 GOSUB 50: PRINT "◊" ;: FOR I
   = 8192 TO 8198:N = PEEK (I):
   GOSUB 50: NEXT : PRINT
1040 RETURN
1100 REM SUPER = $F7 = 247
1105 IF ND THEN 1400: REM no
   detail
1110 L = 4: PRINT D$ "BLOAD◊" FI$
   ",A$2000,L" L ",B" B:B = B + L
1120 M = 0: FOR I = 8195 TO 8192
   STEP - 1:M = M + PEEK (I) * 16
   ^ (2 * ABS (8192 - I)): NEXT
1130 L = M: PRINT D$ "BLOAD◊" FI$
   ",A$2000,L" L ",B" B:B = B + L
1140 ST = PEEK (8192):LL = 3: IF
   ST = 0 OR ST > 13 THEN LL = 2:
   REM 2 or 3 byte load
1150 P = 0:II = 0
1160 FOR I = 8193 TO 8193 + L - 2
1170 J = PEEK (I) + 1: IF J > 128
   THEN P = P + J - 129: GOTO 1300
1180 FOR K = 1 TO J:PV = PEEK (I +
   K)
1190 IF PC(II) + LC(II,1) < P *
   256 + PV THEN II = II + 1: GOTO
   1190
1200 PRINT D$ "BLOAD◊" FI$
   ",A$3000,L" LL ",B" LC(II,0) +
   P * 256 + PV - PC(II)
1205 IF RL AND ST < 2 THEN PRINT
   D$ "write" F2$
1210 IF IS AND ST > 1 THEN PRINT
   D$ "write" F3$
1215 IF RL + IS = 0 THEN PRINT
   "F7" ;:N = ST: GOSUB 50: PRINT
   "◊" ;
1220 N = X(34): GOSUB 50:N = P:
   GOSUB 50:N = PV: GOSUB 50: IF
   RL + IS > 0 THEN PRINT : REM
   patch adr.
1225 IF ST < 2 THEN N = X(34): REM
   reloc2 & 3
1230 IF ST = 2 THEN N = PEEK
   (12290)
1240 IF ST > 2 AND ST < 14 THEN N
   = ST - 1: REM maybe!
1250 IF ST > 13 THEN N = ST - 13 -
   (ST > 25) * 12
1260 GOSUB 50:N = PEEK (12289):
   GOSUB 50:N = PEEK (12288):
   GOSUB 50: PRINT : REM adr.ref.
1270 NEXT
1280 I = I + J
1290 P = P + 1
1300 NEXT
1310 RETURN
1400 REM SUPER = $F7 = 247 - no
   detail
1410 L = 5: PRINT D$ "BLOAD◊" FI$
   ",A$2000,L" L ",B" B:B = B + L
   - 1
1420 IF CD THEN PRINT D$ "write"
   F1$: GOSUB 50: FOR I = 8195 TO
   8192 STEP - 1:N = PEEK (I):
   GOSUB 50: NEXT : PRINT : GOTO
   1440
1430 GOSUB 50:N = PEEK (8196):
   GOSUB 50: PRINT "◊" ;: FOR I =
   8195 TO 8192 STEP - 1:N = PEEK
   (I): GOSUB 50: NEXT : PRINT
1440 M = 0: FOR I = 8195 TO 8192
   STEP - 1:M = M + PEEK (I) * 16
   ^ (2 * ABS (8192 - I)): NEXT :B
   = B + M
1450 RETURN
2000 REM dump segment header
2005 FOR I = 0 TO 30:PC(I) = 0:
   FOR J = 0 TO 1:LC(I,J) = 0:
   NEXT : NEXT :SS = 0
2010 PRINT D$ "BLOAD◊" FI$
   ",A$2000,L$63,B" B
2020 FOR I = 0 TO 99:X(I) = PEEK
   (8192 + I): NEXT
2030 B = B + X(42) + X(43) * 256
2035 IF CD THEN PRINT D$ "write"
   F1$
2040 PRINT HDR$(1);: FOR I = 3 TO
   0 STEP - 1:N = X(I): GOSUB 50:
   NEXT : PRINT : REM bytecount
2050 PRINT HDR$(3);: FOR I = 7 TO
   4 STEP - 1:N = X(I): GOSUB 50:
   NEXT : PRINT : REM resspace
2060 PRINT HDR$(4);: FOR I = 11 TO
   8 STEP - 1:N = X(I): GOSUB 50:
   NEXT : PRINT : REM length
   (real)
2070 PRINT HDR$(5) CHR$ (8) "na"
2080 PRINT HDR$(6);:N = X(14):
   GOSUB 50: PRINT : REM numlen
2090 PRINT HDR$(7);:N = X(15):
   GOSUB 50: PRINT : REM version
2100 PRINT HDR$(8);: FOR I = 19 TO
   16 STEP - 1:N = X(I): GOSUB 50:
   NEXT : PRINT : REM banksize
2110 PRINT HDR$(9);: FOR I = 21 TO
   20 STEP - 1:N = X(I): GOSUB 50:
   NEXT : PRINT : REM kind
2120 PRINT HDR$(10);: FOR I = 25
   TO 22 STEP - 1:N = X(I): GOSUB
   50: NEXT : PRINT : REM org
2130 PRINT HDR$(11);: FOR I = 29
   TO 26 STEP - 1:N = X(I): GOSUB
   50: NEXT : PRINT : REM align
2140 PRINT HDR$(12);:N = X(30):
   GOSUB 50: PRINT : REM numsex
2150 PRINT HDR$(13) CHR$ (8) "na"
2160 PRINT HDR$(14);: FOR I = 35
   TO 34 STEP - 1:N = X(I): GOSUB
   50: NEXT : PRINT : REM segnum
2170 PRINT HDR$(15);: FOR I = 39
   TO 36 STEP - 1:N = X(I): GOSUB
   50: NEXT : PRINT : REM entry
2180 PRINT HDR$(16);: FOR I = 41
   TO 40 STEP - 1:N = X(I): GOSUB
   50: NEXT : PRINT : REM dispname
2190 PRINT HDR$(17);: FOR I = 43
   TO 42 STEP - 1:N = X(I): GOSUB
   50: NEXT : PRINT : REM dispdata
2200 PRINT HDR$(18);: FOR I = 44
   TO 53: PRINT CHR$ (X(I));: NEXT
   : PRINT : REM loadname
2210 PRINT HDR$(19);: IF X(54) < >
   0 THEN FOR I = 55 TO X(42) - 1:
   PRINT CHR$ (X(I));: NEXT :
   PRINT : REM segname
2220 PRINT : PRINT
2230 RETURN
3000 REM label stuff
3010 DATA "Byte◊count◊◊◊:0$"
3020 DATA "Block◊count◊◊:0$"
3030 DATA "Reserved◊space:0$"
3040 DATA "Length◊◊◊◊◊◊◊:0$"
3050 DATA "Label◊length◊◊:0$"
3060 DATA "Number◊length◊:0$"
3070 DATA "Version◊◊◊◊◊◊:0$"
3080 DATA "Bank◊size◊◊◊◊:0$"
3090 DATA "Kind◊◊◊◊◊◊◊◊◊:0$"
3100 DATA "Org◊◊◊◊◊◊◊◊◊:0$"
3110 DATA "Alignment◊◊◊◊:0$"
3120 DATA "Number◊sex◊◊◊:0$"
3130 DATA "Language◊card◊:0$"
3140 DATA "Segment◊number:0$"
3150 DATA "Segment◊entry◊:0$"
3160 DATA "Disp◊to◊names◊:0$"
3170 DATA "Disp◊to◊body◊◊:0$"
3180 DATA "Load◊name◊◊◊◊:◊"
3190 DATA "Segment◊name◊◊:◊"
```

Checksums

```
10-$BADD   740-$9DD0  1440-$7FFF
20-$9B13   800-$4DC3  1450-$EA21
30-$363F   810-$3244  2000-$0A16
40-$610D   820-$184C  2005-$08FF
50-$FDAE   830-$C96A  2010-$3CBE
60-$80FE   840-$8FC3  2020-$2C14
70-$AF8B   850-$EA77  2030-$AA18
80-$752A   860-$3D61  2035-$9425
90-$A85E   900-$A89B  2040-$42AB
95-$0A28   910-$B2BE  2050-$FB22
100-$0C2B  920-$001A  2060-$6BAB
110-$D31C  930-$CE76  2070-$2FC5
120-$C500  940-$D84F  2080-$5767
130-$1989  1000-$A658 2090-$9269
135-$4E39  1010-$9C8A 2100-$BEC2
140-$A54C  1020-$468C 2110-$E541
145-$6384  1030-$6EBC 2120-$4A8C
150-$8EC1  1040-$C7DA 2130-$B125
160-$E038  1100-$020C 2140-$1DA5
170-$6BDA  1105-$B7D7 2150-$AEEC
180-$71AE  1110-$0B4F 2160-$2E41
190-$509C  1120-$C992 2170-$7D40
200-$20CD  1130-$5B5B 2180-$3112
210-$D668  1140-$C3E2 2190-$021F
220-$9563  1150-$0722 2200-$3D0C
230-$2AE2  1160-$71F9 2210-$5931
240-$2A4D  1170-$F561 2220-$7425
250-$4320  1180-$B075 2230-$6C89
260-$A539  1190-$80A7 3000-$8C30
270-$7D80  1200-$52FE 3010-$0F8C
280-$EFB3  1205-$C1DD 3020-$46CD
290-$652C  1210-$7DC1 3030-$993A
300-$86A6  1215-$C58C 3040-$1943
500-$6E98  1220-$1B1E 3050-$154A
510-$486A  1225-$6681 3060-$9B17
520-$EC37  1230-$886D 3070-$FD86
530-$5C6A  1240-$227B 3080-$D49E
540-$E694  1250-$9D17 3090-$B40A
600-$2311  1260-$99FF 3100-$DE8A
610-$12DA  1270-$9BD1 3110-$CC06
620-$3DA0  1280-$8FDF 3120-$1703
630-$F9B7  1290-$F2B5 3130-$137C
640-$1769  1300-$2B36 3140-$5FA3
700-$3264  1310-$7785 3150-$40BC
710-$1306  1400-$73B1 3160-$B7F1
715-$1D0A  1410-$053B 3170-$C8EF
720-$DEA3  1420-$180F 3180-$9CD7
730-$82DD  1430-$A199 3190-$1855
```

Managing OMFR Output

SCRIPT.ENTRY and SCRIPT.LABELS are variations on the same program. SCRIPT.ENTRY makes a script file for the ORCA Disassembler to automate the process of applying intersegment referenced labels and ENTRY statements. SCRIPT.LABELS makes a script file that will automate the labeling process for data segments that will be referenced with USING statements. The script files that are output use the "seg" command to switch to the appropriate segment in the ORCA/disassembly.

To create input files for SCRIPT.ENTRY, SCRIPT.LABELS and XREFFER, you load the RELOC and INTERSEG files generated by OMFR into the Appleworks database. There are returns between each category and there are 2 categories. Category 1 is the address to be patched, and category 2 is the address reference.

To create input for the SCRIPT.ENTRY program: Sort cat.2 of INTERSEG. Delete records where cat.2 references DATA areas and print to disk. You should print cat.2 with no headings. The resulting file is then used by the SCRIPT.ENTRY program. Only one label is created even if there are multiple references to an address.

To create input for the SCRIPT.LABELS program: Combine INTERSEG and RELOC and sort cat.2. Delete records where cat.2 DOES NOT reference DATA areas (in other words, keep only the records that reference data areas) and print (to disk) cat.2 with no headings. The resulting file is then used by the SCRIPT.LABELS program. If you EXEC the resulting script before trying to lay out a DATA segment the labels will appear when areas are defined properly.

To create input for the XREFFER program: Combine INTERSEG and RELOC and sort cat.2 and cat.1 (within cat.2) and print to disk. You should print cat.2 and then cat.1 with returns between categories and no headings. The resulting file is then used by the XREFFER program. The only real shortcomings of the resulting XREF are that only labels referenced in relocation records are included (no relative labels) and the patch address is usually off by 1 byte, because it's the address of an instruction operand and not the address of the instruction. XREFFER, with FI (line 40) set to 0, is useful for reviewing the pile of numbers. Set FI to a non-zero value and XREFFER will put the whole mess into a text file so it can be printed (on both sides of a page) with MCP (described later).

SCRIPT.ENTRY

```
10 REM script.entry
20 FI$ = "/ram5/intersegstuf"
30 FO$ = "/ram5/int.script"
60 D$ = CHR$ (4)
70 PRINT D$ "open◊" FI$: PRINT D$
   "open◊" FO$
75 PRINT D$ "write" FO$: PRINT
   "script◊entlabels"
80 ONERR GOTO 160
90 PRINT D$ "read◊" FI$
100 INPUT A$:S$ = MID$ (A$,1,2):
   PRINT D$ "write◊" FO$: PRINT
   "seg◊" S$
110 PRINT D$ "read◊" FI$
120 INPUT B$: IF A$ = B$ THEN 120
130 SB$ = MID$ (B$,1,2): PRINT D$
   "write◊" FO$
140 IF MID$ (A$,3,4) = "0000" THEN
   PRINT "start◊" MID$ (A$,3,4)
   ",L" A$:A$ = B$: GOTO 150
145 PRINT "entry◊◊" MID$ (A$,3,4)
   ",L" A$:A$ = B$
150 IF S$ < > SB$ THEN PRINT
   "seg◊" SB$:S$ = SB$
155 GOTO 110
160 PRINT D$ "write◊" FO$: IF S$ <
   > SB$ THEN PRINT "seg◊" SB$
170 PRINT "entry◊◊" MID$ (B$,3,4)
   ",L" B$
175 PRINT "ends"
180 CALL - 3288: PRINT D$ "close"
```

Checksums

```
10-$BADD   90-$DF1A   150-$1E59
20-$F40C   100-$5376  155-$C580
30-$3C93   110-$DFEB  160-$B778
60-$F9D5   120-$694E  170-$F611
70-$8F40   130-$18D6  175-$744C
75-$61B2   140-$644F  180-$E443
80-$B876   145-$01CB
```

SCRIPT.LABELS

```
10 REM script.labels
20 FI$ = "/ram5/labelstuf"
30 FO$ = "/ram5/data.script"
60 D$ = CHR$ (4)
70 PRINT D$ "open◊" FI$: PRINT D$
   "open◊" FO$
75 PRINT D$ "write" FO$: PRINT
   "script◊datalabels"
80 ONERR GOTO 160
90 PRINT D$ "read◊" FI$
100 INPUT A$:S$ = MID$ (A$,1,2):
   PRINT D$ "write◊" FO$: PRINT
   "seg◊" S$
110 PRINT D$ "read◊" FI$
120 INPUT B$: IF A$ = B$ THEN 120
130 SB$ = MID$ (B$,1,2): PRINT D$
   "write◊" FO$
140 PRINT "label◊◊" MID$ (A$,3,4)
   ",L" A$:A$ = B$
150 IF S$ < > SB$ THEN PRINT
   "seg◊" SB$:S$ = SB$
155 GOTO 110
160 PRINT D$ "write◊" FO$: IF S$ <
   > SB$ THEN PRINT "seg◊" SB$
170 PRINT "label◊◊" MID$ (B$,3,4)
   ",L" B$
175 PRINT "ends"
180 CALL - 3288: PRINT D$ "close"
```

Checksums

```
10-$BADD   90-$4598   155-$AFDB
20-$925E   100-$2960  160-$0D1F
30-$5BD4   110-$C9BD  170-$D5BA
60-$2082   120-$7F18  175-$891F
70-$3645   130-$A2F4  180-$13EC
75-$77E4   140-$DD28
80-$0EEC   150-$AAE6
```

XREFFER

```
10 REM xreffer
20 FI$ = "/blank4/dmxr"
30 FO$ = "/ram5/dm.xref"
40 FI = 1
60 D$ = CHR$ (4)
70 PRINT D$ "open◊" FI$: IF FI
   THEN PRINT D$ "open◊" FO$
80 ONERR GOTO 160
90 PRINT D$ "read◊" FI$
100 INPUT A1$:A1$ = MID$ (A1$,1,2)
   + "/" + MID$ (A1$,3,4): INPUT
   A2$:A2$ = MID$ (A2$,1,2) + "/"
   + MID$ (A2$,3,4)
```

```
105 GOSUB 200: PRINT A1$ "◊-◊" A2$
  "◊" ;:X = 1
110 PRINT D$ "read◊" FI$
120 INPUT B1$:B1$ = MID$ (B1$,1,2)
  + "/" + MID$ (B1$,3,4): INPUT
  B2$:B2$ = MID$ (B2$,1,2) + "/"
  + MID$ (B2$,3,4)
125 IF A1$ < > B1$ AND LEN (B1$) >
  0 THEN X = 1: GOSUB 200: PRINT
  : PRINT B1$ "◊-◊" B2$ "◊" ;:A1$
  = B1$: GOTO 110
130 IF X = 8 THEN X = 0: GOSUB
  200: PRINT : PRINT "◊" ;: REM 10
  blanks
140 X = X + 1: GOSUB 200: PRINT
  B2$ "◊" ;:A1$ = B1$: GOTO 110
160 PRINT
180 CALL - 3288: PRINT D$ "close"
190 END
200 IF FI THEN PRINT D$ "write"
  FO$
210 RETURN
```

Checksums

```
10-$BADD    90-$0A5B   140-$4740
20-$D282   100-$C64A   160-$4216
30-$1136   105-$DD90   180-$6619
40-$7A42   110-$39A9   190-$35AF
60-$34F3   120-$26A5   200-$27BF
70-$12A9   125-$526E   210-$8250
80-$53A3   130-$A3FD
```

Multi-Column Print Utility (MCP)

Lately I've had to deal with lots of numbers and I hated the idea of wasting a whole page of paper on a single column of data. I did the "Appleworks shuffle" (text to database to DIF file to spreadsheet) a couple of times and said..., well, never mind, but it wasn't pleasant. So, I did the program "mult.col.prnt", or "mcp" for short.

Occasionally, I have software revision notes, VERY long listings or the output from one or more of these other programs to print, and I want to print on both sides of the page. This is partly a desire to save trees and paper (could it be that I'm ... cheap?) and also as a matter of convenience. Since my printer won't do the job automatically (how many PC printers print both sides of a page simultaneously?) I added the capability to MCP to print either even or odd pages. I have to turn the paper around in the printer after printing the odd numbered pages and then print the even numbered pages on the back, but it makes a nice manual.

I've gotten more use out of this little routine than I ever imagined.

I think the easiest way to explain MCP is by explaining the variables that control its operation.

```
40 REM nl=# lines,nc=#
  cols,w=col.width
50 NL = 60:NC = 1:W = 79
```

The variables NL, NC, and W are set in line 50. NL is the number of lines to print per page, NC is the number of columns to print per page, and W is the maximum width of the columns. NL and NC also control the dimension of the DT$ data array. The above example is set for printing a document. To print data with a maximum size of, say, 7 bytes, set W to 9 and set NC to 8. If NL is 60 you will print 8 * 60 (480) values on a page. If NC * W is less than 80 I preview the output on the screen. In fact, NC * W should be less than 80 for "screening" to prevent the possibility of trashing part of the BASIC program.

```
60 REM eo=even/odd,sp=skip
  pg.,pn=page #,lm=left marg
70 EO = 0:SP = 0:PN = 0 :LM = 0
```

Next, SP, EO, PN, and LM are initialized in line 70. SP and EO are boolean values. If SP is not zero every other page will be printed. EO determines whether Even or Odd pages are printed. A non-zero (true) EO will cause even pages to print. It's best to print the odd numbered pages first. Using a small value for NL lets you watch the even/odd behavior on the screen. PN is the page number and is used to determine if you're dealing with an even or odd page. It is printed at the bottom of the page in line 260, but it's not necessary (to print it). LM is the left margin, and it's used when the text needs to be shoved over a bit.

```
90 FI$ = "inputfilename"
```

Line 90 is obvious. It should be edited to name the file to be printed. If you want to print an APW/ORCA SRC type file, change line 110 as follows:

```
110 PRINT D$"open "FI$",t$b0"
```

When everything lines up to my satisfaction I "de-REM" line 95 and include any printer init commands I need. I also add a PRINT command to line 200 just before the loop and a page eject to line 180 immediately following the page number. The lines will look like this:

```
200 PRINT : FOR I = 1 TO NL
260 PRINT : PRINT PN CHR$(12)
```

While MCP is not directly useful in disassembling programs, I find it to be a great little utility for dealing with all the source code and numbers generated in the process.

MCP

```
10 REM multi-column print
20 REM by James A. Hodge - 9/19/90
30 D$ = CHR$ (4): GOSUB 300
40 REM nl=# lines, nc=# cols,
  w=col.width
50 NL = 60:NC = 1:W = 79
60 REM eo=even/odd,sp=skip
  pg.,pn=page #,lm=left margin
70 EO = 0:SP = 0:PN = 0:LM = 0
80 DIM DT$(NL,NC - 1)
90 FI$ = "/dmstuf/dm.xref"
95 REM ? d$ "pr#1" :?chr$(9)
  "100N" chr$(29)
100 ONERR GOTO 190
110 PRINT D$ "open◊" FI$
120 PRINT D$ "read◊" FI$
130 N = 0
140 FOR I = 0 TO NC - 1: FOR J = 1
  TO NL
150 & GET DT$(J,I):N = N + 1
160 NEXT : NEXT :JH = FRE (0)
170 PN = PN + 1: IF SP THEN IF (EO
  AND (PN / 2 < > INT (PN / 2)))
  OR ( NOT EO AND (PN / 2 = INT
  (PN / 2))) THEN 130
180 GOTO 200
190 EF = 1:PN = PN + 1: IF SP THEN
  IF (EO AND (PN / 2 < > INT (PN
  / 2))) OR ( NOT EO AND (PN / 2
  = INT (PN / 2))) THEN 280
200 FOR I = 1 TO NL
210 IF I < = N THEN POKE 36,LM:
  PRINT MID$ (DT$(I,0),1,W);
220 FOR J = 1 TO NC - 1
230 IF I > N - NL * J THEN J = NC:
  GOTO 250
240 POKE 36,W * J + LM: PRINT MID$
  (DT$(I,J),1,W);
250 NEXT : PRINT : NEXT
260 PRINT : PRINT PN: PRINT
270 IF NOT EF THEN 120
280 CALL - 3288: REM clear stack
285 PRINT D$ "pr#0"
290 PRINT D$ "close" : END :#
300 FOR I = 0 TO 56: READ N: POKE
  768 + I,N: NEXT
310 POKE 1013,76: POKE 1014,0:
  POKE 1015,3: RETURN
320 DATA 32,177,0,32,227,223,162,0
  ,32,12,253,168,41,127,201,13,240,8,
  157,0
330 DATA 2,232,224,255,208,238,
  134,253,165,253,32,82,228,162,
  0,160,2,32,226,229
340 DATA 160,0,165,253,145,131,200
  ,165,111,145,131,200,165,112
  ,145,131,96
```

Checksums

```
10-$BADD   120-$C3AE   240-$0E13
20-$9B13   130-$F2A9   250-$7BAA
30-$12E7   140-$3F66   260-$2C4A
40-$F909   150-$9D43   270-$4050
50-$7DF5   160-$79B2   280-$D4CB
60-$8BDC   170-$6C64   285-$838C
70-$440A   180-$47E0   290-$2463
80-$A00A   190-$C408   300-$5028
90-$0037   200-$0D18   310-$8457
95-$2426   210-$9797   320-$9187
100-$F5BE  220-$7968   330-$3005
110-$256A  230-$94AA   340-$97A4
```

ORCA/Disassembler Scripts

These utilities will enable you to see clearly how the data areas are laid out, but you will still have to define them by hand. To aid in the process I came up with several scripts for the ORCA Disassembler that help ease the task. They are:

script quiki1hex: layout repetitive 1 byte hex fields
script quiki2hex: layout repetitive 2 byte hex fields
script quiki4hex: layout repetitive 4 byte hex fields
script quikivarhex: layout repetitive variable len. DC fields
script quikivards: layout repetitive variable len. DS fields
script quiki2int: layout repetitive 2 byte int. fields
script quiki4int: layout repetitive 4 byte int. fields
script quikichar: layout repetitive variable len. char. fields

These scripts all work in a similar fashion. When they are EXECed they will ask for a starting address and then an ending address. Three of them (quikivarhex, quikivards, and quikichar) will also ask for the field length. Once you have supplied the appropriate information the script will then lay out the fields in the specified location. If you are trying to re-create an accurate source for a program these scripts save you from a lot of boring, repetitive manual labor. If you decide to get into some serious disassembly work you might want to consider adding these scripts to the existing DISASM .SCRIPTS file.

Scripts

```
script quiki1hex
input 'start addr:'
@a=@inpval
input 'end addr:'
@e=@inpval
.xxx
DC   @a.@a,h
@a=@a+1
if   @a>@e,.adios
go   .xxx
.adios
ends
```

```
script quiki2hex
input 'start addr:'
@a=@inpval
input 'end addr:'
@e=@inpval
.xxx
DC   @a.@a+1,h
@a=@a+2
if   @a>@e,.adios
go   .xxx
.adios
ends
```

```
script quiki4hex
input 'start addr:'
@a=@inpval
input 'end addr:'
@e=@inpval
.xxx
DC   @a.@a+3,h
@a=@a+4
if   @a>@e,.adios
go   .xxx
.adios
ends
```

```
script quikivarhex
input 'start addr:'
@a=@inpval
input 'end addr:'
@e=@inpval
input 'str. len.:'
@l=@inpval
.xxx
DC   @a.@a+@l-1
@a=@a+@l
if   @a>@e,.adios
go   .xxx
.adios
ends
```

```
script quikivards
input 'start addr:'
@a=@inpval
input 'end addr:'
@e=@inpval
input 'ds len.:'
@l=@inpval
.xxx
DS   @a.@a+@l-1
@a=@a+@l
if   @a>@e,.adios
go   .xxx
.adios
ends
```

```
script quiki2int
input 'start addr:'
@a=@inpval
input 'end addr:'
@e=@inpval
.xxx
DC   @a.@a+1,i2
@a=@a+2
if   @a>@e,.adios
go   .xxx
.adios
ends
```

```
script quiki4int
input 'start addr:'
@a=@inpval
input 'end addr:'
@e=@inpval
.xxx
DC   @a.@a+3,i4
@a=@a+4
if   @a>@e,.adios
go   .xxx
.adios
ends
```

```
script quikichar
input 'start addr:'
@a=@inpval
input 'end addr:'
@e=@inpval
input 'str. len.:'
@l=@inpval
.xxx
DC   @a.@a+@l-1,c
@a=@a+@l
if   @a>@e,.adios
go   .xxx
.adios
ends
```

Krakowicz

The Basics of Kracking (part 5)

Deprotection of Modified DOS Disks

In episode 4 of this series, we began a discussion of protection schemes which are based on modification of a standard Apple DOS. as we mentioned, there are many changes which can be made, and literally thousands of combinations of which can be used to thwart the standard copy programs. However, rather than dwelling on all the possible techniques, let's concentrate on the "shotgun" approach which works to the vast majority. Regardless of the modification technique used, most of these disks can be rendered copyable with some utility programs (both old and new).

In general, it is possible to identify disks with a modified DOS by the appearance of a BASIC prompt at the bottom of the screen during the boot. Some protectors have begun to by-pass the routine which outputs the prompt, but you can still guess that there's a modified DOS present if the boot sounds like a normal DOS boot, but the disk won't copy with COPYA (comparing the sounds made by the boot under different protection schemes can be very valuable after you have a fair amount of experience with a given publisher and his protection scheme. It can also be misleading; I know a lot of people who swore the long head move during the boot of the SSI RDOS disks was a nibble count, while it turned out to be nothing more than loading in a short program called "QWERTY" from Track 18-22).

The classic program for dealing with modified DOS's is called DEMUFFIN PLUS (will the real author please step forward someday to accept the thanks of the entire world of software unprotectors?), and it works in much the same way as Apple's

MUFFIN program. MUFFIN was written to read files from a DOS 3.2 disk and then write them out in 3.3 format. DEMUFFIN (and a similar product called "NIFFUM") were written to convert DOS 3.3 programs to 3.2 for the real diehards. DEMUFFIN PLUS operates on the same principle, but uses whatever DOS is in memory to read, then writes out to an initialized disk under 3.3 format. While this is a powerful utility, you must keep in mind that it is strictly based on DOS, and will only transfer programs which can be located from a catalog as normal Text, Binary, Integer or Applesoft files.

It is safe to say that more software has been unprotected with this utility than with any other, and probably more than with all others combined. It still finds frequent application today, so we'll take a little time here to describe several ways of using it.

In most cases, the classical technique which follows for using DEMUFFIN PLUS to krack a modified DOS disk is restricted to those with an Apple II (not a II+) or a ROM card or other modified F8 ROM which allows you to reset into the monitor (see The Basics of Kracking part 1).

1. Initialize a disk under DOS 3.3, then delete the HELLO program (just to be safe).
2. Load Demuffin Plus into memory at some midrange location.

BLOAD DEMUFFIN PLUS, A$6000

3. Boot the protected disk, and as soon as a prompt appears, hit RESET (generally about 3-4 seconds after the head clackety-clack).
4. From the monitor, move DEMUFFIN PLUS to its normal location.

803<6000.78FFM

6. Startup Demuffin Plus and follow the familiar FID-ish instructions for slot and drive usage.

803G

7. If you want the disk to autorun from the boot, determine the name of the HELLO program and enter it into track 1, sector 9, bytes 75-92 with the INSPECTOR (otherwise the disk will always look for an actual program named "HELLO"). If the hello program happens to be a binary file, change byte 42 in track 0, sector $0D to $34, or use $14 to EXEC a text file for the start.

In addition to hiding the prompt, a number of publishers have added routines which clear out memory during the boot, or look for specific data loaded in previously (the Plato series is a good example of this—without extensive and careful boot tracing, it is very difficult to get the DOS in memory intact). The following approach eliminates the need to reset into the monitor, and also gets around many of the routines being added to thwart those who would reset. Because there is no need to reset during the boot, this approach can be used on any flavor of Apple][(yes, nibblespock, even on a //e). Basically, the difference lies in using the command interpreter and file manager portions of a standard DOS, and adding to it the modified RWTS from the protected disk:

1. Initialize a disk as before.
2. Boot up a standard DOS disk, and BLOAD DEMUFFIN PLUS,A$4000 (just to be safe).
3. Unless you have inspector in ROM, BLOAD INSPECTOR,A$8800.
4. Using the INSPECTOR, read track 0, sector 1 through track 0, sector 9 into $7700 to $7FFF. Use Control-I after the first "R" command to speed the load (see the section below on changing location $B942 if you can't read the sectors with the INSPECTOR).
5. Get into the monitor, then move the RWTS that you just read in on top of the resident RWTS with B700<7700.7FFFM.
6. Move DEMUFFIN as before with 803<4000.58FFM.
7. Type 803G and proceed with the file transfer.
8. There are a few cases where just a little more intelligence is used to modify DOS after the program begins to run (usually the modification of prolog/epilog bytes, but sometimes a little bit more), and in this case you have to boot the disk and let it run a second or two before resetting. The inspector in ROM is a big help in a case like this: you can reset and scan the entire disk with the "shift+" keys to see if all the sectors can be read with the DOS currently in the system. If they can be, chances are good that you will be successful with the conversion. After resetting, save the entire DOS with D00<9D00.BFFFM, then boot up your standard disk and BLOAD DEMUFFIN PLUS. Put the modified DOS back with 9D00<D00.2FFFM, and proceed with steps 6 and 7 above.
9. Assuming that the original copy was good, and that no secondary protection was used, you should now have a COPYA version of the program. In many cases, it's possible to do the job with even less hassle than this, so let's look at what is (maybe) an even easier way.

Many of the RWTS modifications are fairly trivial, and the most common consist only of changing the prolog or epilog bytes for the address or data field. You can often produce an unprotected version of these disks by making a few-byte change to the RWTS in memory, and then running COPYA. The following disassembly contains the routines which read in the address and data fields, and which need to be modified to circumvent a large number of RWTS change schemes:

```
B8DC:A0 20      LDY  #$20
B8DE:88         DEY
B8DF:F0 61      BEQ  $B942
B8E1:BD 8C C0   LDA  $C08C,X
B8E4:10 FB      BPL  $B8E1
B8E6:49 D5      EOR  #$D5
B8E8:D0 F4      BNE  $B8DE
B8EA:EA         NOP
B8EB:BD 8C C0   LDA  $C08C,X
B8EE:10 FB      BPL  $B8EB
B8F0:C9 AA      CMP  #$AA
B8F2:D0 F2      BNE  $B8E6
B8F4:A0 56      LDY  #$56
B8F6:BD 8C C0   LDA  $C08C,X
B8F9:10 FB      BPL  $B8F6
B8FB:C9 AD      CMP  #$AD
B8FD:D0 E7      BNE  $B8E6
B8FF:A9 00      LDA  #$00
B901:88         DEY
B902:84 26      STY  $26
B904:BC 8C C0   LDY  $C08C,X
B907:10 FB      BPL  $B904
B909:59 00 BA   EOR  $BA00,Y
B90C:A4 26      LDY  $26
B90E:99 00 BC   STA  $BC00,Y
B911:D0 EE      BNE  $B901
B913:84 26      STY  $26
B915:BC 8C C0   LDY  $C08C,X
B918:10 FB      BPL  $B915
B91A:59 00 BA   EOR  $BA00,Y
B91D:A4 26      LDY  $26
B91F:99 00 BB   STA  $BB00,Y
B922:C8         INY
B923:D0 EE      BNE  $B913
B925:BC 8C C0   LDY  $C08C,X
B928:10 FB      BPL  $B925
B92A:D9 00 BA   CMP  $BA00,Y
B92D:D0 13      BNE  $B942
B92F:BD 8C C0   LDA  $C08C,X
B932:10 FB      BPL  $B92F
B934:C9 DE      CMP  #$DE
B936:D0 0A      BNE  $B942
B938:EA         NOP
B939:BD 8C C0   LDA  $C08C,X
B93C:10 FB      BPL  $B939
B93E:C9 AA      CMP  #$AA
B940:F0 5C      BEQ  $B99E
B942:38         SEC
B943:60         RTS
B944:A0 FC      LDY  #$FC
B946:84 26      STY  $26
B948:C8         INY
B949:D0 04      BNE  $B94F
B94B:E6 26      INC  $26
B94D:F0 F3      BEQ  $B942
B94F:BD 8C C0   LDA  $C08C,X
B952:10 FB      BPL  $B94F
B954:C9 D5      CMP  #$D5
B956:D0 F0      BNE  $B948
B958:EA         NOP
B959:BD 8C C0   LDA  $C08C,X
B95C:10 FB      BPL  $B959
B95E:C9 AA      CMP  #$AA
B960:D0 F2      BNE  $B954
B962:A0 03      LDY  #$03
B964:BD 8C C0   LDA  $C08C,X
B967:10 FB      BPL  $B964
B969:C9 96      CMP  #$96
B96B:D0 E7      BNE  $B954
B96D:A9 00      LDA  #$00
B96F:85 27      STA  $27
B971:BD 8C C0   LDA  $C08C,X
B974:10 FB      BPL  $B971
B976:2A         ROL
B977:85 26      STA  $26
B979:BD 8C C0   LDA  $C08C,X
B97C:10 FB      BPL  $B979
B97E:25 26      AND  $26
B980:99 2C 00   STA  $002C,Y
B983:45 27      EOR  $27
B985:88         DEY
B986:10 E7      BPL  $B96F
B988:A8         TAY
B989:D0 B7      BNE  $B942
B98B:BD 8C C0   LDA  $C08C,X
B98E:10 FB      BPL  $B98B
B990:C9 DE      CMP  #$DE
B992:D0 AE      BNE  $B942
B994:EA         NOP
B995:BD 8C C0   LDA  $C08C,X
B998:10 FB      BPL  $B995
B99A:C9 AA      CMP  #$AA
B99C:D0 A4      BNE  $B942
B99E:18         CLC
B99F:60         RTS
```

Before we get into alterations of this code, let's get familiar with the terrain. There are two subroutines: "READ", which reads in a sector of data and lives from $B8DC to $B943; and "RDADR", which reads in the address field for a sector from $B944 to $B99F. Note that these are in the reverse order of their use in reading a sector. Let's look first at RDADR: after setting up some preliminaries at $B944-$B94E, we begin to look ($B94F-$B96C) for the three famous bytes of D5 AA 96 to identify the start of the field. After they are found, the volume number, track number, and sector number are stored in locations $2F, $2E, and $2D, respectively, and the checksum for the address field is verified ($B96D-$B98A). Finally, the two epilog bytes of DE and AA are sought at the end of the field ($B98B-$B99F).

After an address field is successfully read, "READ" is executed to read in the data field. The code from $B8DC to $B8FE finds the header bytes of D5 AA AD, and the data sector is read into a pair of buffers with the code at $B8FF-$B924 (the "nibblizing" process stored the 256 bytes from a page of memory as a total of 342 "nibbles" in the sector, but let's not get too worried about that yet). Finally, the checksum (one byte) is checked, and the epilog bytes are once again verified ($B925-$B941). Notice the innocent-appearing "SEC RTS" at $B942-$B943. This is the heart of the error-detection process, and most frequently modified (for our purposes) part of the entire routine.

The one byte which you should become most familiar with in order to do any kracking, snooping, or disk repair is the $38 at location $B942. The carry bit (of the processor status word) is used throughout the RWTS routines to indicate a Disk I/O error. Whenever anything goes wrong, the routines branch to $B942 to set the carry and return. The other routines in RWTS monitor the carry bit, and check it to see if there was a bad address read, a bad data read, no header bytes, wrong epilog bytes, etc., etc.

Note: The most important change you can learn to make is changing $B942 to $18 (or, if you are hopelessly BASIC-bound, POKE 47426,24).

The $18 is "CLC" or "clear the carry". By changing it, you are saying to the RWTS routines: "don't even look to see if there were any errors. Assume everything is all right and go on." This is obviously not a good general programming practice, since you're defeating all of the careful error checking that DOS does, but it's very handy to allow copying of a modified DOS. It will generally handle changes in the epilog bytes or intentional errors in the checksum of either field, but not in the header bytes. Header changes (because the bytes are individually checked for) must be done by modifying the appropriate code in the subroutine. In many cases, this is the only change which will be required to make a COPYA version of the disk.

For instance, let's suppose you are trying to krack a program, and you suspect that the protection consists of a modified DOS. Read in an entire track with the INSPECTOR or NIBBLES AWAY II (there is a bug in the shift-n command in some versions of the INSPECTOR—you can't do a nibble read on another track unless you first nibble-read in track zero). Examine an address field and its data field. If you find both "D5 AA 96" and "D5 AA AD", then remove the disk and boot up COPYA. While the program is asking for the slot and drive information, press reset or type CTRL-C. Delete line 70 (line 90 if you are using the integer version called "copy"), then from the monitor change

B942:18

Re-enter BASIC and run the program. Chances are very good that the result will be a COPYA version of the disk. Be aware, however, that you can propagate or generate errors in this process, since all of the error-checking in RWTS has been turned off. As always, check the program out thoroughly after kracking.

If your earlier snooping revealed non-standard header bytes, make the changes listed below after running and interrupting COPYA:

```
Address Field:  $B955 - BYTE #1
                B95F - BYTE #2
                B96A - BYTE #3
Data Field:     B8E7 - BYTE #1
                B8F1 - BYTE #2
                B8FC - BYTE #3
```

Then proceed as described earlier.

Regardless of whether you make these simple mods, or go through the DEMUFFIN PLUS process, bear in mind that secondary protection schemes can defeat these attempts and require you to do much more in the way of snooping and undoing. We'll pick up with a discussion of those techniques next time, and perhaps begin to explore some non-standard disk formats.

Our quotation of the week (month?) is from Don Lancaster, in the introduction to his book "Enhancing Your Apple II, Vol. 1" (a semi-good but seriously "stretched" compilation of little hardware tricks to make your Apple do new things):

"Any attempt at copy protection will hack off and inconvenience your legitimate users, and it will dramatically increase the number of bootleg copies in circulation...

The big thing about copy protection is that it doesn't. A year's effort by a cracker-jack military cryptography team can usually be undone in fifteen minutes, between Klingon zappings, by your average fourteen-year-old. And, morality and economics aside, one fact stands out... undoing copy protection is fun!

Not only is it fun, but cracking the uncopyable is about the most challenging and rewarding thing that you can possibly do with your Apple. and, the things you learn along the way are exactly the skills that you will need to become a really great programmer. So, I guess we should all be thankful for the copy-protection people since they are giving us all this fascinating entertainment and superb training at an unbeatable price."

Beautifully put, Don; an excellent rendition of the "Krackist's Manifesto."

The Basics of Kracking Part 6

Mating Zone & Nibblizing Mysteries

Congratulations are due to Tom Luhrs and the people at DATAMOST, for providing both an enjoyable game and an enjoyable challenge in kracking their latest

offering: "MATING ZONE". The game is definitely above average for a shoot-em-up, with a novel concept and good variety in the behavior of mated pairs, exploding eggs, and multiple levels. The krack is a little more difficult than the modified DOS's we have been discussing, but we are still dealing with a relatively standard RWTS.

As supplied, the game can be copied with NIBBLES AWAY II (no parms needed) for tracks 0-F and 10.5 to 13.5. except for the half tracking, the only deviation from normal DOS 3.3 sector structure is an epilog of "DF AA" instead of the normal "DE AA", for both the address and data fields. The disk access is controlled by an abbreviated RWTS loaded across screen memory, with a cute little surprise at the end. I'll start with a synopsis of the kracking process, and explain the juicy parts in detail later. The description will be limited to reducing the game to a COPYA disk; however, I strongly suspect that the game can be stuffed into a single long BFILE (I know, I thought that SIGMA 7 could be, too, but that's another story for a later time).

The kracking sequence is to first relocate the half-tracks, then eliminate the secondary protection. Among other utilities, NIBBLES AWAY II (NA II) can be used for the move as follows:

1. Boot NA II and copy tracks 0-F onto an initialized disk (you'll want the other tracks copyable later).
2. Select the track/bit editor (T), then read in track 10.5. type "Z" to allow NA II to analyze the track for write-out.
3. Change the track to 10 even, then insert the copy disk. type "W" to write, then "Y" to confirm. What was on track 10.5 of the original is now on track 10 of your copy.
4. Repeat steps 2 & 3 for tracks 11.5, 12.5, and 13.5.
5. Tell the program that the tracks have been changed by modifying track 1,sector F, byte 19 from $1F to $1E. Correct the epilog byte check by changing $DF to $DE in bytes 35 and AB of T0,S5; and byte 9B of T0,SD. .LI Eliminate the secondary protection and the high score write to disk by changing the following bytes:
6. Track $00, sector $05, bytes $E8-EA, change to 4C B5 04. Track $04, sector $0C, byte $38, change to 60.
7. Load up COPYA, defeat the checksum by changing the byte at $B942 to 18, then make a copy of the disk.
8. Boot and enjoy.

That's the procedure, now let's go over the theory: normally, RWTS occupies the memory space from $B700 to $BFFF. In order to squeeze it into $400-7FF, compromises must be made. First, a minimum of one complete track is read in, and the sectors are destined for sequential pages in memory, but without the interleaving used by DOS 3.3. Tracks are referred to by the equivalent number of half-tracks: track 6 is C, F is 1E, 10.5 is 21, etc. The track read routine increments the track number by two, then reads in the 16 sectors of the new track. Examining the code from $4DC-$55D shows a normal data field read routine with standard post-nibblizing to reconstruct the original bytes:

```
04DC:A0 20      LDY  #$20
04DE:88         DEY
04DF:F0 7B      BEQ  $055C
04E1:AD EC C0   LDA  $C0EC
04E4:10 FB      BPL  $04E1
04E6:49 D5      EOR  #$D5
04E8:D0 F4      BNE  $04DE
04EA:EA         NOP
24EB:AD EC C0   LDA  $C0EC *
04EE:10 FB      BPL  $04EB
04F0:C9 AA      CMP  #$AA
04F2:D0 F2      BNE  $04E6
04F4:A0 56      LDY  #$56
04F6:AD EC C0   LDA  $C0EC *
04F9:10 FB      BPL  $04F6
04FB:C9 AD      CMP  #$AD
04FD:D0 E7      BNE  $04E6
04FF:A9 00      LDA  #$00
0501:88         DEY
0502:84 26      STY  $26
0504:AC EC C0   LDY  $C0EC *
0507:10FB       BPL  $0504
0509:59 00 07   EOR  $0700,Y
050C:A4 26      LDY  $26
050E:99 00 03   STA  $0300,Y
0511:D0 EE      BNE  $0501
0513:84 26      STY  $26
0515:AC EC C0   LDY  $C0EC *
0518:10 FB      BPL  $0515
051A:59 00 07   EOR  $0700,Y
051D:A4 26      LDY  $26
051F:99 00 02   STA  $0200,Y
0522:C8         INY
0523:D0 EE      BNE  $0513
0525:AC EC C0   LDY  $C0EC *
0528:10 FB      BPL  $0525
052A:D9 00 07   CMP  $0700,Y
052D:D0 2D      BNE  $055C
052F:AD EC C0   LDA  $C0EC *
0532:10 FB      BPL  $052F
0534:C9 DF      CMP  #$DF
0536:D0 24      BNE  $055C
0538:EA         NOP
0539:AD EC C0   LDA  $C0EC *
053C:10 FB      BPL  $0539
053E:C9 AA      CMP  #$AA
0540:D0 1A      BNE  $055C
0542:A0 00      LDY  #$00
0544:A2 56      LDX  #$56
0546:CA         DEX
0547:30 FB      BMI  $0544
0549:B9 00 02   LDA  $0200,Y
054C:5E 00 03   LSR  $0300,X
054F:2A         ROL
0550:5E 00 03   LSR  $0300,X
0553:2A         ROL
0554:99 00 3F   STA  $3F00,Y
0557:C8         INY
0558:D0 EC      BNE  $0546
055A:18         CLC
055B:60         RTS
055C:38         SEC
055D:60         RTS
```

(* = These instructions start out as "$C08C", and have the slot-dependent value of "$C0EC" poked in at run-time. according to Mr. Slippery, "real men write self-modifying code!").

This is as it should be for all normal sector reading. at location $5E8, However, another data field read routine begins. This is very suspicious, indeed— why should they waste space on a ->SECOND<- data field read routine? Especially in a "squeezed" RWTS like this.

Remember the third law of kracking:

Acceptance of unusual code is no virtue; suspicion to the point of paranoia is no vice.

```
05E5:20 44 07   JSR  $0744
05E8:20 B9 04   JSR  $04B9
05EB:A0 20      LDY  #$20
05ED:88         DEY
05EE:F0 F8      BEQ  $05E8
05F0:AD EC C0   LDA  $C0EC
05F3:10 FB      BPL  $05F0
05F5:49 D5      EOR  #$D5
05F7:D0 F4      BNE  $05ED
05F9:EA         NOP
05FA:AD EC C0   LDA  $C0EC
05FD:10 FB      BPL  $05FA
05FF:C9 AA      CMP  #$AA
0601:D0 F2      BNE  $05F5
0603:EA         NOP
0604:AD EC C0   LDA  $C0EC
0607:10 FB      BPL  $0604
0609:C9 AD      CMP  #$AD
060B:D0 E8      BNE  $05F5
060D:A2 31      LDX  #$31
060F:EA         NOP
0610:86 26      STX  $26
0612:AC EC C0   LDY  $C0EC
0615:10 FB      BPL  $0612
0617:B9 00 07   LDA  $0700,Y
061A:9D 00 02   STA  $0200,X
061D:EA         NOP
061E:EA         NOP
061F:EA         NOP
0620:CA         DEX
0621:10 EF      BPL  $0612
0623:AD EC C0   LDA  $C0EC
0626:10 FB      BPL  $0623
0628:C9 DF      CMP  #$DF
062A:D0 BC      BNE  $05E8
062C:AD EC C0   LDA  $C0EC
062F:10 FB      BPL  $062C
0631:49 AA      EOR  #$AA
0633:D0 B3      BNE  $05E8
0635:A2 31      LDX  #$31
0637:5D 00 02   EOR  $0200,X
063A:CA         DEX
063B:10 FA      BPL  $0637
063D:0A         ASL
063E:D0 A8      BNE  $05E8
0640:4C B5 04   JMP  $04B5
```

Once again, the canonical prolog bytes of "D5 AA AD" are located, and then, strangely, only $31 (49 decimal) nibbles instead of the normal $156 (342 decimal) are read in before searching for the epilog of "DF AA". The bytes read in are EOR'd together, and if the result, shifted left once, is not zero, the read is redone. This is a very clever little anti-copy routine which works as follows: When the sector is read into memory from the disk, all $156 (342 decimal) nibbles are read in and postnibblized to reconstruct 256 bytes. These bytes are then re-nibblized and written out to the new disk being made. Since the original nibbles of DF and AA were not the result of a prenibblizing process, they will have disappeared as the 50th and 51st nibbles of the data field, and been replaced by the nibbles which result from a legitimate nibblizing and exclusive-oring process. One pass through any standard DOS sector-based copier will thus "destroy" the sector from the standpoint of the protection scheme.

The scheme, although a subtle and cute secondary protection system, is rather hollow, since nothing further is done with the checksum of the $31 nibbles. This way, as soon as the scheme is decoded, the routine can simply be bypassed with no penalty.

Now, I realize that this discussion brought many of you to the "MEGO" point (government-talk acronym for "my eyes glaze over") at the first use of the terms pre- and post- nibblizing. Those who know it all need read no further, but for those to whom this is still dialectic swahili, I will humbly offer my version of an explanation (in full knowledge that it may do no more than increase the eye-glaze coefficient). Once again, the primary source for this sort of exposition is "Beneath Apple DOS", Which by now has achieved the status of the most frequently-referenced text in all of Apple krackdom, if not Apple programming in general.

We're still not ready to go into the innermost workings of the disk storage process (that will be basics 107) but let's stipulate for the moment that there is a need to use only bytes which meet certain strict requirements when writing onto an Apple DOS 3.3 disk. The stone tablets carried up from CUPERTINO list those requirements:

1. The high bit of the byte must be "1"
2. The byte must contain no more than one pair of adjacent zeroes.
3. There must be at least two adjacent ones in the byte, not including the high bit.

As it turns out, there are exactly 64 bytes which meet all of these criteria. In order to store information on the disk, we must "encode" a total of 256 bytes (one page and also one sector) in the 64 pseudo-bytes which can be written.

64 different bytes means that we can set up a table in which each byte uniquely corresponds to one of the six-bit numbers from 00 to $3F (in binary, 0000 0000 to 0011 1111). the process of chopping up full 8-bit bytes into pieces which can correspond to 6-bit bytes is called "nibblizing". We can begin to simulate the "prenibblizing" process by making up two tables. the first one, which normally resides at $BB00-$BBFF, is set up to contain the first six bits of each of the 256 bytes:

If the corresponding original byte entry in the $BB00 table:

VALUE WAS:		TABLE IS:		
hex	binary	hex	binary	Table addr
00	0000 0000	00	0000 0000	BB00
3F	0011 1111	3F	0011 1111	BB01
47	0100 0111	07	0000 0111	BB02
69	0110 1001	29	0010 1001	BB03
7F	0111 1111	3F	0011 1111	BB04
85	1000 0101	05	0000 0101	BB05
BC	1011 1100	3C	0011 1100	BB06
F0	1111 0000	30	0011 0000	BB07
FF	1111 1111	3F	0011 1111	BB08

As you can see, in each case the first two bits have been chopped off and replaced with zeroes. The resulting byte, now between 0 and $3F (0 and 63) can be related, one for one, to the writeable bytes. However, if we don't store, in some organized fashion, those two bits we lopped off every byte, we won't be able to reconstruct the original bytes when we read these funny little 6-bit nibble bytes from the track. The way that's done is to construct a second table, normally at $BC00-$BC55, which contains all the little bits and pieces (ho-ho-ho) left over after the truncation of the original bytes to six bits. in the example list given above, the leftovers are:

```
00 ----------------\
00 --------------\   \
01 ------------\   \   \
01              \   \   \
01               \   \   \
10    This second \   \   \
10    list works   |   |   |
11    from the     |   |   |
11    bottom up,   |   |   |
      so the       |   |   |
      contents     |   |   |
      would be:    |   |   |
                   |   |   |
  ^       ^   ^    |   |   |
  |       |   |    |   |   |
  |       |   |    |   |   |
BC4D    00XX YY11   |   |   |
BC4E    00XX YY11   |   |   |
BC4F    00XX YY10   |   |   |
BC50    00XX YY10   |   |   |
BC51    00XX YY01   |   |   /
BC52    00XX YY01    /   /   /
BC53    00XX YY01 -/   /   /
BC54    00XX YY00 ---/   /
BC55    00XX YY00 -----/
```

So, the first two bits of the original byte become the last two bits of the bytes in this table, working from the bottom up. After $56 (86 decimal) bytes have had their first two bits stuffed into the table, the next one replaces the "YY" at location $BC55, Then at $BC54, etc. After $AC (172) bytes, the next pair of leftovers goes into the "XX" slot of location $BC55, and works up again until the last two bits are stuffed into the "XX" slot of location $BC00. Remember that the two most significant bits must always be zero to stay within the $0-$3F restriction.

After these two tables have been constructed, each value in the table is exclusive-ORed on with those that went before, to form a new six-bit byte. The resulting value, which is still between $0 and $3F, is translated to one of the 64 bytes which obey all the laws listed above for the disk bytes, and then really and truly written to disk. The table which does this conversion is called the "write translate table" and lives at $BA29 to $BA68. In our example, then the process goes like this:

The first byte

1. Get a byte from $BB00 = 00
2. Exclusive-or it with 00 (it's the first byte) 00 EOR 00 = 00
3. Look up the byte at $BA29 + 0 = 96
4. Write it to disk.

The next byte

1. Get the byte from $BB01 = $3F Exclusive-OR it with the previous value of $00. $3F EOR $00 = $3F
2. Look up the byte at $BA29 + $3F = $FF
3. Write it to disk

The third byte

1. Get the byte from $BB02 = $07, exclusive-or it with the previous value of 3F. (ie. 07 EOR 3F = 34)
2. Look up the byte at $BA29 + $34 = $F3
3. Write it to disk

The fourth byte

1. Get the byte from $BB03 = $29
2. Exclusive-or it with the previous value of $34. $29 EOR $34 = $1D
3. Look up the byte at $BA29 + $1D $CE
4. Write it to disk

and so on until the total of $156 or 342 bytes from the two tables is written to disk. (Looking at this process, you can see that a sector with all zeroes would never change the first byte written out, and would display a sector full of 96's on a nibble read).

When the data field of a sector is read back in, the process is reversed. after all 342 bytes are read into $BB00-$BC55, each byte is exclusive-ored off the pile, and the result is used to look up a value of $0-$3F in a "read translate table" at $BA96-$BAFF. Through some elegant, if intricate code, this 6-bit "byte" is recombined with its long-lost 2 bits, and the final, real byte is stored where $3E and $3F are pointing.

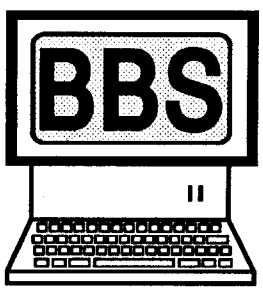
In the few remaining lines, let me expound for a moment on the exclusive-or operator. The instruction EOR (which uses the mnemonic XOR in every other assembly language) works like this: For each bit of the two bytes to be operated on, the output is a 1 if one and only one of the bits is 1, but a 0 if both are 0 or both are 1. the truth table below summarizes:

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

and the last example given above is:

```
        0010 1001   (29)
EOR     0011 0100   (34)
        0001 1101   (1D)
```

This is basically a neat little operator which has been frequently perverted by the enemy to do their dirty work. More on this later - stay tuned for the next episode: "non-standard encoding schemes."



Sysop - Dave Goforth

User #21

Ⓠ In "Leath Goddesses of Phobos", does anyone know how to get the cotton ball near the end of the game? I have tried everything I can think of. Help!

User #19

Ⓠ I typed in all of the source code for Rocket Ranger CDA but have run into a few problems. If someone could just put it up in the GS section of X-fer I'd appreciate it! I'm pretty sure I got it in right (I've proofread it about 10 times) and it's assembled and linked under ORCA. After installing it in the newly created DESK.ACCS subdirectory and booting, I get the message "Unable to load Desk Accessories $110B"...or something to that effect. Someone suggested my machine didn't have enough memory, but it runs fine off of a friend's machine.

User #2

To User #19: if it does run fine off your friends machine then something might not match between the two computers. What I'm saying is you might have something interfering with the program (such as a particular ram card). Try to find the differences between the two computers to come up with possible solutions to your problem.

For instance, on my friends computer the time is messed up when he runs GBBS. When the very same GBBS is run on my IIgs, we have no problem. We traced it to his 4 meg RAM card. Removing it solved the problem. And it created a new one, less memory.

I hope this helps you, if any.

Sysop

To User #19: The CDA will appear in the Computist Library soon (as sent in to Computist. I have not & cannot verify that it works, only that it was sent & published as is on the BBS.) All files published in Computist are available on the library disk. The library disk is NOT on the BBS however some items are posted. Space limitations (8 meg) restrict what can be on the BBS so the files are rotated & requests accepted via FEEDBACK.

Jack Moravetz (#12)

Running Teacher's Toolkit v3.1 (3.5") on a Laser 128

To Scott Jelsma: This softkey is for Teacher's Toolkit on the 3.5" disk. I've written about the various 5.25" disk protections previously, but they may have changed the protection since then. The original would not run on a Laser 128. HI TECH of Santa Cruz said the reason it wouldn't run was because of the difference between the Apple's ROMs and the Laser's ROMs. However, after defeating the copy-protection, it ran just fine on the Laser 128.

Softkey for...

Teacher's Toolkit version 3.1

Hi Tech of Santa Cruz

This program came on two 3.5" disks. It had a startup disk containing ProDOS and TOOLKIT3.SYSTEM and a program disk containing Multiple Choice, Word Scramble, Word Search, and Word Match files. The copy-protection was found in the file TOOLKIT3.SYSTEM on the startup disk. After making a copy of the startup disk, I tried booting it. I was greeted with a "This is not a valid 'HI TECH of SANTA CRUZ' STARTUP DISK" message. I ran the original and after resetting into the monitor, I found that the program would start from $21F0. I chose the area where I thought the copy-protection was being done and replaced the code there with a $4C F0 21.

Using a sector editor or block editor (Block Warden) search for the bytes $2C 00 C0 10 FB 2C 10 C0. Replace the $2C 00 C0 with $4C F0 21 and write the changes back to the disk. This code is in the file TOOLKIT3.SYSTEM on the startup disk.

It wasn't too difficult to place all the files on one disk, but it involved more than copying ProDOS and TOOLKIT3.SYSTEM to the program disk. For some reason you can't just copy the startup files to the program disk. I was met with a message that it wasn't the original startup disk. I copied all the files including the sub-directories first to RAM and then to a disk formatted with the volume name /HI.TECH.PROGRAM. The pathname isn't critical for the startup disk, but it is hard coded into the TOOLKIT3.SYSTEM file. Now I had a functioning Teacher's Toolkit that only required one 3.5" disk.

Softkey for...

ABM

?

I found ABM and Kabul Spy on an old cracking disk. I haven't seen either of these two programs, but they've both been on the Most Wanted list at one time or another. I hope you have good luck on the cassette tape transfers.

1. Boot disk and press ctrl-C before it finishes booting.
2. CATALOG disk (repeat step 1 if it won't catalog).
3. Transfer all programs using a tape recorder and the tape IO routines in BASIC. (LOAD Then SAVE, BLOAD->BSAVE) (Use the value at $AA61-AA62 for the length and the value at $AA72-AA73 for the start address for BSAVE) for cassettes save of binary, specify start and end of program (eg 800.1000W) and use 'R' for reading it back into memory.

Bitkey for...

Kabul Spy

?

1. Copy tracks 0-21 (Probably bit copy).
2. Use sector editor to edit:

Trk	Sct	Byte	From	To
00	00	$49-4B		EA EA EA

Softkey for...

Mickey's Crossword Puzzle Maker

Disney Software

Mickey's Crossword Puzzle Maker is a versatile crossword program that provides for creating, editing, and printing crossword puzzles with Disney characters. On the package it stated "No on-disk copy protection", but they did have a type of password protection that asked answers to certain crossword puzzles that were printed on red paper. This answer sheet was difficult to read and the company wanted $15.00 to replace a lost sheet.

The password protection wasn't my problem. When I tried to boot my 3.5" disk, ProDOS came up on the screen and the program hung. I tried booting the 5.25" disk and got the same results. I had a CMS hard drive card in slot 7 that I removed and the program booted and ran fine. I didn't want to have to remove the CMS card every time I wanted to use the program, so I began looking for a way for it to work. In the file LEGACY at byte $319 I found a JSR to $631D ($20 1D 63). Following the jump to the subroutine was an RTS ($60). The bytes at $631D were $AE 31 BF. By changing (Any one of these three changes) the $20 1D 63 to EA EA EA, changing the $20 to $60 in $20 1D 63, or changing the $AE to $60 in the bytes $AE 31 BF would allow the program to run with the CMS hard drive card in slot 7.

The password protection was still there though. Later in the file I found the area where I thought that the password protection was being done. This was at byte $C08. It started with a JSR $E075 and later in byte $C54 there was another JSR $E075. I began by changing the branching instructions and NOPing a bunch of code before I realized that all those checks were finally leading to a JMP to $6C65. All I did was replace the $20 75 E0 at byte $C08 in the file LEGACY with $4C 65 6C and the password protection was bypassed.

I called Disney software and asked them why it wouldn't run with the CMS hard drive card and why it would when I made those changes to the JSR to $631D, but they couldn't answer me then and never did get back to me.

Don't let it bother you Jack. Software companies don't like knowledgeable programmers unless they're working for them (and sometimes not even then)...RDEXed

Making PLATO software run on the Enhanced //e

I read with much interest Jim Hart's article in COMPUTIST #74 dealing with "Modifying Programs to RUN on a Laser 128". Ever since the Enhanced //e arrived, I have had several good educational programs that will not run properly. Plato software, for some reason that I never could figure out, will not run on an enhanced //e. It will boot to a certain point and then drop into the monitor or hang. I wondered if Jim Hart's method of making programs run on the Laser 128 would help me run Plato software on an enhanced //e. The answer is a definite YES! This can only be done on a deprotected Plato disk. The one that I used was WHOLE NUMBERS: Practice. In case you didn't read Jim Hart's article, here is enough to do this modification.

Whole Numbers: Practice

1. First, boot a DOS 3.3 disk on an Apple II+ and save the ROM code to a disk.

```
BSAVE II+ ROM, A$D000, L$3000
```

2. Create a TEXT file to do the modifications. Use a text editor and type these commands then save it as a normal ASCII text file. Call it "START.TEXT".

```
CALL -151
C081 N C081
BLOAD II+ ROM, A$D000
C080
3D0G
SPEED = 255
BRUN PB.BOOT
```

Replace PB.BOOT with whatever the original HELLO program name is.

3. Make a HELLO program to EXEC the text file from step 2 such as:

```
10 PRINT CHR$(4);"EXEC START.TEXT"
SAVE HELLO
```

COPY //+ will allow you to change the boot program also. Pronto DOS has an enhancement that will allow that DOS to EXEC a file on the boot. I'm hoping this modification will allow the enhanced //e to run other programs properly too. Thanks Jim Hart!

- End of BBS News -

Bob Cherochak AR

Ⓠ I've been trying to win at Task Force GS. Is there a cheat patch out there that will allow for more life points and more ammunition?

Notes on Task Force GS

How to run Task Force on your hard drive

1. Deprotect it from past issues of Computist.
2. Copy all the files and folders, from both disks, except ProDOS into a folder on your hard drive.
3. Using a Block/File Editor (I like Deliverance from Vitesse) to search for all occurrences of "/TF1/", "/TF2/", and "/TF_/". Replace these 5 bytes with spaces. More precisely 5 HEX 20's. Make sure you use HEX "20".
4. Save the file back to disk.

That's it! You should be able to launch "STARTUP.SYSTEM" from the Finder.

Task Force and Wings by Vitesse

If you are having trouble launching Task Force from your hard disk launcher (I did with Wings by Vitesse) then first check your previous work then try this:

1. Rename "STARTUP.SYSTEM" to "START.SYSTEM".
2. Type in this BASIC program and save it as "FINDER.MAKER".

FINDER.MAKER

```
10 FOR Z = 8192 TO 8313: READ ZZ:
   POKE Z,ZZ: NEXT
20 HOME : VTAB 9: INPUT "NAME◊OF◊
   SYSTEM◊FILE◊TO◊LAUNCH◊:" ;N$
30 IF LEN (N$) > 13 THEN PRINT
   "NAME◊TOO◊LONG◊13◊CHARACTERS
   ◊MAXIMUM◊" : CALL - 678: GOTO 20
40 POKE 8314, LEN (N$)
50 FOR N = 1 TO LEN (N$)
60 POKE (8314 + N), ASC ( MID$
   (N$,N,1))
70 NEXT N
80 S$ = LEFT$ (N$, LEN (N$) - 6)
90 PRINT CHR$ (4); "CREATE◊" ;S$;
   "FINDER,TSYS" : PRINT CHR$ (4);
   "BSAVE◊" ;S$; "FINDER,TSYS,
   A$2000,L" ;122 + N
```

```
100 DATA 162,0,189,16,32,157,0,3,
  232,224,128,144,245,76,0,3,32,
  24, 3,176,6,32,250,248,76,0,32,
  32,0,191,101,17,3,4,0,0,0,0,0
  ,0,32,0,191,200,85,3,176,35,
  173,90,3
110 DATA 141,92,3,141,102,3,32,0,
  191,209,101,3,176,18,173,103,3,
  141,95,3,173,104,3,141,96,3,32,
  0,191,202,91,3,8,72,32,0,191,
  204,99,3,170,104,40,176,4,138,
  240,1,56,96,3
120 DATA 106,3,0,8,0,4,0,0,32,0,0,
  0,0,1,0,2,0,0,0,0
```

Checksums

```
10-$DDA0    50-$8317    90-$B880
20-$FCB9    60-$E79E   100-$DE4D
30-$C241    70-$6F63   110-$3D65
40-$85B8    80-$FB53   120-$2AB1
```

To use it just load this program; set your prefix to where Task Force is located; and type "RUN". Once you answer the question as to what ".SYSTEM" file you want to launch, ("START.SYSTEM" in this case) the Finder.Maker will create a mini launcher that will get you up and running into your ".SYSTEM" program.

Remember: This will only launch ".SYSTEM" files with a filename of 13 characters or less including the ".SYSTEM" on the end. It works with Zany Golf (DOS.8.SYSTEM), and Battle Chess (CHESS.SYSTEM)

I got the original launcher from GENI as a "SYS" type file but it required a block editor to change the filename you want to launch. I kept messing up with the block editor and the 13 character limit so I converted the "SYS" file to a BASIC program that will catch my mistakes. Also there is no block editor required because this program writes a new launcher with the filename to launch included each time. I don't know who originally wrote the launcher but he deserves all the credit and a pat on the back.

Notes on Battle Chess

To J.C. in NJ: I have Battle Chess running off my Vulcan hard drive. To launch it from GS/OS first you must have the program located in a partition that IS accessible without your modified Vulcan P8 file meaning one of your first two partitions. Second you might need a launcher that will gracefully get out of GS/OS and run Battle Chess. I suggest the aforementioned launcher. You can reach me for more info on GENI or through the RDEX editor.

Notes on Silent Service GS

If anyone can figure out how to get ProDOS 8 V1.3 to run a ".SYSTEM" file from a subdirectory or get the SS.SYSTEM file from Silent Service GS to run under ProDOS 8 V1.9 we could save a lot of time and trouble in launching Silent Service GS from a folder on a hard drive. I tried all the other ProDOS versions I own and none of them will work with Silent Service except V1.3. I changed the global page version number of ProDOS 8 V1.9 at locations $BFFD to $01 and $BFFF to $03 but SS.SYSTEM caught up with me on the "War Patrols" screen and locked up.

For anyone out there who wants to run Silent Service GS from a folder on your hard drive with only three files in your root directory you could:

(1) Deprotect it using the back issues of Computist.

(2) Copy all the files to a folder called "SS.FILES" on your hard drive.

(3) Put a copy of Basic.System in the root directory and rename it but be sure it ends with ".SYSTER". example "Basic.Syster"

(4) Use a file editor to change the ".SYSTEM" name in the ProDOS file to ".SYSTER" the "R" is located at byte $4ED in the file ProDOS file. Change Byte $4ED (1261) From "CD" to "D2" or from BASIC type:

PREFIX/*your hard drive name* **/SS.FILES**
POKE 4096,210
BSAVE PRODOS, TSYS, A4096, L1, B1261

(5) Use a file/block editor to change the startup file name in the BASIC.SYSTEM file you renamed in step 3 to "SS.SETUP". Change Bytes $06 thru $0E from 07 53 54 41 52 54 55 50 00 (= length byte $07 + "STARTUP") to 08 53 53 2E 53 45 54 55 50 (= length byte $08 + "SS.SETUP)

Or from BASIC (make sure the renamed BASIC.SYSTEM is on the drive) type:

PREFIX/*your hard drive root volume name*
POKE 4096,8
POKE 4097,83
POKE 4098,83
POKE 4099,46
POKE 4100,83
POKE 4101,69
POKE 4102,84
POKE 4103,85
POKE 4104,80
BSAVE *renamed Basic.System* **, TSYS, A4096, L9, B6**

(6) Create this BASIC program and save it to the root directory with the name "SS.SETUP".

```
10 PRINT CHR$(4);"PREFIX": INPUT
  P$: PRINT CHR$(4);"PREFIX";P$ +
  "SS.FILES":PRINT CHR$(4);"-
  SS.SYSTEM"
```

That's it! Just launch the ProDOS file in the "SS.FILES" folder.

The reason why I decided to change the ".SYSTEM" to ".SYSTER" in the ProDOS file and change the name of the "STARTUP" file to "SS.SETUP" is to eliminate conflict with other programs in the root directory of your hard drive. This way only the ProDOS from Silent Service will end up launching the "SS.SYSTEM" file. If you have trouble with this or have a better way to run Silent Service let me know. I'm also on GENI.

Last minute addition:

Update on the Silent Service GS v925.01 crack

Yes you can run the whole thing from a folder on your hard drive. Finally! The only problem you will have (I hope) is if you are using "Wings" by Vitesse as your launcher. This will not work with Wings due to something in Wings itself. Probably the quit code it installs into memory. You can use the Finder, JumpStart, or Hyperlaunch that I know of. So here we go.

1. Copy all the files and folders from your original Silent Service disk to a folder on your hard drive. (Except PRODOS)
2. Get into BASIC.SYSTEM and BLOAD SS.SYSTEM into memory ($2000).

BLOAD *Pathname* **/SS.SYSTEM, TSYS, A$2000**
CALL-151
2106:80 **was F0**
2227:80 **was F0**
2BE6:60 **was A9**
BSAVE *Pathname* **/SS.SYSTEM, TSYS, A$2000, L$C7B**

That's it just use the finder to launch SS.SYSTEM and you're in.

For the technical side

The $80 at 2106 causes a branch always when the ProDOS Kernal is checked. This allows you to use whatever P8 file you want. The second $80 at 2227 defeats the checksum test and branches always as if everything is ok. The $60 at 2BE6 is the alternate method of defeating the protection (nibble count I think) from issue 67 of Computist. All this was possible through the efforts of others and my desire to get this program on a hard drive. Special thanks to James A. Hodge for coming up with the two branches in the first place and all the help he has given me.

Now can anyone out there whip up a short .SYSTEM file that will shut down Wings completely and launch SS.SYSTEM ? Even if you launch the Finder from Wings and then launch SS.System, Wings will still be there and will get you. After all you have to reboot to quit Silent Service.

Softkey for...

Algebra 1-6

Britanica

Here is a controller that I wrote to, once and for all, completely deprotect the Algebra series 1 thru 6 by Britanica (Edu-Ware). This controller works with Super IOB V1.5. As you can see it formats your destination disk, copies your original to it and renames the proper file to "HELLO" so that it is bootable under normal DOS 3.3.

Controller

```
1000 REM Algebra Series (1-6)
  Britanica Software
1010 HOME : VTAB 10: HTAB 7: PRINT
  "PLACE◊A◊BLANK◊DISK◊IN◊THE" :
  PRINT "" : HTAB 3: PRINT
  "DESTINATION◊DRIVE◊AND◊PRESS◊
  RETURN" ;: CALL - 678
1020 HOME : PRINT "" : PRINT CHR$
  (4); "INIT◊HELLO,S" ;S2; ",D"
  ;D2: POKE 47426,24
1030 TK = 3:LT = 35:ST = 15:LS =
  15:CD = WR:FAST = 1
1040 GOSUB 490: GOSUB 610
1050 GOSUB 490: GOSUB 610: IF PEEK
  (TRK) = LT THEN 1070
1060 TK = PEEK (TRK):ST = PEEK
  (SCT): GOTO 1040
1070 POKE 47426,56: ONERR GOTO
  1100
1080 HOME : VTAB 10: HTAB 8: PRINT
  "RENAMING◊THE◊STARTUP◊FILE" :
  PRINT CHR$ (4); "VERIFY◊EDU-
  WARE,S" ;S2; ",D" ;D2
1090 PRINT CHR$ (4); "UNLOCK◊EDU-
  WARE" : PRINT CHR$ (4); "RENAME
  ◊EDU-WARE,HELLO" : PRINT CHR$
  (4); "LOCK◊HELLO" : GOTO 1110
1100 PRINT CHR$ (4); "UNLOCK◊BOOT"
  : PRINT CHR$ (4); "RENAME◊BOOT,
  HELLO" : PRINT CHR$ (4); "LOCK◊
  HELLO"
1110 HOME : PRINT "" : VTAB 10:
  HTAB 8: PRINT "DEPROTECTION◊
  COMPLETED◊!" : END
```

Checksums

```
1000-$356B  1040-$672A  1080-$8A20
1010-$AEE8  1050-$0DD9  1090-$6CEF
1020-$9143  1060-$63FF  1100-$FE71
1030-$675E  1070-$5824  1110-$6683
```

John E. Wanner _____ **WI**

Copy-able Questron II

The first thing that surprised me was seeing both Questron II and Wasteland still on your most wanted list. As I stated in my walk-through for Questron II published in COMPUTIST #65, there is NO copy protection of any kind I've ever had problems with on this program. The original documentation states something about having to look up stuff in the manual, but in the 5 times I completed the game to verify the walk-through and half-dozen people I had test it for me, no one ever had any problem with this.

Softkey for...

Wasteland

Electronic Arts

Requirements:
Wildcard II with utilities disk
Fast DOS

Wasteland is a different story. I thoroughly enjoyed the game and wanted to be able to take it to several other places (i.e. friends' houses where we could work on it together) without dragging around the original disks. I tried several methods used to deprotect Electronic Arts software in Computist, but, frankly, I'm not too good with the subtleties of machine languages, etc. So I decided to use brute force.

Most of my current work is being done on a IIgs, but in '88 when Wasteland came out, I only had a IIe. This machine has a bunch of easily accessible slots and I had a Wildcard II installed in one of them. The following procedure will make a perfectly usable copyable version of the program. Unfortunately, it loses the rather cute opening sequence.

Step-by-step

1. Initialize a blank disk with fast DOS (I use Beagle Brothers Pronto-DOS; it's old but excellent) and put it aside for later use. I also find it handy to have as my boot program an auto-catalog DOS 3.3 program called S-S Catalog Program, but none of these specific programs is necessary; all you need is an initialized, blank disk.
2. Re-boot with the original Wasteland disk labeled BOOT #1. Enjoy the opening sequence, because it's the last you'll see of it (at least on the backup). Press U for the utilities section. When it comes up, you will notice that the menu also includes the START option you saw on the opening. Press the WILDCARD II button. Remove the Wasteland disk and insert a COMPLETELY BLANK (not initialized) disk. Select the SAVE 64K option from the WILDCARD menu. The card will save the memory to disk.
3. You now have a copy which will only work if you have a WILDCARD in your computer. To get one which will work on any 128K (That's not a misprint; you need the 128K.) Apple II machine, you need the utility disk that came with the NMI card. Remove the disk with the memory copy you just made, insert the utility disk & reboot. From the main menu, choose (2) MAKE 64K DOS COPY. Remove the utility disk, insert the memory copy disk you made in step 2 into drive 1 and the initialized disk you made in step 1 into drive 2, name your back-up file and press return. After a few minutes, you will have 3 files which will automatically load in the proper order, giving you a backup of the boot program. The remaining disks can be copied with any standard fast disk copier. (I use Locksmith 6.0 Fastcopy.)

That's it!

Playing tips for...

Elite

?

To Bob Igo (and incidentally, Jeff Hurlburt, too), issue #76, and any other interested parties. Your first goal should be to get money to outfit your ship better. At first, you'll have to buy cheap stuff, like food at agricultural planets and sell it at industrial planets. Work little by little into expensive items like returning with a few computers to the agricultural planets. Eventually, the best way I've found is by trading in contraband like narcotics. It seems that you generally pay less for them at industrial planets and more at agricultural ones, but the prices are truly random; once I bought some at a planet for 14 cr/tonne, hyperspaced out and for some reason returned immediately to the same planet. They were then worth 87 cr/tonne. Anyway, the best method is to find 2 planets very close together (less than 1 light year) and go back and forth, trading narcotics in one direction and whatever is low at the other on the return; furs and liquor are often the best deal going the other way. Eventually, the cops will chase you as your legal status deteriorates, but you can get around this. By the way, shooting cops doesn't seem to help your combat status much. If you're pursued before you've got a way out, jump to the nearest space station and they'll stop chasing you once you're inside the station's field. Oh, yes, and also DON'T attack the station or ships within its field if you value your lives.

As soon as you have enough cash, buy a beam laser—this will eventually take out almost any attacking enemy— then a large cargo bay, ECM system extra energy unit and a docking computer. It becomes a serious pain to mess around docking manually all the time, but be aware that the computer is NOT idiot-proof; if you line up wrong, you can still crash. Eventually, you should be able to get an escape pod, military lasers

and galactic hyperspace. Once you have the escape pod, when the cops get too nagging, you can leave a space station after converting all your cargo to cash, escape with the pod and your legal status is cleared. The pod is lost, but your ship will be resurrected with its full compliment of goodies. Then you can trade stuff like computers from the industrial to agricultural planets—less profit but less of a pain from the cops, too.

Save the game regularly and check your status. You will progress fairly steadily with your kills of pirate craft (and anything else you shoot, too, but I didn't want to mess with the cops again—probably my conservative nature—I just waited until I was attacked.) Be aware, too, that you can accidentally crash into another spacecraft while docking as it leaves the station and this can adversely affect your legal status, even if it's the other guy's fault.

To improve your fighting status, all you have to do is shoot down other spacecraft. The fastest way to do this is by finding an Anarchy planet; these almost always have a lot of pirates around and you will be attacked at each jump around the system. Since they sometimes come at you in large groups, it's a good idea to have your hyperspace locked onto a nearby planet and to have an energy bomb. Then go hunting. Your fighting status progresses roughly as follows:

Mostly harmless—25 kills
Poor—50
Average—75
Above average—125
Competent—750*
Dangerous—3500
Deadly—7500†
Elite—over 10,000†

* - There is a way to increase your speed of becoming DANGEROUS. When you reach COMPETENT, you will be offered a special mission—to kill a tough ship called the Constrictor. It is difficult to locate and eventually is found in the system of Orarra in Galaxy 2 (which I learned after searching a couple of dozen systems at least). You should have a military laser, but if you don't, you can also crash into it (you WILL survive, it won't) or fire a missile on a collision course at so close a range its ECM system won't have time to work. Just be aware that this latter tactic can work both ways. After this you will be promoted to DANGEROUS!

† - I've never gone past the DANGEROUS stage, but one of my students who borrowed my disk tells me that these numbers are about right. There is also a second mission (which I have completed) which apparently speeds up other ratings and gets you a neat energy cell as well. Anyway, good luck in your quest to become ELITE!

Notes on Wildcard II card

To Kris Kirk: (and anyone else who's got a WILDCARD II and doesn't know what to do with it): The card can be installed in any convenient slot except 0 or the IIe aux. slot; I put mine in slot #3 since some other cards don't work there. It can be used for a variety of things—see the menu when you push the button—but is most useful in backing up single-load programs. Examples of these would be PFS: FILE or ELITE. It's most powerful with the utility disk and the manual has some really useful tips and other info. I really respect Central Point Software, but if they can't supply any of the documentation or utility disk, please contact me and we'll talk about it. Nothing bugs me more than a company that makes a device and doesn't support it any more. The automobile manufacturers invented this idea decades ago; it's called planned obsolescence.

To Steve Kalynuik: First, a tip of my mage's cap and thanks for the tips on Wizardry 5; they will be really helpful when I have more time to play the game. Wizard's Workbench (they didn't tell me that there was a WWB II) used to be available from Magicsoft, PO Box 908-Z, Danville, CA 94526. The original WWB allowed you to construct your own scenario from the original Wizardry disk, Proving Grounds... You could also get a sort of sci-fi hybrid, Atomic Wasteland 2098 and another Wiz variation, Knight of the Grave from them. If Magicsoft can't help you, again, feel free to contact me about those goodies; if you've read my earlier rambling, you already know that companies who don't support their stuff do not make me happy. If you want to contact me, my address is:

W1786 Circle Drive
Sullivan, WI 53178

Chuck _____ CA

Has anyone been able to unprotect Math Facts Tracker or Mastery Arithmetic Games (new versions) by Mastery Development, Redmond, WA.

I would like to thank Wayne Zurow of Saudi Arabia for both his help and pointing me in a direction that would let me contribute the Softkeys that follow. I would like to also thank Floyd Splidnik from Computist #8, Page 6 for his description which I will use in using DEMUFFIN PLUS which is also in Computist #8, Page 15. Also I would like to thank the Computist for their description on the CDA (visit monitor) that is on the IIGS, for without it I would not have been able to use DEMUFFIN PLUS. I would also like to get together with anyone in the LA area who has some expertise in deprotection, as I would like to learn more from someone who would be willing to help. Please contact me through the Computist if you are interested.

Softkey for...

Who Am I?
Ocean Life
Animals of the Past
Birds Trees and Flowers
Mammals Reptiles and Insects
Your Body
Your Body: Series II
Skeletal/Muscular
Nervous/Endocrine
Chemistry: Series I
The Solar System
Water and Weather
Non-Western Cultures
India and Latin America
Africa/Middle East
China and Japan
Economics
Definitions and Laws
Systems
Capitalism, Communism and Socialism
Teacher's Class Demonstration
The Great Knowledge Race and the Data disks
US History
The Young Republic
The Time Tunnel
American History Series
The History of Europe
Writing Chemical Formulas
Focus
Integer Arcade
Graphmaster
HRM
Safari Search
Missing Links
Sunburst
Mind Benders
Midwest Publications
Cause and Effect
Learning Well
Special Product and Algebraic Factors
Hi Res Computer Golf
Avant Garde
Word Problems for Algebra
Metier
Worksheet Generator
Micro Learningware
Master Grades
Midwest Software
SAT Score Improvement Series
Go
Laser Bounce
Hayden

The general Deprotect procedure

1. Initialize as many disks in DOS that you will be deprotecting. I used Copy II+ to format and copy DOS to the disk I used.
2. Boot up your disk that you are going to deprotect. As soon as the prompt (]) shows up get into monitor. Some disks this will not show up, and the method will still work.
3. To give you an idea if this is going to work type A56EG to catalog your disk. If it catalogs, it will probably be able to be deprotected.
4. Move the RWTS tracks to a safe place so they will not be destroyed when DOS is booted.

6000<B800.BFFFM

5. Replace your protected disk with one of your formatted disks from step 1, and boot it up.

6 ctrl P

6. Load DEMUFFIN PLUS, do not run it.

BLOAD DEMUFFIN PLUS

7. Enter monitor.

CALL-151

8. Move the RWTS tracks back where they can be used by DEMUFFIN PLUS.

B800<6000.67FFM

9. Activate DEMUFFIN PLUS

803G

10. Follow the prompts after selecting #1 to convert the files. Use the wildcard (=) for the file name. Answer NO to prompting. Watch DEMUFFIN PLUS copy the files.
11. If there is a HELLO used on your protected disk for the boot program then you will be done. If not you will get a FILE NOT FOUND error on boot. At this point look for an APPLESOFT (A) file and try to run this. At this point you have two ways to go. 1. Change the boot program with Copy II+ 2. Write a short one line program to boot your disk

```
5 HOME:?CHR$(4);"RUN XXX"
```

(Where XXX is the name of the program that will boot your disk.)

That's all.

Softkey for...

Computer Generated Mathematics Vol. 2

MECC

Follow the general step by step procedure but copy file by file since it will get errors trying to read the Integer BASIC file. Don't copy this file, the program seems to work fine without it.

Brian A. Troha _____ WI

Softkey for...

Cribbage/Gin King

The Software Toolworks

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" disk editor

Cribbage/Gin King (CGK) is a great new game set from the Software Toolworks. This program allows you to play a game of cribbage or gin against your computer the only problem is; you need to answer a question before you can start to play the game. This is of course KEYWORD (pirate) protection. After checking the code from the CRIBKING file, I was able to bypass the routine. Simply applying the information from CRIBKING to GIN.KING resulted in a fully operational program without the KEYWORD routine. To make a "fixed" copy of your ORIGINAL Cribbage/Gin King disk follow these steps:

Step-by-step

1. Make a copy of your original game disk.
2. Make the following edits to the copy only:

Blk	Byte	From	To
$14E	$8D	22 BE CF 00	AF BE CF 00
	$99	D0 03	EA EA
$2A9	$FD	85 CC	EA EA
$2F5	$1DA	22 BC 8F 00	AF BC 8F 00
	$1E6	D0 03	EA EA
$3CD	$11E	85 CC	EA EA

3. Write the blocks back to the copy.

That's it! The entire requestor routine has been disabled.

Softkey for...

Designer Puzzles

MECC

Requirements:
768K Apple IIgs
3.5" disk copier
3.5" disk editor

Designer Puzzles (DP) is a new educational program from MECC. Like the other education programs from this company, it uses a bad block check for the copy protection. The same string of 90 05 20 is found on this program. Checking the surrounding code matched up with earlier programs ie: JSR, BCC +5, JSR, BCS +?? Making a simple patch defeats the CP altogether.

Step-by-step

1. Make a copy of the disk (ignore errors).
2. Make the following edits to a copy only:

Blk	Byte	From	To
$2B3	$79	20 EC 0A	AD EC 0A
	$7C	90 05	80 03
	$7E	20 D1 09	AD D1 09
	$81	B0 07	EA 18

3. Write the block back to the copy.

That's it, now you can enjoy the newly deprotected program.

Softkey for...

Stickybear GS Talking series
Talking Alphabet
Talking Opposites
Talking Shapes

Optimum Resources

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" disk editor
Stickybear original

Optimum Resources has come out with three GS versions of their popular Stickybear educational software series. These GS version are copy protected by the use of checking for a bad block. There is a simple GET_DEVICE (22 A8 00 E1 20) followed by a simple READ_BLOCK (22 A8 00 E1 22) and can be patched to skip the read routines by placing a CLC and RTS (18 60) for the first two bytes of the GET_DEVICE call. However I did find the call to the block read on each disk and found that changing the JSR's to LDA's will deprotect the programs. Here are the patches by programs:

Step-by-step

1. Copy the disks.
2. Edit the copy.

Talking Alphabet on disk one:

Blk	Byte	From	To
$F4	$14F	20 B6 18	AD B6 18

Talking Opposites

Blk	Byte	From	To
$EC	$15D	20 07 3A	AD 07 3A

Talking Shapes

Blk	Byte	From	To
$EC	$183	20 7A 2E	AD 7A 2E

3. Write the blocks back to the disk.

Softkey for...

TrianGO

Logical Design Works/California Dreams

Requirements:
512K Apple IIgs
3.5" disk copier
3.5" disk editor

Although this krack appeared a while ago in an earlier issue, it's on the MOST WANTED list so here's the patches:

Step-by-step

1. Copy the original disk.

2. Make the following edits to a copy only:

Blk	Byte	From	To
$14	$1D9	22 A5 22 00	AF A5 22 00
$203	$FE	22 AF A5 00	AF AF A5 00
	$106	D0 06	80 06

3. Write the block back to the copy.

Softkey for...

Task Force

Britannica

Requirements:
512K Apple IIgs
3.5" disk copier

Task Force (TF) is an awe-some new arcade game for the GS programmed by Scott L. Patterson (who wrote the Sword of Sodan demo). The game is loosely based on the NARC arcade game with similar play. Actually, the year is 1997 and your job is eliminate the gang type terrorist activities in five different cities around the US. TF has the best graphics and animation of any GS game I have ever seen. This game is a MUST HAVE in your game library.

Anyway, TF is copy protected in the same way as Gnarly Golf, which is also by Britannica. Block $63F (the last block of the disk) has altered data headers of AA CC D5 and is unreadable by normal ProDOS. The trick is, the program modifies the header table and tries to correctly read in the block. Then another routine does a checksum on the data read in to see if it's being run from an original disk. So, when you make a copy with COPY][+ (or other program that will continue on read errors) the data is lost and a normal formatted block $63F is written to the copy. TF will try to read this block in the altered format and will come back with a read error. Even if you were to disable the block read routine, the game still runs a checksum on where the data is supposed to be. As the checksum routine will fail on a copy the program jumps back to the intro/title display after you select 1 or 2 players. To further protect the program from piracy, almost all of the program is encrypted on the disk, that is, except for the STARTUP .SYSTEM file. So I used what I call the front door method, starting with the system file I traced and followed the program as it goes. Eventually I was able to cheat the copy protection and produce a deprotected backup copy that runs just fine. Below I will show part of the CP and in the end I'll show you how to make your own deprotected copy.

As all system files start at $2000 in memory, I bloaded the file and examined the code. The first thing the program does is to move itself down to $800 and jumps to the start the actual program at $837. The code clears the super hi-res screen and starts loading in files and decrypting those file that need it. Finally I came to a JMP 02427F, this is where we will start looking. The following code is in bank 2:

```
427F:78       SEI          No interrupts
4280:18       CLC          Set up for mode
                           switch
4281:FB       XCE          Switch to GS mode
4282:4B       PHK          Push PROGRAM
                           bank
4283:AB       PLB           and make it the DATA
                           bank
4284:C2 30    REP  #$30    16 bit wide
                           Accumulator & Index
                           registers
4286:F4 00 00 PEA  0000    Push a 16 bit zero on
                           the stack
4289:2B       PLD          Direct(zero) page
                           0000 for //E emulation
428A:A9 FF 01 LDA  #$01FF  Load 01FF
428D:1B       TCS          Set STACK location to
                           1FF for //E emulation
428E:AF CA 00 E1 LDA E100CA
4292:85 AC    STA  AC
4294:64 AA    STZ  AA
4296:9C 47 D4 STZ  D447
4299:20 C5 AC JSR  ACC5    Store original data-
                           header values
429C:20 91 C1 JSR  C191    Modify data-header for
                           CP read
429F:22 43 0A 00 JSL 000A43 Read the block
42A3:20 AD C2 JSR  C2AD    Restore the original
                           data-header values
42A9:4C 88 41 JMP  4188    Jump to the start (init)
                           of the game
```

Well lets play the CPU game and follow the JSR's & JSL's:

JSR ACC5 - (this code is in bank 02)

```
ACC5:AF 58 0F E1 LDA  E10F58 Load two bytes from
                             data-header table
ACC9:AD 0A A6    STA  A60A   Store in a temp
                             location
ACCC:AF 5A 0F E1 LDA  E10F5A Load two more bytes
ACD0:8D 0C A6    STA  A60C   Store in another temp
                             location
ACD3:60          RTS         Return to caller
```

- JSR C191 -

```
C191:A9 D5 CC LDA  #$CCD5 Load two "corrupt"
                          values
C194:8F 58 0F E1 STA  E10F58 Store in data-header
                             table
C198:E2 20    SEP  #$20   Switch to 8 bit wide
                          accumulator
C19A:A9 AA    LDA  #$AA   Load the last "corrupt"
                          value
C19C:8F 5A 0F E1 STA  E10F5A Store in data-header
                             table
C1A0:C2 30    REP  #$20   Back to 16 bit wide
                          accumulator
C1A2:60       RTS         Return to caller
```

JSL 000A43 - (this code is in bank 00)

```
A43:20 E5 0C  JSR  0CE5   Stores zero page
                          memory in bank 01
A46:48        PHA         Push (save) the
                          accumulator
A47:08        PHP         Push (save) the
                          processor status
A48:E2 20     SEP  #$20   8 bit wide accumulator
A4A:AF 35 C0 E1 LDA E1C035 Load shadow register
A4E:29 BF     AND  #$BF   No shadowing
A50:8F 35 C0 E1 STA E1C035 Store in shadow register
A54:C2 20     REP  #$20   16 bit wide accumulator
A56:28        PLP         Pull processor state
A57:68        PLA         Pull accumulator
A58:8B        PHB         Save current DATA
                          bank
A59:4B        PHK         Push PROGRAM bank
A5A:AB        PLB         and make it the DATA
                          bank
A5B:64 AE     STZ  AE
A5D:20 AD 0C  JSR  0CAD   Store more memory in
                          bank E1
A60:38        SEC         Set up for mode change
A61:FB        XCE         Switch to //E emulation
A62:20 00 BF  JSR  BF00   ProDOS 8 MLI entry
                          point
A65:80        80          Command number for
                          read block
A66:B4 0A     0AB4        Read block parms
                          location
A68:08        PHP         Push the processor
                          status
A69:18        CLC         Set up for mode change
A6A:FB        XCE         Switch back to GS mode
A6B:28        PLP         Pull the processor status
A6C:C2 30     REP  #$30   16 bit wide Accumulator
                          & index registers
A6E:A9 00 00  LDA  #$0000
A71:20 F9 0C  JSR  0CF9   If read error ask for
                          original
A74:B0 EA     BCS  A60    Carry set means a copy,
                          try again
A76:20 BF 0C  JSR  0CBF   Move block data from
                          00/1800 to 02/9800
A79:AB        PLB         Restore the DATA bank
A7A:48        PHA         Push (save)
                          accumulator
A7B:08        PHP         Push (save) the
                          processor status
A7C:E2 20     SEP  #$20   8 bit wide accumulator
A7E:AF 35 C0 E1 LDA E1C035 Load shadow register
A82:09 5F     ORA  #$5F   Modify
A84:8F 35 C0 E1 STA E1C035 Store in shadow register
A88:C2 20     REP  #$20   16 bit accumulator
A8A:28        PLP         Restore processor
                          status
A8B:68        PLA         Restore the accumulator
A8C:20 9D 0C  JSR  0C9D   Move data back from
                          bank E1
A8F:20 D1 0C  JSR  0CD1   Move zero page back
                          from bank 01
A92:6B        RTL         Return to caller
AB4:03 50 00 18 3F 06     ProDOS Block read
                          parameters, 3
                          parameters: Slot x
                          16=$50, buffer=$1800,
                          block=$63F
```

JSR C2AD - (this code is in bank 02)

```
C2AD:AD 0A A6 LDA  A60A   Load original data-
                          header values
C2B0:8F 58 0F E1 STA  E10F58 Store in data-header
                             table
C2B4:E2 20    SEP  #$20   Switch to 8 bit wide
                          accumulator
C2B6:AD 0C A6 LDA  A60C   Load the last date-
                          header value
C2B9:8F 5A 0F E1 STA  E10F5A Store in data-header
                             table
C2BD:C2 20    REP  #$20   Switch to 16 bit wide
                          accumulator
C2BF:60       RTS         Return to caller
```

That's most of the code that loads the altered block information into memory. Now let's take a look at the code that does the checksum on the data (in memory bank 02):

```
D3FD:A2 FE 01 LDX  #$01FE Go through a whole
                          blocks worth of data
D400:A0 00 00 LDY  #$0000
D403:8A       TXA
D404:18       CLC
D405:C8       INY
D406:98       TYA
D407:2A       ROL
D408:A8       TAY         Self generating
                          checksum
D409:08       PHP         Push/store processor
                          status
D40A:DD 00 98 CMP  9800,X Compare to block $63F
                          data
D40D:D0 1E    BNE  D42D   Not equal, then exit
                          through other routine
D40F:28       PLP         Pull/restore processor
                          status
D410:CA       DEX
D411:CA       DEX         Decrement the X index
                          register
D412:10 F1    BPL  D405   Do we loop through all
                          $200 bytes?
D414:18       CLC         Carry clear means it
                          passed
D415:60       RTS         Return to sender
D416:29 FF 00 AND  #$00FF
D419:09 00 03 ORA  #$0300
D41C:E0 00 00 CPX  #$0000
D41F:F0 03    BEQ  D424
D421:09 00 80 ORA  #$8000
D424:20 87 C9 JSR  C987
D427:A9 00 04 LDA  #$0400 From D416-D42C is
                          code for some other
D42A:20 87 C9 JSR  C987   routine, not sure what
                          it's for.
D42D:28       PLP         Checksum exits through
                          here on failure
D42E:38       SEC         Carry set (checksum
                          routine only) means
                          COPY
D42F:60       RTS         Return to sender
```

As you can see the checksum routine does some kind of self creating checksum data for block $63F and checks to see if the data is correct or not. If not it exits through the ending of a non related routine that ends with the carry set. Otherwise, the program returns with carry clear which is used as a flag to see if the CP has passed or not. Searching for calls to this routine I discovered it is called nine times during the run of the program. Again this code is encrypted on the disk so there is no easy patch.

With all these routines in mind I thought I would read block $63F in and capture the data. Then add this data to the system file and move it into place as the system file runs and jump over the block read code. This turned out to easier then trying to figure out the encrypting routine and making edits to the disk.

To make a deprotected copy of Task Force you must enter a little program to read the block into memory, then save the data to disk. Type in the hex dump and save it to disk.

```
1000:18 FB C2 20 AF 58 0F E1 $E733
1008:8D 00 03 AF 5A 0F E1 8D $9280
1010:02 03 A9 D5 CC 8F 58 0F $3512
1018:E1 E2 20 A9 AA 8F 5A 0F $8282
1020:E1 38 FB 20 00 BF 80 40 $C031
1028:10 18 FB C2 20 AD 00 03 $039A
1030:8F 58 0F E1 E2 20 AD 02 $CF30
1038:03 8F 5A 0F E1 38 FB 60 $0A0F
1040:03 50 00 18 3F 06 00 00 $6048
1048:00 00 00 00 00 00 00 00 $60C8
```

BSAVE TF.READ.B63F, A$1000, L$50

Next you must load in the system file, add the data and modify the system file to do the things we want it to do.

Step-by-step

1. First thing to do is WRITE PROTECT your original Task Force disks!
2. Make a copy of both of your Task Force game disks.
3. Get into BASIC and:
BLOAD TF.READ.B63F, A$1000
4. Insert the ORIGINAL Task Force disk #1.
PREFIX,S5
CALL-151
1000G
5. Insert temp storage disk.
BSAVE TF.BLOCK.63F, A$1800, L$200
6. Insert the COPY of Task Force disk #1.
BLOAD STARTUP.SYSTEM, A$2000, TSYS
2026:A9 00 0E
2034:4C 43 0A
20C9:5C 88 41 02
2243:A9 FF 01 A2 00
2248:14 A0 00 98 54 02 00 4B
2250:AB 18 4C 37 08
EA<2255.2290Z
2291:18
7. Insert the disk with "TF.BLOCK.63F".
BLOAD TF.BLOCK.63F, A$2C00
8. Reinsert the COPY of Task Force disk #1.
BSAVE STARTUP.SYSTEM, A$2000, TSYS, L$E00
9. Store the ORIGINAL Task Force disks in a safe place.

Now you have a completely deprotected backup copy of Task Force that can uploaded to a hard disk or run from 3.5" disks without the need to insert the original disk 1 for the copy protection routine.

Softkey for...

RoboCOP

Data East

Requirements:
128K Apple IIe
5.25" disk editor
COPYA

Data East has tried to port RoboCOP from the arcades to the 128K (double Hi-res) Apple IIe. They have changed the epilogues to FF FF and use the PROLOK copy protection on the loader. The disk is easy to softkey using COPYA and the SAME edits (different location) I gave for The Sporting News Baseball (Epyx) in COMPUTIST #71. However, you must get the disk in a normal format before you can make any edits to the disk. This is easy to do using COPYA from the old DOS 3.3 master disk or you could use Super IOB with a fast controller copying ALL the tracks.

Step-by-step

1. Type the following:
CALL-151
B942:18
RUN COPYA (or Super IOB)
2. Copy the whole disk both sides.
3. Make the following changes to side one:

Trk	Sct	Byte	From	To
$00	$01	$3E		A9 FC 85 F0
85				F3 85 F6 A9
EE				85 F1 85 F2 85 F5 A9 E7 85
F4				85 F7 18 90 6E

4. Write the sector back to the copy.

Softkey for...

Thief

Data Most

Requirements:
48K Apple II series
Demuffin Plus

How to make Thief into a BRUNable file

After re-reading issue number 69 (every once in awhile I re-read a bunch of COMPUTISTs) I came a cross an article by Paul R. Wilson concerning my update to his softkey of Thief. So... I got out my original article (not published) and followed the steps and produced a perfect working copy. With Paul's cheat I was able to get over 8,000 points so I know there is was no problem with my methods. So then I got out issue 49 and read my article and found a "bug" in the line of the hex that started : 5B48:40 5B EE 43 5B CA F0 EE the F0 should be a D0 (BEQ-Branch EQual to BNE-Branch Not Equal). With the one byte change my update works as stated. Anyway I'll show the easy way to capture the game and show all the steps. First you need a copy of Demuffin plus and a 48K DOS 3.3 slave (ancient tools!, but very powerful!).

Step-by-step

1. First boot your original Thief disk and at the title page press reset. Keep pressing reset until the disk stops (may take a couple of times). Now you should be at the Applesoft prompt "]"
2. You will have to change the C5 48 (CMP $48, compare to value stored at $48 in the zero page) to C9 DE (CMP #$DE, compare accumulator to the value of $DE the hex number) in the modified DOS 3.2 that Data Most is using. Then move the RWTS from Thief's DOS down to safe memory so we can reboot and use it later.

CALL-151
B902:C9 DE
B971:C9 DE
B99B:C9 DE
B90C:C9 AE
B947:C9 AE
B967:C9 AE
8800<B800.BFFFM

3. Put the slave disk (with no HELLO, but Demuffin plus on it) in the drive.

C600G

4. This will cause the disk in the drive to boot up (assuming you have your controller card in slot 6). Now bload Demuffin Plus and get back into the monitor for some work.

BLOAD DEMUFFIN PLUS *Loads the program*
CALL-151 *Enters the monitor*
B800<8800.8FFFM *Moves the captured RWTS back into place*
803G *Starts up Demuffin Plus*

5. Now follow Demuffin's prompts and transfer all the files from your original Thief disk to the normal slave disk. Use the "wildcard" (=) for the filename.
6. After you have transfered all the files reboot the DOS slave disk to get a normal DOS 3.3 into memory. Now it's time to make the game into a single file:

BLOAD ROBOT, A$2000
BLOAD MZ.OBJ0, A$4000
BLOAD TBGEN.OBJ0, A$4C00
BLOAD MSSL.OBJ0, A$5300
BLOAD MZS.OBJ0, A$5C00
BLOAD DROUTS.OBJ0, A$6000
CALL-151
5B00:2C 50 C0 2C 52 C0 2C 54
5B08:C0 2C 57 C0 2C 10 C0 A2
5B10:FF A9 60 20 A8 FC AD 00
5B18:C0 30 03 CA D0 F3 2C 10
5B20:C0 A2 06 A0 00 B9 00 4C
5B28:99 00 08 C8 D0 F7 EE 27
5B30:5B EE 2A 5B CA D0 EE 20
5B38:00 08 A2 08 A0 00 B9 00
5B40:53 99 00 08 C8 D0 F7 EE
5B48:40 5B EE 43 5B CA D0 EE
5B50:4C 00 40
1FFD:4C 00 5B
BSAVE THIEF, A$1FFD, L$4DE0

That's all there is to it! However I thought you might like to see the neat little routine I used above, explained so you can use it in your programs. This little routine (totally relocatable) will wait for a keypress or "time out" with a selectable delay. The routine looks like this:

```
        BIT  C010   Clear the keyboard strobe
        LDX  #$FF   Loop through 255 times
lp1     LDA  #$60   Wait value, higher number
                    longer delay
        JSR  FCA8   Built in ROM wait routine
        LDA  C000   Load keyboard
        BMI  out    BMI taken for a key pressed
        DEX         Otherwise decrement loop
                    counter
        BNE  lp1    If not out of loops goto LDA
                    #$60
out     BIT  C010   Clear the strobe again
```

Put this code anywhere and when you execute it, it will wait for about 5 seconds then continue or continue right away if you press a key. You can get a longer wait by increasing the value you load in the accumulator. The longest wait would be using LDA #$00.

Softkey for...

Genesis

Datasoft

Requirements:
Non 65C02 Apple
Slave disk with no HELLO

Making Genesis into a single BRUNable file

Way back in issue #53 of COMPUTIST (see that issue for references), I wrote an article on how to capture Genesis where you needed a way into the monitor. Well I got out my original and started looking into it a little. Although you still need a non 65C02 machine (Apple][+ or un-enhanced //E) to perform the softkey the results can be used on any machine. With a little boot code tracing I was able to write a little "auto-loader" for Genesis so you wont need a NMI card or absolute reset styled mod to your machine. If you list the auto-loader, you will see it changes the jumps to the next stage of Genesis' loader. Then it jumps back to the auto-loader which redirects the next jump and executes that stage of the loader. To get a single BRUN-able file of Genesis follow these instructions:

Step-by-step

1. First find a non 65C02 based Apple // series computer. Get into the monitor and move the boot code controller down and add the auto loader:

CALL-151
9600<C600.C6FFM
9700:A9 0D 8D B4 08 A9 97 8D
9708:B5 08 4C 01 08 A9 1A 8D
9710:6D 07 A9 97 8D 6E 07 4C
9718:05 07 A9 FF 8D 85 04 A9
9720:58 8D C8 04 4C 84 04
96F8:4C 00 97 *(Change your 4C 01 08 to 4C 00 97 wherever it is)*

2. Insert your original Genesis disk and execute the modified loader with a 9600G, the whole program will load in and dump you back to the monitor.

9600G

3. Remove the original Genesis disk and boot your slave disk. Get back into the monitor then move some memory down to an unused area and write a move routine to move it back when you run the game.

C600G
CALL-151
900:4C 55 60
8089:EA EA EA EA EA EA EA EA *(9 EA's)*
92D:85 AD A9 FF *Optional: To have the paddles as the default controllers*
5600<8300.8CFFM
5560:A0 00 B9 00 56 99 00 83
5568:C8 D0 F7 EE 64 55 EE 67
5570:55 AE 67 55 E0 8D D0 E8
5578:A9 09 A0 80 8D 01 09 8C
5580:02 09 4C C1 80
BSAVE GENESIS, A$900, L$7B00

Simple and quick to do, now store the file on any DOS 3.3 disk and BRUN it when you want to play the game.

Softkey for...

Hard Hat Mack

Electronic Arts

Requirements:
48K Apple
DOS 3.3 Slave disk

Making Hard Hat Mack into a single BRUNable file

Although Hard Hat Mack (HHM) was softkeyed back in Computist #5 and revisited in issue #38 (this will also fix the bugs in that article), I see it's on the most wanted list so here is yet another softkey for the old but still playable game.

The game is actually a total load game the checks for the disk each time you start a game and also during the run of the demo. With total load (the whole program is memory at all times) programs it's better to capture the game as a BRUN-able file rather than try to convert the whole disk. The best way to capture these total load programs is by boot code tracing.

Boot code tracing was the most powerful weapon krackers had when the early games came out. The only problem was you needed to a bit about machine language to really use it. The theory is easy to understand, just load in each part of the program while maintaining control until you have the whole program in memory. Then save the need memory to a normal disk and build a program that puts everything back where it's supposed to go then run it.

Now that programs are larger and take up several disks, the only need for boot code tracing is to capture the RWTS (read/write/track/sector routines) so you can swap it into your DOS and read from the protected disk, then swap it out and write to a normal formatted disk. Actually HHM can be deprotected in a full disk format but you will see later why people go through the effort to make single files out of games.

To start, we must move the boot code ROM from the controller card down into RAM where we can change it.

9600<C600.C6FFM
96F8:4C 59 FF *GS use 96FA:4C 59 FF*

The ROM code loads track zero, sector zero into $800-8FF in your Apple's memory. At 96F8 on the old disk][controller cards there is a JuMP 801 that starts the first stage DOS boot which will load in DOS. Since we don't want the code to start running we change it to jump to the monitor entry point of FF59. However the ROM code uses the value at $800 to see how many sectors to load in. On a normal DOS 3.3 disk this value is one, yet for HHM we have a value of $10 or all sixteen sectors. Now with that little bit of background out of the way lets load in the first stage loader and see what we can find out. To do this we will need to execute our modified code at $9600.

9600G

This will load in the sixteen sectors from track zero and then instead of jumping to the start of the first stage loader we jump to the monitor entry point so we can maintain control. Listing from $801 we see the first thing the program does is to jump to $80E. So let's list from $80E and see what's going on. You can tell from $80E to $827 the program is moving memory from $800 through $17FF up into $B000 through $BFFF and at $828 is jumping to $B047. So let's get the code to where it's going to run by moving it ourselves.

B000<800.17FFM

Now scanning through $B000 at $B041 we see JMP (B00A) followed by JMP (B00C), which means, jump to the memory address stored at B00A which is 0000, however at B00C the memory reads 0800. So eventually the program jumps to 800, after experimenting we find this is a jump to the actual start of the game. So if you change B00C to point to FF59, the whole game will load in and jump to monitor entry instead of starting to run. To do this make the following changes in memory:

96F8:4C 47 B0 *Redirect the jump to the modified first stage loader*
B00C:59 FF *Jump to the monitor instead of the game start*
9600G *Load the entire game into memory*

From here if you knew which memory was needed you could save it out to disk, then later put it back into memory and run it. After much code tracing and plain experimentation I was able to figure out the memory which needs to be saved. Also I found the check for the original disk is first loaded into $3500-3700 and moved down onto the text page to run. To make a long story shorter, I found if you kill the calls to the disk check you save quite a bit of memory. You will need a 48K DOS 3.3 slave disk with NO HELLO program to reboot and save the game to disk.

Step-by-step

1. Move the Boot ROM down to RAM so we can modify it to maintain control of the program load. Then modify the jump to the first stage loader so the code will jump to the monitor and return control the the user. lastly, execute the modified ROM code to load in the first stage of the Hard Hat Mack loader.

9600<C600.C6FFM
96F8:4C 59 FF *for GS change 96FA:*
9600G *Load first stage loader*

2. Move the first stage loader to it's proper running memory and redirect the jump to the beginning to the monitor entry point

B000<800.17FFM
B00C:59 FF

3. Now redirect the modified ROM code to jump to the modified first stage loader, then execute it to load the entire Hard Hat Mack game into memory

96F8:4C 47 B0
9600G

4. Make the following changes in memory to remove the check for the disk.

864:EA EA EA EA EA EA
1114:60
1212:60
12D8:60
5F32:60
70A8:60

5. Move the $800 page up to a save location for the reboot. Then insert the DOS 3.3 slave disk with no HELLO program on it and reboot

3400<800.8FFM
(Insert DOS 3.3 slave.)
C600G

6. Restore the $800 page and move some high memory dow to save disk space and compact the new binary file version of HHM.

800<3400.34FFM
2A00<7F00.94FFM
2000<7D00.7EFFM

7. Enter a short routine to move the compacted memory back to where it's supposed to be.

2310:A0 00 B9 00 2A 99 00 7F
2318:C8 D0 F7 EE 14 23 EE 17
2320:23 AE 17 23 E0 95 D0 E8
2328:A0 00 B9 00 20 99 00 7D
2330:B9 00 21 99 00 7E C8 D0
2338:F1 60

8. Fix the start of the game to overwrite unneeded code and replace it with a call to our unpack routine and the regular HHM start up calls.

800:20 10 23 20 04 22 4C 2D
808:08 00 00 00

9. Save the file to the disk.

BSAVE HARD HAT MACK, A$800, L$7600

There now you should have a completely deprotected binary file version of Hard Hat Mack that takes up 120 DOS 3.3 sectors and can be BRUN from any version of DOS including fast DOS'es.

Vince Andrews WA

The CWD (Change Working Directory) and ONLINE Command

Here are two new external BASIC.SYSTEM commands called CWD and ONLINE. They are used to display or change the current pathname or available devices. These commands are very useful if you get frustrated typing in long prefixes, especially if you only want to go back to the previous directory, or finding out what is the volume name of a device.

The CWD/ONLINE program can be executed by using the BRUN command. The first part of the program is responsible for installing the image of the command handler code that begins at $7100. It first finds a safe place above HIMEM to store the image, patches it so that it will execute in this new position, and then it moves the code to its new home. It also links in the command handler by storing its starting address at EXTRNCMD+1, EXTRNCMD+2 ($BE07-BE08). Just in case there's another user command handler installed, it grabs the address previously stored in EXTRNCMD+1 and EXTRNCMD+2 and stores it in the target address of a JMP instruction in the body of the CWD/ONLINE command handler. This jump is executed only if the handler doesn't recognize the command that was passed to it. This means that control will always daisy-chain back to a previously installed external command handler so that it will have a chance to claim the command.

The GETBUFR ($BEF5) subroutine was used to allocate a safe buffer large enough to store the command handler. It is called with the number of pages required in the accumulator (usually 1). The carry flag will be set and a "PROGRAM TOO LARGE" error will be displayed if the GETBUFR routine couldn't allocate any memory. Otherwise, the first memory page in the block freed-up will be returned in the accumulator.

Since the CWD/ONLINE command handler is not relocatable, all references to internal addresses must be altered to reflect the change in the position of the code. The way I have patched my code was to replace only the high-order part of each absolute address in the handler with the new page number that was acquired from GETBUFR.

The code is moved into place by making use of the system monitor block move subroutine, MOVE ($FEC2). This subroutine moves the block of memory beginning at the address stored in $3C/$3D and ending at the address stored in $3E/$3F to the block beginning at the address stored in $42/$43. MOVE must be called with the Y-register set to zero.

The CWD command

The syntax of the CWD command is:
CWD [..] [<] [\] [*Pathname*]

The brackets mean that the specified parameter is optional. The CWD command can be typed Applesoft direct mode or it can be executed within a program using a PRINT CHR$(4);"CWD" statement.

To print the prefix to the screen:
CWD

To back up to the previous directory:
CWD < *or [..] or [<] or [....] or [<<] or [..<] or [<<<<]*

To back up to the previous directory and enter a new directory:
CWD < *Pathname or [..] [Pathname]*

To go back to the volume name (main prefix):
**CWD **

To go back to the volume name (main prefix) and enter a new directory:
**CWD ** *Pathname*

To change the pathname (change drives, diskettes, etc.):
CWD /*Pathname must be proceeded with a (/)*

Once the command has been claimed by verifying that the first bytes of the input buffer at $200 matches our command name, it will go through a routine to ignore any spaces. If a carriage return (CR) was found then it will print the prefix. If a backslash (\) was found then it will automatically change the prefix to the volume name of the device. Once that has been done, it checks to see if the next byte was a (CR) and exits if found. Otherwise, it uses it for setting a new prefix. It keeps doing this until the next byte is a (CR) and then it will Set_Prefix and exit. If two, four, or six periods [..] or one, two, or three less thens [<] or periods and less thens [..<] then the program will go back one or more levels of the prefix (go back to the previous directory.) After it has backed up to the previous directory, it then gets the next character. If it is a (CR) then exit. Otherwise, it uses it to set the next prefix. If there was none of the above found, then it will set the prefix just like the original BASIC.SYSTEM command. The only thing you can't use with this command are the parameters Slot and Drive.

ONLINE

The syntax of the ONLINE command is:
ONLINE [,S#] [,D#]

The brackets mean that the specified parameter is optional. The ONLINE command can be typed in while in Applesoft direct mode or it can be executed within a program using a PRINT CHR$(4);"ONLINE" statement.

To print all the available devices (volume names):
ONLINE

To find what the volume name of a single device is:
ONLINE ,Sn,Dn *replace n with slot & drive #*

Once the "ONLINE" command is detected, the length of the command (minus 1) is stored at XLEN ($BE52), the external command number (0) is stored at XCNUM ($BE53), and the address of the post-parsing subroutine, EXECUTE, is stored at XTRNADDR, XTRNADDR+1 ($BE50-BE51). Finally, the parsing rules are stored in PBITS, PBITS+1 ($BE54-BE55): "Path name optional" and "Slot/Drive is allowed." The "Pathname option" bit must be set because the ONLINE command does not use a pathname. After the parsing rules have been set up, the carry flag is cleared (everything is OK) and an RTS returns control to BASIC.SYSTEM. BASIC.SYSTEM then parses the command line in accordance with the instructions in PBITS, updates FBITS, FBITS+1 ($BE56-BE57) to indicate the results of the parse, and then jumps to EXECUTE (its address was previously stored in XTRNADDR). The ONLINE program examines FBITS to see if a specific slot/drive was specified. If so, then the slot and drive specified are retrieved from VSLOT ($BE61) and VDRIV ($BE62) and they are used to form the unit number which is required by the MLI ON_LINE command. If not, the unit number is set to 0; this indicates to the MLI that all volumes are to be examined. Once the command has been executed, the names of the active volumes will be stored in the buffer beginning at HIMEM. The volume names are then extracted from the buffer and displayed in the following way:

```
Slot 5, Drive 1: /PROGRAMS/
```

Source code

I have included the source files for those programmers who would like to examine the code and, perhaps, create their own personal commands.

CWD

```
                TTL  "Change Working Directory
                      [CWD]"
                LST  On
*
* BASIC.SYSTEM "CWD" COMMAND
* CWD [..] [<] [\] [Path]
* BY Vince Andrews 27 Jan 91
*
* This will install a new
* command to basic. You can
* use this within your very own
* basic programs
*
SBLOCK          EQU  $3C        ;Parameter for block
                                move
EBLOCK          EQU  $3E
FBLOCK          EQU  $42
IN              EQU  $200       ;Command input buffer
IN2             EQU  $280       ;Command output buffer
EXTRNCMD        EQU  $BE06      ;External command JMP
                                ; instruction
ERROUT          EQU  $BE09      ;Error handler
XTRNADDR        EQU  $BE50      ;Start of external cmd
                                ; handler
GETBUFR         EQU  $BEF5      ;Get a free space
MLI             EQU  $BF00      ;Entry point to MLI
CROUT           EQU  $FD8E      ;Print a CR
COUT            EQU  $FDED      ;Std. character output
                                ; subroutine
MOVE            EQU  $FE2C      ;Block move subroutine
                XC              ;65C02
                MX   %11        ;Full 8-bit registers
                ORG  $7000      ;Assemble at location
                                of...
Y               EQU  1
y               EQU  1
N               EQU  0
n               EQU  0
SAVOBJ          KBD  "Save object code? (Y or N)"
                DO   SAVOBJ
                DSK  Cwd
                FIN
* Print our little message to the screen:
PRINT_TTL       LDX  #0         ;Reset pointer
:1              LDA  :2,X       ;Get byte
                BEQ  START      ;If (0) then branch
                JSR  COUT       ;Otherwise, print it
                INX             ;Increase pointer
                BNE  :1         ;If not (0) then branch
:2              ASC  "CWD [..] [<] [\] [Path]",00
* Calculate # of pages that we need to
  reserve:
START           JSR  CROUT      ;Print Carriage Return
                                (CR)
                JSR  CROUT      ;Print CR
                SEC             ;Prepare for subtraction
                LDA  #>END      ;Get end of command
                                (Hi)
                SBC  #>CMDCODE  ;Subtract it from
                                start of
                                ; command (Hi byte)
                INC             ;Add one to the total and
                STA  PAGES      ; store it here
                LDA  PAGES      ;Reserve the pages for
                                the
                JSR  GETBUFR    ; command handler
                BCC  :1         ;Carry clear if OK
                LDA  #14        ;"PROGRAM TOO
                                LARGE" error
                JMP  ERROUT
:1              STA  PGSTART    ;Save starting page #
* Install the new command handler:
                LDA  EXTRNCMD+1 ;Set up link to
                                existing
                STA  NEXTCMD+1  ; external
                                command.
                LDA  EXTRNCMD+2
                STA  NEXTCMD+2
* Install the external command handler by
  storing its address after the JMP at EX-
  TRNCMD.
                LDA  #0         ;Address of Command
                                (Lo)
                STA  EXTRNCMD+1 ; and put it here.
                LDA  PGSTART    ;Get Starting Page #
                                and
                STA  EXTRNCMD+2 ; put it here.
* Rellocate the code:
                STA  REL_1+2    ;Put Starting Page #
                                here
                STA  REL_2+2    ; and here
                STA  REL_3+2    ;  "
                STA  REL_4+2    ;  "
                STA  REL_5+2    ;  "
                STA  REL_6+2    ;  "
                STA  REL_7+2    ;  "
                STA  REL_8+2    ;  "
                STA  REL_9+1    ;  "
                STA  REL_10+1   ;  "
* Set up parameters for block move to final
  location:
                LDA  #<CMDCODE  ;Load start
                                address (Lo)
                                ; of our command and
                STA  SBLOCK; put it here.
                LDA  #>CMDCODE  ;Load start
                                address (Hi)
                                ; of our command and
                STA  SBLOCK+1 ; put it here.
                LDA  #<END      ;Load end address (Lo)
                                ; of our command and
                STA  EBLOCK;Put it here.
                LDA  #>END      ;Load end address (Hi)
                                ; of our command and
                STA  EBLOCK+1 ; put it here.
                LDA  #0         ;Address (Lo) of where
                                we
                                ; moving our code to
                                and
                STA  FBLOCK;Put it here.
                LDA  PGSTART    ;Get starting page #
                                and
                STA  FBLOCK+1 ; put it here.
                LDY  #0         ;Reset Y-Reg.
                JMP  MOVE       ;Move it!
* Storage area:
PAGES           DS   1          ;Length of command
                                handler
PGSTART         DS   1          ;Starting page of
                                command
                                ; handler
                DS   $7100-*    ;(Must always start on
                                ; page boundary)
CMDCODE         EQU  *          ;Beginning of our
                                Command
*
* This is the command checker. It scans the
  input buffer to see if the command has
  been entered.
*
                CLD
                LDY  #0
                LDX  #0
CHKCMD          LDA  IN,X       ;Get command
                                character
                INX
REL_1           CMP  CMDNAME,Y  ;Same as our
                                command?
                BEQ  CHKCMD1    ;Yes, so branch
                CMP  #$E0       ;Is it lower case?
                BCC  NOTFOUND   ;No, so branch
                AND  #$DF       ;Convert to uppercase
REL_2           CMP  CMDNAME,Y  ;OK, now?
                BNE  NOTFOUND   ;No so branch
CHKCMD1         INY
                CPY  #CMDLEN-CMDNAME ;At end?
                BNE  CHKCMD     ;No, so branch
                BEQ  PREP_1     ;Yes, so branch
* This is where we go if the entered com-
  mand does not match our command. The
  jump should be pointing to the next ex-
  ternal command handler.
NOTFOUND        SEC             ;Set carry to indicate
                                ; failure
NEXTCMD         JMP  $BE9E      ;(Fill in when installed)
* Calculate where the first valid input is and
  put our pointer at that location
PREP_1          LDX  #CMDLEN-CMDNAME ;Length
                                of command
:1              LDA  IN,X       ;Get byte between
                                command
                INX             ; and command input
                                buffer
                CMP  #" "       ;Is it a space?
                BEQ  :1         ;If it is then ignore it
                LDY  #0         ;Reset (length pointer)
                DEX             ;Backup command
                                pointer
                LDA  IN,X       ;Get character
                CMP  #"\"       ;Is it a backslash?
                BNE  PREP_2     ;No, so branch
* Set prefix to main prefix (Volume name):
REL_3           JSR  GET_PFX    ;Get prefix
                LDY  #1         ;Set pointer (length) to
                                ; beginning of our prefix
:1              INY             ;Next byte (past 1st (/))
                LDA  IN2,Y      ;Get byte
                CMP  #'/'       ;Is it at the end of the
                                ; main prefix (Volume
                                name)?
                BNE  :1         ;No, so branch
                STY  IN2        ;Update path with
                                ; new length
```

```
REL_4     JSR  SET_PFX  ;Set Prefix
          LDY  #0       ;Reset length
          INX           ;Put pointer at next byte
          LDA  IN,X     ;Get byte
          CMP  #$8D     ;Is it a <RETURN>?
          BNE  PREP_2 ;No, so branch
          BRA  EXIT     ;Yes, so branch
```

* Check keyboard input and do what we want it to do:

```
PREP_2    LDA  IN,X     ;Get character
          CMP  #$8D     ;Is it a <RETURN>?
          BEQ  :1       ;Yes, so branch
          CMP  #"." ;Is it a period (.)?
          BEQ  GET_PATH ;Yes, so branch
          CMP  #"<" ;Is it a (<)?
          BEQ  GET_PATH ;Yes, so branch
          CMP  #"/" ;Beginning of a prefix?
          BNE  PREP_3 ;No, so branch
          INX
          LDA  IN,X     ;Get next character
          CMP  #$8D     ;Is it a <RETURN>?
          BNE  L3       ;No, so branch
          BRA  REL_3    ;Branch to set Prefix to
                        ; main prefix
```

* Routine that determines if we are going to Set the prefix or Print the prefix

```
:1        STY  IN2      ;Store length here
          CPY  #0       ;Does length = 0?
          BNE  SET_PATH ;No, so branch
```

* We are going to Print the prefix to the screen:

```
          JSR  CROUT ;Print a CR
          JSR  CROUT ;Print a CR
REL_5     JSR  GET_PFX
          LDY  #0       ;Reset our pointer to (0)
:1        LDA  IN2+1,Y ;Get character in prefix
          ORA  #$80     ;Set hi bit
          JSR  COUT     ;Print it to the screen
          INY           ;Increase pointer
          CPY  IN2      ;Does it equal to length?
          BNE  :1       ;No, so branch
          JSR  CROUT ;Print a CR
          BRA  EXIT
```

* We are going to Set the prefix and exit:

```
SET_PATH                ;This label is important
REL_6     JSR  SET_PFX
```

* I put the Exit here so that both ends of the program can reach it when doing a branch.

```
EXIT      JSR  CROUT ;Print CR
          JMP  $3D0     ;Re-enter DOS vector
```

* Routine that fixes backslash and move character into the input buffer ($280) for Get/Set prefix.

```
L3        DEX           ;Backup one
L4        LDA  #'/'     ;Load normal slash (/)
PREP_3    CMP  #"\" ;Is it a backslash? (\)
          BEQ  L4       ;Yes, so fix it
          INX
          INY           ;Increase length
          STA  IN2,Y    ;Store it at our path
          BRA  PREP_2
```

* Location that determines what we do next:

```
GET_PATH  CMP  #"<" ;Is it a (<)?
          BEQ  REL_7 ;Yes, so branch
          INX
          LDA  IN,X     ;Get next character
          CMP  #"." ;Is it a period?
          BEQ  REL_7 ;Yes, so branch
          DEX           ;Backup and
          LDA  IN,X     ; get previous character
          BRA  PREP_3 ;Branch to move
                        character
```

* Backup the path (prefix) by one level

```
REL_7     JSR  GET_PFX ;Get prefix
          LDY  IN2      ;Get Command output
                        character
          DEY           ;Shorten length by one
          BEQ  EXIT     ;Branch if zero
:1        DEY           ;Backup behind the (/)
          BEQ  EXIT     ;Branch if zero
          LDA  IN2+1,Y ;Get character from
                        prefix
          CMP  #'/'     ;Is it a slash (/)?
          BNE  :1       ;No, so branch
          STY  IN2      ;Store the new length
                        here
REL_8     JSR  SET_PFX ;Set prefix
          BRA  L4       ;Add slash (/) to the end
                        ; of the pathname
```

* Set_Prefix call:

```
SET_PFX   JSR  MLI
          DFB  $C6      ;SET_PREFIX call
REL_9     DA   CD_PARMS ;Address of
                        parameter table
          BCS  ERROR ;(BNE or BCS)
          RTS
```

* Get_Prefix call:

```
GET_PFX   JSR  MLI
          DFB  $C7      ;GET_PREFIX call
REL_10    DA   CD_PARMS ;Address of
                        parameter table
          BCS  ERROR ;(BNE or BCS)
          RTS
```

* Error handler (Path not Found):

```
ERROR     LDA  #6       ;Error #6
          JMP  ERROUT ;Error handler
```

* Data area:

```
CD_PARMS  DFB  01       ;One parameter
          DA   IN2      ;Buffer address ($280)
CMDNAME   ASC  "CWD" ;External command name
CMDLEN    EQU  *
          LST  ON
END       EQU  *        ;End of file
          LST  OFF
```

CWD hex

```
7000:A2 00 BD 0D 70 F0 1E 20  $4FB9
7008:ED FD E8 D0 F5 C3 D7 C4  $6A23
7010:A0 DB AE AE DD A0 DB BC  $3BD9
7018:DD A0 DB DC DD A0 DB D0  $3307
7020:E1 F4 E8 DD 00 20 8E FD  $A8CA
7028:20 8E FD 38 A9 71 E9 71  $6236
7030:1A 8D 97 70 AD 97 70 20  $A459
7038:F5 BE 90 05 A9 0E 4C 09  $64F3
7040:BE 8D 98 70 AD 07 BE 8D  $B307
7048:22 71 AD 08 BE 8D 23 71  $0FBF
7050:A9 00 8D 07 BE AD 98 70  $B664
7058:8D 08 BE 8D 0B 71 8D 16  $8028
7060:71 8D 3A 71 8D 4A 71 8D  $4804
7068:83 71 8D 9B 71 8D C4 71  $5D92
7070:8D DA 71 8D E2 71 8D EB  $7FE7
7078:71 A9 00 85 3C A9 71 85  $0B17
7080:3D A9 FA 85 3E A9 71 85  $01B5
7088:3F A9 00 85 42 AD 98 70  $F204
7090:85 43 A0 00 4C 2C FE 00  $5C66
7098:00 00 00 00 00 00 00 00  $9C86
70A0:00 00 00 00 00 00 00 00  $5C66
70A8:00 00 00 00 00 00 00 00  $9C86
70B0:00 00 00 00 00 00 00 00  $5C66
70B8:00 00 00 00 00 00 00 00  $9C86
70C0:00 00 00 00 00 00 00 00  $5C66
70C8:00 00 00 00 00 00 00 00  $9C86
70D0:00 00 00 00 00 00 00 00  $5C66
70D8:00 00 00 00 00 00 00 00  $9C86
70E0:00 00 00 00 00 00 00 00  $5C66
70E8:00 00 00 00 00 00 00 00  $9C86
70F0:00 00 00 00 00 00 00 00  $5C66
70F8:00 00 00 00 00 00 00 00  $9C86
7100:D8 A0 00 A2 00 BD 00 02  $8516
7108:E8 D9 F7 71 F0 0B C9 E0  $9F41
7110:90 0E 29 DF D9 F7 71 D0  $7959
7118:07 C8 C0 03 D0 E7 F0 04  $442E
7120:38 4C 9E BE A2 03 BD 00  $D517
7128:02 E8 C9 A0 F0 F8 A0 00  $5A32
7130:CA BD 00 02 C9 DC D0 1F  $6737
7138:20 E6 71 A0 01 C8 B9 80  $7BF9
7140:02 C9 2F D0 F8 8C 80 02  $E083
7148:20 DD 71 A0 00 E8 BD 00  $9F33
7150:02 C9 8D D0 02 80 45 BD  $7B87
7158:00 02 C9 8D F0 16 C9 AE  $4CF4
7160:F0 4E C9 BC F0 4A C9 AF  $E757
7168:D0 3B E8 BD 00 02 C9 8D  $C0E4
7170:D0 30 80 C4 8C 80 02 C0  $1764
7178:00 D0 1E 20 8E FD 20 8E  $B839
7180:FD 20 E6 71 A0 00 B9 81  $7F07
7188:02 09 80 20 ED FD C8 CC  $A49E
7190:80 02 D0 F2 20 8E FD 80  $7214
7198:03 20 DD 71 20 8E FD 4C  $EACC
71A0:D0 03 CA A9 2F C9 DC F0  $8BB6
71A8:FA E8 C8 99 80 02 80 A7  $9048
71B0:C9 BC F0 0E E8 BD 00 02  $31B9
71B8:C9 AE F0 06 CA BD 00 02  $0D9D
71C0:80 E3 20 E6 71 AC 80 02  $B86B
71C8:88 F0 D1 88 F0 CE B9 81  $F1F0
71D0:02 C9 2F D0 F6 8C 80 02  $A162
71D8:20 DD 71 80 C6 20 00 BF  $4A5C
71E0:C6 F4 71 B0 0A 60 20 00  $4263
71E8:BF C7 F4 71 B0 01 60 A9  $87E1
71F0:06 4C 09 BE 01 80 02 C3  $B21F
71F8:D7 C4                    $C54E
```

ONLINE Source

```
          TTL  "Online"
          LST  OFF
*
* BASIC.SYSTEM "ONLINE" COMMAND
* ONLINE [,Sn] [,Dn]
* BY Vince Andrews 27 Jan 91
*
* This will install a new command to basic.
  You can use this from within your own
  BASIC programs.
SBLOCK    EQU  $3C      ;Parameter for block
                        move
EBLOCK    EQU  $3E
FBLOCK    EQU  $42
HIMEM     EQU  $73      ;Use this as ONLINE
                        buffer
IN        EQU  $200     ;Command input buffer
EXTRNCMD  EQU  $BE06    ;External command JMP
                        ; instruction
ERROUT    EQU  $BE09    ;Error handler
XTRNADDR  EQU  $BE50    ;Start of external cmd
                        ; handler
XLEN      EQU  $BE52    ;External cmd name
                        ; length (-1)
XCNUM     EQU  $BE53    ;Command # (0 for
                        external)
PBITS     EQU  $BE54    ;Command parameter
                        bits
FBITS     EQU  $BE56    ;Parameters found in
                        sparse
VSLOT     EQU  $BE61    ;Slot parameter
                        specified
VDRIV     EQU  $BE62    ;Drive parameter
                        specified
GETBUFR   EQU  $BEF5    ;Get a free space
MLI       EQU  $BF00    ;Entry point to MLI
LEVEL     EQU  $BF94    ;Contains most current
                        ; file level
CROUT     EQU  $FD8E    ;Print a CR
COUT      EQU  $FDED    ;Std. character output
                        ; subroutine
MOVE      EQU  $FE2C    ;Block move subroutine
          ORG  $7000    ;Assemble at location
                        of...
Y         EQU  1
y         EQU  1
N         EQU  0
n         EQU  0
SAVOBJ    KBD  "Save object code? (Y or N)"
          DO   SAVOBJ
          DSK  Online
          FIN
```

* Print our little message to screen:

```
PRINT_TTL LDX  #0
:1        LDA  :2,X     ;Print drive #
          BEQ  START
          JSR  COUT
          INX
          BNE  :1
:2        ASC  "Online [,Sn] [,Dn]",00
```

* Calculate # of pages that we need to reserve:

```
START     JSR  CROUT
          JSR  CROUT
          SEC
          LDA  #>END
          SBC  #>CMDCODE
          STA  PAGES
          INC  PAGES
          LDA  PAGES    ;Reserve the pages for
                        the
          JSR  GETBUFR  ; command handler
          BCC  :1       ;Carry clear if OK
          LDA  #14      ;"PROGRAM TOO
                        LARGE" error
          JMP  ERROUT
:1        STA  PGSTART  ;Save starting page #
```

* Install the new command handler:

```
          LDA  EXTRNCMD+1 ;Set up link to
                        existing
          STA  NEXTCMD+1  ; external
                        command.
          LDA  EXTRNCMD+2
          STA  NEXTCMD+2
```

* Install the external command handler by storing its address after the JMP at EXTRNCMD.

```
          LDA  #0
          STA  EXTRNCMD+1
          LDA  PGSTART
          STA  EXTRNCMD+2
```

* Rellocate the code:

```
          STA  CMDCODE3+1 ;Found within
                        page boundary
          INC           ;Next page boundary
          STA  CMDCODE1+2
          STA  CMDCODE2+2
          STA  CMDCODE4+2
          STA  CMDCODE5+2
          STA  CMDCODE6+2
          STA  SAVEUN+2
          STA  CMDCODE7+1
          STA  CMDCODE8+1
          STA  CMDCODE9+1
          STA  PRTMSG+2
          STA  PRTMSG1+2
          STA  PRTMSG2+2
          STA  PRTMSG3+2
          STA  NEXTNAME+2
```

* Set up parameters for block move to final location:

```
          LDA  #<CMDCODE
          STA  SBLOCK
          LDA  #>CMDCODE
          STA  SBLOCK+1
          LDA  #<END
          STA  EBLOCK
          LDA  #>END
          STA  EBLOCK+1
          LDA  #0
          STA  FBLOCK
          LDA  PGSTART
          STA  FBLOCK+1
          LDY  #0
          JMP  MOVE     ;Move it!
PAGES     DS   1        ;Length of command
                        handler
PGSTART   DS   1        ;Starting page of
                        command
                        ; handler
          DS   $7100-*  ;(Must always start on
                        ; page boundary)
CMDCODE   EQU  *
```

* This is the command checker. It scans the input buffer to see if the command has been entered.

```
          CLD
          LDY  #0
          LDX  #0
CHKCMD    LDA  IN,X     ;Get command
                        character
          INX
          CMP  #$A0     ;Is it a blank?
          BEQ  CHKCMD   ;If it is, ignore it
CMDCODE1  CMP  CMDNAME,Y ;Same as our
                        command?
          BEQ  CHKCMD1 ;Yes, so branch
          CMP  #$E0     ;Lower case?
          BCC  NOTFOUND ;No, so branch
          AND  #$DF     ;Convert to uppercase
CMDCODE2  CMP  CMDNAME,Y ;OK, now?
          BNE  NOTFOUND ;No, so branch
CHKCMD1   INY
          CPY  #CMDLEN-CMDNAME ;At end?
          BNE  CHKCMD   ;No, so branch
          BEQ  SETRULES ;Yes, so branch
NOTFOUND  SEC ;Set carry to indicate
                        ; failure
NEXTCMD   JMP  $0000    ;(Fill in when installed)
SETRULES  DEY
          STY  XLEN     ;Store command length -
                        1
          LDA  #<EXECUTE ;Put address of
                        command
                        ; handler
          STA  XTRNADDR ; Into
                        XTRNADDR.
CMDCODE3  LDA  #>EXECUTE
          STA  XTRNADDR+1
          LDA  #0
          STA  XCNUM    ;External cmd number =
                        0
```

* Set up string parsing rules:

```
          LDA  #$10     ;Pathname is optional
          STA  PBITS
          LDA  #$04     ;Slot/Drive allowed
          STA  PBITS+1
          LDA  HIMEM    ;Set ON_LINE buffer
                        (minimum
CMDCODE4  STA  BUFFER; 256 bytes) to free area
          LDA  HIMEM+1  ; beginning at
                        HIMEM.
CMDCODE5  STA  BUFFER+1
          CLC           ;Clear carry to indicate
                        ; success
          RTS
```

* BASIC.SYSTEM comes here after it has successfully parsed the command line.

```
EXECUTE     LDA  #0
CMDCODE6    STA  UNITNUM  ;(Assume all
                          volumes)
            LDA  FBITS+1 ;Examine result of parse
            AND  #$04     ;Slot/drive specified?
            BEQ  CLOSE    ;No, so check everything
            LDA  VSLOT    ;Get slot # specified
            ASL
            ASL
            ASL
            ASL           ;Slot * 16
            LDX  VDRIV    ;Get drive # specified
            CPX  #2       ;Drive 2?
            BNE  SAVEUN;No, so branch
            ORA  #$80     ;Set "drive 2" bit
SAVEUN      STA  UNITNUM  ;Store slot/drive as
                          ; unit number

CLOSE       LDA  #0
            STA  LEVEL    ;Set LEVEL to 0
            JSR  MLI
            DFB  $CC      ;CLOSE
CMDCODE7    DA   CPARM    ;Address of parameter
                          table
            BCC  DOCALL
            CLC           ;Exit
            LDA  #$00
            RTS

DOCALL      JSR  MLI
            DFB  $C5      ;ONLINE call
CMDCODE8    DA   OLPARM   ;Address of parm
                          table
            JSR  CROUT
```

* Print the Title:

```
            LDX  #0
PRTMSG      LDA  VOLMSG,X ;Print slot #
            BEQ  :1
            JSR  COUT
            INX
            BNE  PRTMSG

:1          LDY  #0
SCAN        TYA
            PHA
            LDA  (HIMEM),Y ;Get slot/drive +
                          length
            BEQ  SCAN2    ;If $00 then all done
            AND  #$0F     ;Isolate Length bits
            BEQ  NEXTNAME ;If 0, then must
                          be error
            PHA
```

* Print the Slot #

```
            LDX  #0
PRTMSG1     LDA  SLOTMSG,X  ;Print slot #
            BEQ  :1
            JSR  COUT
            INX
            BNE  PRTMSG1

:1          LDA  (HIMEM),Y ;Get slot/drive +
                          length
            AND  #$70     ;Isolate slot bits
            LSR
            LSR
            LSR
            LSR           ;WE have slot #
            ORA  #$B0     ;Convert to ASCII digit
            JSR  COUT
```

* Print the Drive #

```
            LDX  #0
PRTMSG2     LDA  DRIVEMSG,X  ;Print drive #
            BEQ  :1
            JSR  COUT
            INX
            BNE  PRTMSG2

:1          LDX  #$B1     ;Assume drive 1
            LDA  (HIMEM),Y
            BPL  :2       ;Branch if drive 1
            LDX  #$B2     ;Must be drive 2
:2          TXA
            JSR  COUT
            LDA  #":"
            JSR  COUT
            LDA  #$A0
            JSR  COUT
```

* Print the Pathname:

```
            LDA  #"/" ;Load "/"
            JSR  COUT  ; and print it
            PLA
            TAX
:3          INY
            LDA  (HIMEM),Y ;Get next character in
                          name
            ORA  #$80     ;Set high bit
            JSR  COUT  ; and display it
            DEX
            BNE  :3       ;Branch until done
            LDA  #"/" ;Load "/"
            JSR  COUT  ; and print it
            JSR  CROUT
NEXTNAME    LDA  UNITNUM  ;Was only one
                          volume
                          ; specified?
            BNE  SCAN2 ;Yes, so branch
            PLA
            CLC
            ADC  #16      ;Move to next name
            TAY
            CPY  #224     ;At end of table?
            BNE  SCAN     ;No, so branch
            PHA
SCAN2       PLA
CMDCODE9    LDA  UNITNUM  ;Was only one
                          volume
                          ; specified?
            BEQ  EXIT     ;No, so branch
            LDY  #0
            LDA  (HIMEM),Y ;Get slot/drive +
                          length
            BEQ  EXIT     ;If $00 then all done
            AND  #$0F     ;Isolate Length bits
            BNE  EXIT     ;If not 0, then must be
                          OK
            INY
            LDA  (HIMEM),Y ;Get error
            CMP  #$2F     ;Is it an "No disk in
                          drive"?
            BEQ  NO_DISK  ;Yes, so branch
            CMP  #$28     ;Is it an "No disk error"?
            BEQ  NO_DEVICE    ;Yes, so branch
EXIT        JSR  CROUT
            CLC           ;CLC ==> no error
            LDA  #0       ;Error code = 0
            RTS
NO_DISK     LDX  #0
PRTMSG3     LDA  :1,X     ;Print "NO DISK"
            BEQ  EXIT
            JSR  COUT
            INX
            BNE  PRTMSG3  ;Always taken
:1          ASC  87,"Error trying to read drive",00

NO_DEVICE   LDX  #0
PRTMSG4     LDA  :1,X     ;Print "NO DEVICE"
            BEQ  EXIT
            JSR  COUT
            INX
            BNE  PRTMSG4  ;Always taken
:1          ASC  87,"No Device Connected",00
CPARM       DFB  1        ;One parameter
            DFB  0        ;Close all files
OLPARM      DFB  2        ;Two parameters
UNITNUM     DFB  0        ;Unit number
                          (DSSS0000)
BUFFER      DA   $0000    ;Device Buffer
CMDNAME     ASC  "ONLINE" ;External command
                          name
CMDLEN      EQU  *
VOLMSG      ASC  "Volumes found Online",8D
            ASC  "~~~~~~~~~~~~~~~~~~~~", 8D, 00
SLOTMSG     ASC  "Slot ",00
DRIVEMSG    ASC  " Drive ",00
            LST  ON
END         EQU  *
            LST  OFF
```

ONLINE Hexdump

```
7000:A2 00 BD 0D 70 F0 19 20 $42BE
7008:ED FD E8 D0 F5 CF EE EC $0127
7010:E9 EE E5 A0 DB AC D3 EE $25E7
7018:DD A0 DB AC C4 EE DD 00 $2EB1
7020:20 8E FD 20 8E FD 38 A9 $59BF
7028:72 E9 71 8D A4 70 EE A4 $873E
7030:70 AD A4 70 20 F5 BE 90 $EA10
7038:05 A9 0E 4C 09 BE 8D A5 $DF9E
7040:70 AD 07 BE 8D 26 71 AD $5007
7048:08 BE 8D 27 71 A9 00 8D $BB63
7050:07 BE AD A5 70 8D 08 BE $1A2D
7058:8D 32 71 1A 8D 0F 71 8D $60B2
7060:1A 71 8D 49 71 8D 4E 71 $5A42
7068:8D 55 71 8D 6F 71 8D 7A $BE1F
7070:71 8D 86 71 8D 0C 72 8D $B753
7078:8E 71 8D A8 71 8D C2 71 $7C47
7080:8D 30 72 8D FD 71 A9 00 $07DC
7088:85 3C A9 71 85 3D A9 BC $AB58
7090:85 3E A9 72 85 3F A9 00 $C48B
7098:85 42 AD A5 70 85 43 A0 $C90B
70A0:00 4C 2C FE 00 00 00 00 $71A2
70A8:00 00 00 00 00 00 00 00 $2182
70B0:00 00 00 00 00 00 00 00 $71A2
70B8:00 00 00 00 00 00 00 00 $2182
70C0:00 00 00 00 00 00 00 00 $71A2
70C8:00 00 00 00 00 00 00 00 $2182
70D0:00 00 00 00 00 00 00 00 $71A2
70D8:00 00 00 00 00 00 00 00 $2182
70E0:00 00 00 00 00 00 00 00 $71A2
70E8:00 00 00 00 00 00 00 00 $2182
70F0:00 00 00 00 00 00 00 00 $71A2
70F8:00 00 00 00 00 00 00 00 $2182
7100:D8 A0 00 A2 00 BD 00 02 $A8D2
7108:E8 C9 A0 F0 F8 D9 7D 72 $72B6
7110:F0 0B C9 E0 90 0E 29 DF $5491
7118:D9 7D 72 D0 07 C8 C0 06 $21DC
7120:D0 E3 F0 04 38 4C 00 00 $681B
7128:88 8C 52 BE A9 51 8D 50 $CEE6
7130:BE A9 71 8D 51 BE A9 00 $AD2C
7138:8D 53 BE A9 10 8D 54 BE $39E3
7140:A9 04 8D 55 BE A5 73 8D $6CC6
7148:7B 72 A5 74 8D 7C 72 18 $E610
7150:60 A9 00 8D 7A 72 AD 57 $B4A3
7158:BE 29 04 F0 13 AD 61 BE $5486
7160:0A 0A 0A 0A AE 62 BE E0 $0E9B
7168:02 D0 02 09 80 8D 7A 72 $BDE2
7170:A9 00 8D 94 BF 20 00 BF $0DFA
7178:CC 77 72 90 04 18 A9 00 $FF0C
7180:60 20 00 BF C5 79 72 20 $6E3E
7188:8E FD A2 00 BD 83 72 F0 $47D8
7190:06 20 ED FD E8 D0 F5 A0 $0DD2
7198:00 98 48 B1 73 F0 6B 29 $132C
71A0:0F F0 58 48 A2 00 BD AE $7F0B
71A8:72 F0 06 20 ED FD E8 D0 $B204
71B0:F5 B1 73 29 70 4A 4A 4A $64BA
71B8:4A 09 B0 20 ED FD A2 00 $9E0F
71C0:BD B4 72 F0 06 20 ED FD $3785
71C8:E8 D0 F5 A2 B1 B1 73 10 $73DD
71D0:02 A2 B2 8A 20 ED FD A9 $B95A
71D8:BA 20 ED FD A9 A0 20 ED $48DE
71E0:FD A9 AF 20 ED FD 68 AA $9FDE
71E8:C8 B1 73 09 80 20 ED FD $1D1C
71F0:CA D0 F5 A9 AF 20 ED FD $1FA7
71F8:20 8E FD AD 7A 72 D0 0A $69CF
7200:68 18 69 10 A8 C0 E0 D0 $2F2E
7208:90 48 68 AD 7A 72 F0 15 $86EB
7210:A0 00 B1 73 F0 0F 29 0F $678C
7218:D0 0B C8 B1 73 C9 2F F0 $B66F
7220:0B C9 28 F0 30 20 8E FD $6858
7228:18 A9 00 60 A2 00 BD 39 $0D37
7230:72 F0 F2 20 ED FD E8 D0 $3A4C
7238:F5 87 C5 F2 F2 EF F2 A0 $AE13
7240:F4 F2 F9 E9 EE E7 A0 F4 $D514
7248:EF A0 F2 E5 E1 E4 A0 E4 $46F8
7250:F2 E9 F6 E5 00 A2 00 BD $D1A1
7258:62 72 F0 C9 20 ED FD E8 $4CF1
7260:D0 F5 87 CE EF A0 C4 E5 $56BA
7268:F6 E9 E3 E5 A0 C3 EF EE $04B6
7270:EE E5 E3 F4 E5 E4 00 01 $66EC
7278:00 02 00 00 00 CF CE CC $0C2C
7280:C9 CE C5 D6 EF EC F5 ED $A01C
7288:E5 F3 A0 E6 EF F5 EE E4 $BC98
7290:A0 CF EE EC E9 EE E5 8D $7780
7298:FE FE FE FE FE FE FE FE $AF58
72A0:FE FE FE FE FE FE FE FE $F700
72A8:FE FE FE FE 8D 00 D3 EC $CCB2
72B0:EF F4 A0 00 A0 C4 F2 E9 $6FD5
72B8:F6 E5 A0 00              $2939
```

The following softkeys and patches are for Apple "Mac" computers.
.....RDEXed

David Todd — MD

I am sending you a couple of Mac deprotect files that I recently downloaded from Compuserve.

Stuart Fischbach

Special thanks to MacBlade; whoever and wherever he is.

Background:

I am the proud owner of a hard disk - the model is not important, but my ability to use software on it is, which is how I came to create this list. Originally, I came across a list which contained many of the entries given here, though not in the same consistent format, nor in alphabetical order. These entries were for the specific changes to be made to copies of program disks so that they could be backed up by "sector" copying (usually associated with Finder copying), instead of requiring the purchase of Copy II Mac, Hard Disk utility, or other backup programs. I kept this list on one of my disks as a useful bit of information that others could use - oh, yes, I did use the patch in order to make backup copies of Microsoft's Word (I was one of the idiots who thought it would be bug-free). In general, though, this list was not of use to me until I bought my hard disk.

As all owners of hard disk's know, NO ONE LIKES TO WAIT FOR THE TIME IT TAKES A FLOPPY DISK PROGRAM TO LOAD. Because of this, I started modifying the original list of patches, and added all the patches which I came across on various BBS's in the public domain.

With that in mind (and done with), I offer the following list of patches for various programs. Not all of the programs will work on a hard disk. In this case, the patch is simply for backup purposes.

In order to use these patches, you will need to have a disk editing program such as MacZap Tools, MacTools, or preferably FEDIT.

Follow the procedures listed using a copy of your program disk, not the original, otherwise you risk the consequences.

ABCBase

Search for: A9F4 42A7
Change to: 4E71
Search for: A9F4 4246
Change to: 4E71

Animation Toolkit1

Search for: 6608 45FA 0020 3012 9041
Change to: 6608 to 600A

Aztec C version 1.00c

In Block 71 (decimal): Change byte $1A9 from: 01 to: 00

Aztec C 1.0

In Block 214 (decimal): Change byte $214 from: 01 to: 00

Championship Boxing

Search for: 66 0A 3D 7C 27 08 FF
Change to: 60 00 00 0A 27 08 FF

Chart

Search for: DD60 DD00 1260 81
Change to: DD60 DD00 124E 71

Checkminder

Search for: 3B5F F6C8 4A6D F6C8 6707
Change to: 3B5F F6C8 4A6D F6C8 6007
Search for: 101F 0A00 0001 670A
Change to: 101F 0A00 0001 600A

Cutthroats

Search for: 6FDA 1006 6706 3B7C
Change to: 6004 1006 6706 3B7C

Alternate for Cutthroats/Hitchhiker's

Block 44, byte $15D. Change from 67 to 60.

Deja Vu

Disk two is copyable with Copy II Mac (4.5 version, sector copy). Disk one needs to be copied in two parts because there is a laser hole burned into track 59! Imagine, the nerve of those slobs! Copy tracks 0 to 58 using sector copy with format. Next copy tracks 60 to 79 using sector copy without format. Now bring up FEDIT and make the patches.

Search for: 56C0 671E 7200
Change to: 51C0 601E 7200
Search for: 56C0 6722 223C
Change to: 51C0 6022 223C
Search for: 6708 41F8 00EC
Change to: 6008 41F8 00EC
Search for: 6748 486E FDEA
Change to: 6048 486E FDEA
Search for: 56C0 6720 322D
Change to: 51C0 6020 322D
Search for: 57C0 670C 41FA
Change to: 50C0 600C 41FA

Search for: 57C0 6716 41FA
Change to: 50C0 6016 41FA

Desk Toppers

Search for: 0C40 FFAF
Change to: 0C40 0000

Dollars & Sense

Blk	Byte	From	To
425	146	6704	4E71
	FC	6700	6600
468	1E0	487A	6606

alternate

Blk	Byte	From	To
184	$17C	56C0	4E71

Electric Checkbook

Search for: 3E1F 206E FCF0
Change to: 4E71 206E FCF0

Excel

Search for: 37CF 0267 DD79
Change to: 00CF 0267 DD79

alternate fix

Search for: 608E BEDD
Change to: B103 6ADD

Fact Finder 1.0

Search for: 670A 206E FEF6 2D50
Change to: 600A 206E FEF6 2D50

Factfinder

Search: 66 04 A0 02 60 02 A0 03 3D 40 00 12.
Change: 4E 71 4E 71.

(Yeah, I know. Don't ask me if this is the same or different than above!)

Farenheit 451

Search for: 0C42 005A 6F00 0014
Change to: 0C42 005A 4E71 4E71

Feathers & Space

Search for: 6706 38BC FFC2 600C
Change to: 6006 38BC FFC2 600C
(Should find it in 6 places.)

File

Search for: 1260 82BE DD
Change to: 124E 71BE DD

FileMaker

Search for: 6702 2854 200C 57C0
Change to: 6002 2854 200C 57C0

Filevision

Edit the file Filevision.

Blk	Byte	From	To
CE	$1B0	66 04	4E 71

Alternate Filevision

Search for: 6700 FEC6 200B
Change to: 4E71 4E71 200B

(Should be on Block 227, Byte 118.)

Forecast

(same as MacMatch)

Frogger

They named the Frogger program "Finder" and made it invisible, locked, and Bozo bit set, so just use setfile and undo all three and copy it to your disk, and then set the original back to the way it was. Now you have Frogger as a single file!!! One note of interest: After each game it executes the file named "finder" Since that was what it was called. So if you don't want to have to reclick on it everytime you want to play, then make sure the Frogger program is named finder.

FunPak

Search for: 4A5F 670B 48
Change to: 4A5F 6608 48

(each occurrence)

Gato

Search for: 6700 0006 4EAD 04
Change to: 6000 0006 4EAD 04

Grid Wars

Search for: C934 C1B6 9592 F588 5DF2 D666 0D2A A200
Change to: C834 C1B6 9592 F588 5DF2 D666 0D2A A200

Search for: 5CBE 4BDE 27DC 8052 D7DC B346 9FF4 18BE
Change to: 5DBE 4BDE 27DC 8052 D7DC B346 9FF4 18BE

Griffin Terminal

Search for: 6712 4267 3F3C
Change to: 60

Haba-Comm

Search for: 3B5F F3D2 4A6D F3D2 662C
Change to: 3B5F F3D2 4A6D F3D2 602C
Search for: 101F 0A00 0001 670A
Change to: 101F 0A00 0001 600A

or,

Search for: 672C 2E3C 0000
Change to: 60
Search for: 670A 2F0E 4EBA
Change to: 60

HabaCheckMinder

Search for: 670A 2F0E 4EBA
Change to: 60
Search for: 670A 2F0E 4EBA
Change to: 60

Habadex 1.1

Search for: 0167 0A2F 0E
Change to: 0160 0A2F OE

Harrier Strike Mission

Search for: 03FA 87EC EFD4 BB42 870C 10AC 476C DC5A
Change to: 02FA 87EC EFD4 BB42 870C 10AC 476C DC5A
Search for: 9A84 1198 6996 BA8C 1116 7900 C18E 52F0
Change to: 9B84 1198 6996 BA8C 1116 7900 C18E 52F0

Hippo^C Level 1

Search for: 6110 4CDF 7FFE 3E80 4ED0 CCAF
Change to: 4E71 4CDF 7FFE 3E80 4ED0 CCAF

Hitchhiker's Guide to the Galaxy

Search for: 6FDA 1006 6706 3B7C
Change to: 6004 1006 6706 3B7C

Home Accountant

Search for: 46DF 3280 4A40 6B00 000A
Change to : 46DF 3280 4E71 4E71 4E71
Search for: 4A6E FFFE 6D08 1D7C 0001 000C
Change to: 4E71 4E71 4E71
Search for: 0240 0001 6700 FF74 3F3C 0FD2
Change to: 303C

Legacy

Search for: 6702 604C 4EAD 0112
Change to: 6002 604C 4EAD 0112
Search for: 670C 487A 00CC 4EAD
Change to: 600C 487A 00CC 4EAD

Lode Runner

Search for: 6058 4EBA 00EA
Change to: 4E71 4EBA 00EA
Search for: 7E01 602E 4FBC
Change to: 7E1C 602E 4FBC

Macattack

Open up 'Playfield1'. Search for: 67FA 4EAD and change to: 4E71 4EAD.

(You need their system, because it has ten fonts for the game)

MacChkrs/Rvrsi

Blk	Byte	From	To
393	$155	67	60
450	$137	67	60

MacCommand

Search for: 4EEA 0010 31FC 00
Change to: 6000 E8F4

Then move the 5 major files to another disk.

MacDraft 1.0

Search for: 6726 2F0E 4EBA FF90
Change to: 6026 2F0E 4EBA FF90
Search for: 4EAD 00E2 395F FFFE
Change to: 4E71 4E71 395F FFFE

MacDraft 1.1

Search for: 6726 2F0E 4EBA FEC4
Change to: 6026 2F0E 4EBA FEC4
Search for: 4EAD 00EA 395F FFFE
Change to: 4E71 4E71 395F FFFE

Mac Fortran

Search for: 6700 000A 43FA
Change to: 6000 000A 43FA
Search for: 6602 604C
Change to: 4E71 604C

MacGammon/Cribbage

Blk	Byte	From	To
202	$1AB	FA75	FB74
362	$6B	6851	6950

MacJack/Poker II

Search for: 2007 4FBC 00FF
Change to: 6016 4FBC 00FF

MacLabeller

Search for: 0002 2E4E 4D00
Change to: 4002 2E4E 4D00

MacMatch

Search for: 6616 4267 6016
Change to: 6716 4267 6016

MacPascal (version 1.0)

In blocks 521, 553, 587 and 588: Search for 4E56 and change to: 4E75.

In blocks 521 and 588: search for: B26E 0008 67xx xx and change to B26E 0008 674E 71.

MacPoker

Search for: 6722 4247 6010 4FBC
Change to: 6022 4247 6010 4FBC

MacType

Blk	Byte	From	To
25	$FF	67	66

Master Type

Edit the MasterType file.

Blk	Byte	From	To
$0074	$01A5	40	50

Alternate

Blk	Byte	From	To
339	$149	3F00	6098
	$1B0	3F00	6098

Mouse Stampede

Search for: 3F3C 0006 4EBA 03EC 6000 0096
Change to: 4E71 (all of them)

Multiplan version 1.02

There is an invisible file named Neil Konzen. After that is made visible, unlocked, and deprotected, it can be transfered by a normal copy (Finder, or disk copy).

The Neil Konzen file must be protected again so that your copy does not need the original master. Chart, File, and Word operate similarly (the invisible file name is different, however.)

alternate fix:

Search for: 1260 8007 BE
Change to: 124E 7107 BE

OverVue

Search for: 1842 8112 3C00 314E
Change to: 0242 8112 3C00 314E

PageMaker

In sector $DF, byte $B6, change 670E to 6028.

PageMaker 1.0

Search for: 4E56 FFFE 422E 0008 486E
Change to: 205F 301F 1F3C 0001 4ED0

Pensate

Search for: 1B7C 0001 DD00
Change to: 4E71 4E71 4E71

PFS

Search for string: 6704600003B23B7C (found twice). Change the first byte (the 67) to a 60 (a BRA).

PFS version A.03

Search for: 6608 45FA 0028 3012
Change to: 4E71 45FA 0028 3012
Search for: 6604 4267 6008 3F3C
Change to: 4E71 4267 6008 3F3C

PFS File/Report

Search for: 6000 03B2
Change to: 4E71 4E71

Change to: 4E71 4E71 395F FFFE

Real Poker

Search for: 43FA FE80 32BC FFDC
Change to: 4E71 4E71 4E71 4E71

Make the files SALOON and POKER DATA visible. There is a Poker Font in the fonts file you must have in your system file.

Rogue

Search for: 3BEA 7702 C370 54EC 8BA8 208E F5E8 139C
Change to: 3AEB 7702 C370 54EC 8BA8 208E F5E8 139C

Sargon III

Search for: 6608 45FA 0002 3012 9041 6608
Change to: 4E71 45FA 0002 3012 9041 4E71

SkyFox

Search for: 2F10 4217 205F 4E90 (twice)
Change to: 203C 99C4 5501 4E71

Smooth Talker

Search for: 6608 45FA 0038 3012
Change to: 4E71 45FA 0038 3012
Search for: 6614 4267 3B7C
Change to: 4E71 4267 3B7C

Hayden Speller

Search for: 225F 508F 4ED1 4E56 FFC0 xxxx xxxx
Change to: 225F 508F 4ED1 4E71 4EF9 0000 E888

alternate

Search for 4E56 FFC0 41EE. Change to 4E71 6000 0AA8.

The Quest

Search for: 611C 4CDF 7FFE (block 584, byte $18E)
Change to: 4E71 4CDF 7FFE

ThinkTank 128

Search for: 6608 3D6E FDAC 000C 6004
Change to: 4E71
Search for: 674E 486D E184
Change to: 60

Think Tank (dunno which)

Search for: 6608 3D6E FDAC 000C 6004 3D47 000C
Change to: 4E71
Search for: 670C 487A 01A6 4EBA DBA4 6000 0188
Change to: 600C 487A 01A6 4EBA DBA4 6000 0188

ThinkTank 1.1

Search for: 6718 2F0E 4EBA FDDE
Change to: 6018 2F0E 4EBA FDDE

ThinkTank 512

Copy II Mac sector copy the original.
Search for: A002 6002 A402
Change to: 4E71 6002 A402
Search for: 674E 486D E184
Change to: 604E 486D E184

Transylvania

Make all files visble using Set File or some such program. Copy all files except track2.text and track3.text onto another disk.

Transylvania has the annoying habit of ejecting the disk and rebooting when you quit the program. You can cure this by doing the following: Use FEdit to look at Block#25 and change bytes 423 and 424 from A0 17 to ADF4. That does it (A017 is the OS Trap 'Eject' and ADF4 is the OS Trap 'Exit-ToShell').

This will force the program to return to the Finder instead of ejecting and rebooting.

Triple Play 1.0

Search for: 6720 487A 00C0
Change to: 6620 487A 00C0

Trivia Arcade

Search for: 6604 302E
Change to: 6704 302E
Search for: 6756 3F3C
Change to: 6656 3F3C

Trivia Fever

Search for: 0C40 01A6
Change to: 4E75 01A6

Typing Intrigue

Search for: 6702 2854 200C 57C0
Change to: 6002 2854 200C 57C0
or,
Search for: 672A 4267 486E
Change to: 4E71 4E71

Ultima][

Search for: 6608 44FA 003C
Change to: 4E71 44FA 003C
Search for: 6616 4267 6016
Change to: 4E71 4267 6016

Ultima III

Search for: 0A00 0001 1B40 FB19
Change to: 4E71 7000 1B40 FB19

If that doesn't work, try this:

Search for: 4EAD 01E2 101F 0A00 0001
Change to: 4EAD 01E2 101F 0A00 0000

VideoWorks 1.0

Search for: A007 3D40
Change to: 4E71 3D40

This patch NOPs a call to Get_Vol_Info and may have some side effects. So far none have surfaced, if I hear of any I'll keep everyone informed.

Winter Games

Search for: 0BFA 8712 F340 449C 9B98 109E 05F8 238C
Change to: 0AFA

Xyphus

Search for each string seperately and change the 67's to 60's.

Search for: 4EBA 5F6C 101F 6708
Change to: 4EBA 5F6C 101F 6008
Search for: 4EBA 5E6A 101F 6708
Change to: 4EBA 5E6A 101F 6008
Search for: 4EAD 007A 101F 6708
Change to: 4EAD 007A 101F 6008

Mac Hard Disk Ejection Fix

It has come to our attention that many games are obnoxious when run (in patched form) on a hard disk. These games cause a warm reboot and bring down the hard disk.

The solution is to use a disk-editor to scan the games for the Trap A017 (_Eject) and replace it with ADF4 (_ReturnToFinder). For example...

MacAttack

Block	byte	from	to
00C	00BE	A017	ADF4

Frogger

Block	byte	from	to
00C	005C	A017	ADF4

Frogger

Block	byte	from	to
00D	01B4	A017	ADF4

Transylvania

Block	byte	from	to
025	0423	A017	ADF4

The list is endless..... Have Fun and make the world safe for Hard Disks !

Frank Price

WellTris Patch (Mac)

Requirements:
FEdit

Well, the previous posting of my patch does seem to have been lost since I have a bunch of letters in my mailbox asking exactly where it is, so I will just start it in a new topic.

1. File Position: $1A24. Change: 4EAD 016A 4A40 660C 4EAD 021A 4267 4EBA 0162 548F. Change all of those to 4E71. (Possibly $00FE instead of $0162... DT.)
2. File Position: $64B8. Change: 4267 4EAD 00BA 548F. Change all of those to 4E71.
3. File Position: $291E. Change: 4267 4EAD 00BA 548F. All of those to 4E71.

You will notice the second two patches are the same. Those were both to eliminate little mini-checks the Welltris author put in there to make sure no one did what I did. The first one gets rid of the dialog box. This is for the black and white version (as they didn't bother to make the color version work on all color Macs before releasing it). But it should definitely work on the color version unless it uses different copy protection. The file positions may be different though so just search around.

Terry Waskowich

Softkey for...
Dungeon Master Assistant Vol. 2
SSI/TSR

Requirements:
Apple II w/64K
Fast Disk Copier
1 Blank Disk
Sector Editor

This article will show SSI's word protection and help you understand how the scheme works. This article will also show how to take the word protection completely out of Dungeon Master Assistant Volume 2 (DM2). This Program is for the true AD&D player, but the protection scheme used is word protection. The routine SSI uses will not allow you to format a data disk when you try to bypass the routine, that is, until now.

Make a copy of the DM2 disk using a Fast Disk Copier. Since the program will reboot when you hit ctrl-reset, its better to take that out, so here are edits to disable the reboot:

Trk	Sct	Byte	From	To
03	09	$14	A2 00 8E F3	EA EA EA EA
			03 8E F4 03	EA EA EA EA

Write the edits back to the copy of DM2. Boot up the copy and note the way the program runs.

a) show a hi-res page
b) switches to a text credit page,
c) Space Bar to continue.
d) Runs the word-protection.

Boot the disk again, and wait until the drive loads a couple of tracks, then hit ctrl reset, and it will reset into RDOS. RDOS is SSI's heavily modified DOS 3.3. Like War of the Lance, to check if SSI is using a BASIC program in RDOS to launch DM2, type LIST. This is what you should find.

```
10 & RECALL "MAIN"
20 & RECALL "SUBS",43008
30 & RECALL "EDITOR",41216
40 & RECALL "START"
50 & RECALL "TITLE",16384
60 CALL 6144
```

Well, by looking at this listing, it loads some programs in and then does a CALL 6144. When converting this decimal to a HEX value it equals $1800. To get these programs loaded in, yet not execute the you need to delete line 60. Then type RUN.

After loading, the program returns you to RDOS, so let's get into the monitor by typing CALL-151. Start looking at the code starting with $1800 and you'll notice that it pulls up the hi-res page. Keep listing the code, because if you remember after hitting a key, it switches out of the graphic mode into the text mode. That code is found at location $182A.

```
182A:AD 00 C0   LDA  C000
182D:30 0D      BMI  183C (+0D) if key pressed
                     continue to $183C.
102F:C8         INY
1030:D0 2E      BNE  181E (-14)
1832:E8         INX
1033:D0 E9      BNE  181E (-17)
1835:CE 39 18   DEC  1839
1838:A9 FF      LDA  #FF
183A:D0 E2      BNE  181E (-1E)
183C:20 00 A8   JSR  A800   clears the text screen
183F:2C 56 C0   BIT  C056
1842:2C 54 C0   BIT  C054   gets to page #1.
1845:2C 51 C0   BIT  C051   gets into text mode.
1848:20 0F A8   JSR  A80F
184B:20 54 A8   JSR  A854
184E:20 78 18   JSR  1878   pulls up the text credit
                            page
1851:20 74 19   JSR  1974   runs the word-protection
1854:AD 0D 1A   LDA  1A0D
1857:CD 67 18   CMP  1867
185A:D0 08      BNE  1864 (+08)
185C:A9 C4      LDA  #C4
185E:85 50      STA  50
1860:A9 D7      LDA  #C7
1862:85 51      STA  51
1864:4C 99 A8   JMP  A899   At location $1864, it
                            runs the program at
                            location (JMP) $1000.
...
1A13:A9 00      LDA  #00
1A15:8D 9A A8   STA  A89A   if answer is right note
                            STA address.
1A18:A9 10      LDA  #10
1A1A:8D 9B A8   STA  A89B   wrong answer not STA
                            address.
1A1D:60         RTS         returns to address
                            $1854.
```

Now we are At Location $1000

```
1000:A9 00      LDA  #00
1002:8D 2F 03   STA  032F
1005:A9 FF      LDA  #FF
1007:8D 30 03   STA  0330
100A:A5 50      LDA  50
100C:C9 C4      CMP  #C4
100D:D0 05      BNE  1016 (+06) not equal menu
                                selection doesn't work
1010:A5 51      LDA  51
1012:C9 D7      CMP  #C7
1013:F0 05      BEQ  101B (+05) will jump to $101B to
                                load SSI.INIT
1016:EE 95 11   INC  1195
1019:D0 1A      BNE  1035
101B:A9 72      LDA  #72    Code for SSI.INIT to be
                            loaded
101D:85 E6      STA  E6
101F:A9 11      LDA  #11
1021:A2 00      LDX  #00
1023:A0 08      LDY  #08
1025:20 78 A8   JSR  A887
1028:20 81 A8   JSR  A881
102B:E0 00      CPX  #00
102D:F0 06      BEQ  1035 (+06)
102F:20 CE 13   JSR  13CE   asks to insert Master
                            Program Disk
1032:4C 1B 10   JMP  101B   Loads SSI.INIT file
1035:20 7B 11   JSR  117B   Loads
1038:20 0F A8   JSR  A80F
103B:20 00 A8   JSR  A800   clears screen
103E:A9 08      LDA  #08
```

When I was trying this out, I thought I had it, but it didn't allow me to format a data disk until I had modified $1013. After looking at location $1035 I got down to $112B.

```
112B:20 06 A8   JSR  A806
112E:20 0F A8   JSR  A80F
1131:AD 95 11   LDA  1195
1134:D0 FB      BNE  1131 (-05) last check, will
                                branch in circles.
1136:20 08 A8   JSR  A809   continues the program.
1139:A2 07      LDX  #07
```

Once you get familiar with SSI/RDOS Word-protections, you will notice that the program will have code, then the text it will load upon the screen then more code and more text. This will make it easier to look for the correct code instead of the wrong code.

Step-by-step

1. Make a copy of the game disk.
2. Make the following edits to a copy.

Trk	Sct	Byte	From	To
03	09	$53	20 74 19	20 13 1A
02	0D	$0C	C9 C4 D0 06	C9 C4 EA EA
02	0D	$12	C9 D7 F0 05	C9 D7 D0 05
02	0C	$31	AD 95 11 D0 FB	AD 95 11 EA EA

3. Write the edits back to the copy.

When this version of DM2 is booted it will show the hi-res page, text credit screen, and then the main menu will show up. There are no longer any signs of the word protection. Enjoy!

Editorial Notes

Hey, another issue made it out the door. I guess issue #78 had you worried, all that text and only a few softkeys. But I "gotta" print what you send me. Usually in the same order that I receive it. So, in this issue, you'll find some MAC softkeys and some new stuff from Vince Andrews for the IIgs owners. I had to break Brian A. Troha's submission in two and push the rest into issue #80.

Send more letters (on disk, of course)

That reminds me, I've only filled seven (7) pages in issue #80. I could use some more material. So what have you been up to? Have you written any neat programs, read any interesting news that you can pass on, heard any rumors? Send me what you've got. I want to fill those pages with something besides my niece's crayon artwork.

Change of Address (for regular subs)

Some of you are still moving without letting me know in time. About a dozen for issue #78. If you tell me your new address after I've already mailed the next issue, there's no way to stop the post office from destroying your issue. They usually send the torn offcover of your issue with the change of address notice. Also, they charge me 35 cents for the notice. So that means sending you another issue by 1st Class mail will cost:

change of address notice (USPS)	.35
1st Class postage	.98
9 1/2 x 11 envelope	.18
	$1.51

And that's not counting the cost of the second issue and the time to process the order.

So please, let me know when you are going to move or send me $1.50 with your change of address notice and I will mark your record so that the your next issue goes out 1st Class. The Post Office will forward 1st Class mail.

NEWS - MAC Clones on the way?

In the April issue of "MACWORLD" is a story that could be the answer to the die-hard Apple II user. There's a new company that has developed a 3 chip set of ASIC's (Application Specific Integrated Circuit) that clones the hardware on the MAC computer. They have also licensed the "Motif" interface and have written software that emulates the MAC toolbox and system software. Using this technology, a hardware company could sell "a MAC-compatible with a color monitor, hard drive, and a 68030 processor for (about) $600".

This is what the Apple II community has been waiting for. A Mac clone at a price we can afford. Set your new "Mac" next to your II and connect a serial cable between them and you have the best of both parts of the Apple world. Check your club or local library for a copy of the April issue of "MACWORLD" and read all about the 1st real crack in Apple's castle or write and see if you can't get "MACWORLD" to send you a copy of the April issue.

MACWORLD
501 Second Street
San Francisco, CA 94107

And don't worry, I know it's the April issue, but they printed (in very clear type) that, "This is not an April Fool's joke.".

Removable Media hard drives

Talking with Dave at the BBS brought home the fact that a lot of Apple II users don't realize that all Syquest mechanism removable media drives use the same "removable media". Don't pay $89 for your next cartridge by shopping in the Apple II rags. Check out "MacUser" and you'll see the same 44M cartridge for $63 from U.S. Computer Merchants (800-888-8779 Visa, MC).

IRS Fund

Many thanks to everyone who sent some bucks for the IRS fund. We haven't paid it all off yet but we took a hefty bite out of the total and the IRS seems to be happy. If you haven't sent anything yet, how about diverting $5 or $10 for a good cause.

Subscriptions

The drop in paid subscribers seems to have bottomed out at 2,123 with issue #77 and climbed to 2,145 as of issue #78's mailing. I think we've peeled off the chaff and what's left is the hard core of Apple II users. So keep writing and here's what you've already sent me (to print). Enjoy!

unClassifieds

How to place an UnClassified Ad

Send a typed sample copy with appropriate instructions. (If possible, send text on a 5.25" Apple format disk.) Use up to 40 characters per line, we will adjust word wrap.

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 bold letters. Bold letters are wider than normal. If the typed copy does not show bold, circle the words you want bolded and, on the side, write BOLD. 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 a $5 processing fee.

★★★★ **New Rates** (per line) ★★★★

Computist club member .....25¢
All others .....35¢

The minimum order is $5.

- 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 a check or money order (funds drawn on US bank only) for the entire amount to:

COMPUTIST unCLASSIFIEDS
33821 East Orville Road
Eatonville, WA 98328

TRADE YOUR APPLE SOFTWARE

Send your list of programs to trade. I have over 120 originals to trade.

Byron Blystone
PO Box 1313
Snohomish, WA 98290

WANTED

C.I.A. Files
by
Golden Delicious Software
Must have 132 page tutorial

Anthony M. Smith
PO Box 1131
Kulpsville PA 19443

BOOK SALE

Keys to Solving Computer Adventure Games. Maps, hints, clues, & some solutions. Black Cauldron, Kings Quest II, Leather Goddesses, & more. 26 games for Apple & others.

M.K. Simon .....286 pgs
Prentice Hall.....$15.00

Keys to Solving Computer Adventure Games, Book II. Maps, hints, clues, & some solutions. Indiana Jones, Gunslinger, Hollywood Hi-Jinx & more. 13 games for Apple & others.

M.K. Simon .....292 pgs
Prentice Hall.....$15.00

Add $2.50 for postage & Insurance

Software — Books — Magazines

We buy & sell out-of-print & hard-to-find Apple II originals, old and new. Send $1 for catalog.

Frank Polosky
PO Box 9542
Pgh, PA 15223

Ultima V Dungeon Construction Set

Design, study or modify all of Ultima V's 64 dungeon levels and 128 chambers for the Apple version

Seven (7) programs to fully Assist you. Demo dungeon disk, program disk, 2 reference sheets, and manual for $41.00. Send request to:

Rob Fiduccia
2809 Broadview Terrace
Annapolis MD 21401

FOR SALE

Senior PROM

for the IIc

$60

The Senior PROM is a hardware (NMI) device with a complete set of programs in ROM that allow you to examine modify and backup your software.

Brand new — complete with all documentation

Call 1- (313) 846-8340

RDEX Contributors:

Vince	Andrews	18
B. Dudley	Brett	7
Bob	Cherochak	14
	Chuck	15
James A.	Hodge	7
Jeff	Hurlburt	4
	Krakowicz	11
Jack	Moravetz	13
Frank	Price	22
Stuart	Fischbach	20
	Sysop	13
David	Todd	20
Brian A.	Troha	15
	User #19	13
	User #2	13
	User #21	13
John E.	Wanner	15
Terry	Waskowich	22

Most Wanted

65	Airheart	Broderbund
63	Alcon	Taito
74	Algebra Shop	Scholastic
63	Alien Mind	PBI Software
73	American History Explorer Series	Mindscape
75	Anchorman	Virginia Reel
74	Animals of the Past	Focus Media
72	Ankh	Datamost
73	Ant Farm	Sunburst
67	Aquatron	Sierra
69	Axis Assassin	?
63	Bad Street Brawler	Mindscape
73	Bank Street Beginner's Filer	Sunburst
73	Bank Street School Filer	Sunburst
63	Beyond Zork	Infocom
65	Bilestoad	Datamost
69	Blue Powder - Grey Smoke	Grade
74	Birds - Trees & Flowers	Focus Media
63	Border Zone	Infocom
65	Borg	Sirius
67	Bouncing Kamungas	Penguin
66	Boxing	?
65	Bureaucracy	Infocom
67	C'est La Vie	Adventure International
69	Caverns of Callisto	Origin
69	Checker	Odesta
69	Chess 7.0	Odesta
69	Chuck Yeager's Adv Flt Trainer	Electronic Arts
75	Clue Master Detective	Leisure Genius
68	Comics	Accolade
63	Cosmic Relief	Datasoft
65	Crime & Punishment	Imagic
69	Crossword Magic v4.0	?
69	Cybernation	Nexa Corp.
74	Decimal Dungeon	Unicorn
74	Decisions Decisions: Colonization v1.0	Tom Snyder Productions
69	Delta Squadron	Nexa Corp.
67	Desecration	Mind Games
66	Disk Optimizer System	Nibble Notch
65	Dondra	Spectrum Holobyte
69	Dragon Eye	Epyx
69	Dueling Digits	Broderbund
68	D&D-Master Assistant vol2	SSI
62	DROL	Broderbund
67	Epoch	Sirius
74	Exploring Tables & Graphs Level 2 (SU)	Weekly Reader
67	Evolution	Sydney
67	Falcons	Piccadilly
68	Factastics Trivia	Daystar
75	Final Frontier	Softsmith
73	Fisher's Cove	Tom Snyder Productions
69	Flt Wars	Sirius
74	Fraction Action	Unicorn
69	Gemstone Healer	SSI
73	Geometric Supposer (the)	Sunburst
66	GEOS	Berkley Softworks
72	Galactic Gladiators	SSI
63	Gladiator	Taito
73	Goodell Diamond Caper	Tom Snyder Productions
67	Gorgon	Sirius
66	GradeBuster 1 2 3	Grade Buster
61	Gutenberg Sr.	Micromation LTD.
65	Halls of Montezuma	Electronic Arts
67	High Orbit	Softsmith
67	Horizon V	Softsmith
75	Hunt for Red October GS	Datasoft
69	Impossible Mission	Epyx
62	Indoor Sports	Mindscape
68	Infocomics	Infocom
66	Jane	?
63	Joker Poker	Mindscape
72	Kabul Spy	Sirius
71	Keyboarding Klass	Mastery Developement
75	King's Bounty	New World Computing/Broderbund
68	Kingdom of Facts	Santa Barbara/Thunder Mountain
75	Kobayashi Alternative (The)	Simon & Schuster
72	Lane Mastodon	Infocom
67	Lancaster	SVS
72	Laser Force (IIgs)	Britannica
75	L.A. Land Monopoly	Softsmith
66	Legacy of the Ancients	Electronic Arts
65	Lost Tomb	Datasoft
74	Mammals - Reptiles & Amphibians	Focus Media
65	Manhunter New York IIgs	Sierra On Line
65	Mavis Beacon Teaches Typing (gs)	Software Toolworks
73	McGraw-Hill Problem-Solving Lvl 5&6	Tom Snyder
74	Micro-Typewriter v3.1/4.0	S.E. Warner
67	Microwave	Cavalier
66	Might and Magic II	Activision
73	Mind Castle I	MCE Inc.
69	Minotaur	Sirius
63	Modem MGR	MGR Software
68	Mr. Pixel's Cartoon Kit	Mindscape/Thunder Mountain
73	Mystery of Hotel Victoria	Tom Snyder Productions
63	National Inspirer	Tom Snyder Productions
75	Neptune	Softsmith
66	Observatory (The)	Mindscape/Lightspeed Software
74	Ocean Life	Focus Media
66	Odin	Odessta
63	Operation Wolf	Taito
68	Pensate	Datasoft/Softdisk
69	Phantasie II	SSI
67	Phantoms 5	Sirius
67	Pig Pen	Datamost
74	Plants & Animals of the Desert	Focus Media
75	Prince of Persia (5.25")	Broderbund
67	Project: Space Station	Avantage
75	Promethean Prophecy (The)	Simon & Schuster
67	Pulsar II	Sirius
68	Pure Stat Basketball	?
62	Quadratic Equations II	Olympus Educational Software
63	Questron II	Electronic Arts
68	Rails West	SSI
67	Rear Guard	Adventure International
63	Renegade	Taito
67	Rescue Raiders	Sir Tech
67	Rings of Saturn - Level 10	?
63	Rocket Ranger (IIgs)	Cinemaware
69	Roundabout	Datamost
75	Russki Duck	Softsmith
63	S.D.I. (IIgs)	Cinemaware
62	Sea Stalker	Broderbund
67	Serpentine	Broderbund
74	Seven Cities of Gold	Electronic Arts
68	Skeletal System	Brainbank
63	Sky Shark	Taito
63	Sound Song & Vision	Advanced Software
67	Space Ark	Datamost
62	Spare Change	Broderbund
67	Spectre	Datamost
62	Speedy Spides	Readers Digest
67	Star Cruiser	Sirius
67	Star Maze	Sir Tech
63	StickyBear Math: Add & Subtract	Optimum Resources
68	Stickybear GS Versions 3.5	Xerox
67	Succession	Piccadilly
65	Superstar Ice Hockey	Mindscape
61	Superstar Indoor Sports	Mindscape
74	Surveys Unlimited	Mindscape
68	Talking Text Writer GS	Scholastic
68	Tangled Tales	Origin Systems
69	Tetris (IIe)	Spectrum Holobyte
72	Theatre Europe	PBI
74	The Other Side v2.0	Tom Snyder Productions
65	Thunder Chopper	?
63	Ticket to Washington D.C.	Blue Lion Software
74	Time Explorers	Gameco
74	Time Liner v1.1	Tom Snyder Productions
63	Tomahawk	Electronic Arts
68	Tomahawk (IIgs)	Datasoft
69	Track Attack	Broderbund
68	Triad	Thunder Mountain
72	Triango (IIgs)	California Dreams
68	Trinity	Infocom
73	Unicorn 5.25" software	Unicorn
73	Vincent's Museum	Tom Snyder Productions
68	Volcanoes v1.8	Earthware Comp. Services
66	War in the Middle Earth	Melbourne
67	Wayout	Sirius
63	Wings of Fury	Broderbund
63	Wizardry:Return of Werda	Sir-Tech.
68	Word Attack Plus (IIgs)	Davidson
65	Works (the)	First Star Software
67	Zenith	Softsmith

IBM Most Wanted

75	Empire	Intersil
72	GBA Championship Football	Electronic Arts
68	Graphitti	George Best Phillips Academy
61	Gunship	Microprose
63	Heros of the Lance	SSI
72	Kings Quest III	Sierra
72	Operation Wolf	Taito
72	Radio Baseball	Electronic Arts

The Book of Softkeys (volume 1-3) for only $1 each (* plus postage & handling)

Book of Softkeys I

(Compiled from issues 1-5)

You get all of these softkeys: •Akalabeth •Amper-Magic •Apple Galaxian •Aztec •Bag of Tricks •Bill Budge's Trilogy of Games •Buzzard Bait •Cannonball Blitz •Casino •Data Reporter •Deadline •Disk Organizer II •Egbert II •Hard Hat Mack •Home Accountant •Homeword •Lancaster •Magic Window II •Multi-Disk Catalog •Multiplan •Pest Patrol •Prisoner II •Sammy Lightfoot •Screenwriter II •Sneakers •Spy's Demise •Starcross •Suspended •Ultima II •Visifile •Visiplot •Visitrend •Witness •Wizardry •Zork I •Zork II •Zork III

Plus:
How to crack Applesoft program disks
Introduction to Boot Code Tracing
How to create DeMuffin (A disk cracking tool)
Diskedit (Make changes directly on your disks)
Diskview (Examine protected disks)
Getting on the Right Track (Fix your Apple drive so it tells you what track it is on.)
Hidden Locations on your Apple Revealed
Fix your Rana drive so you can Boot Code Trace
Neat tricks and bombs that you can put in your programs

Book of Softkeys II

(Compiled from issues 6-10)

You get all of these softkeys: •Apple Cider Spider •Apple LOGO •Arcade Machine •The Artist •Bank Street Writer •Cannonball Blitz •Canyon Climber •Caverns of Freitag •Crush, Crumble & Chomp •Data Factory v5.0 •DB Master •Essential Data Duplicator I •Essential Data Duplicator III •Gold Rush •Krell LOGO •Legacy of Llylgamyn •Mask of the Sun •Minit Man •Mouskattack •Music Construction Set •Oil's Well •Pandora's Box •Robotron •Sammy Lightfoot •Screenwriter II v2.2 •Sensible Speller IV •Sensible Speller 4.0c/4.1c •Spy Strikes Back •Time Zone •Visible Computer 6502 •Visidex •Visiterm •Zaxxon •Hayden Software •Sierra OnLine Software

Plus:
How to create DeMuffin Plus (A disk cracking tool)
Super IOB v1.5
Using ProDOS on a Franklin Ace
Crunchlist II
The Controller Saver
Making Liberated Backups That Retain Their Copy Protection
Examining Protected Applesoft BASIC Programs
Game Cheats for: Castle Wolfenstein, Miner 2049er, Serpentine, Star Maze, Ultima II, Wizard and the Princess, Wizardry: Proving Grounds of the Mad Overlord, Zaxxon

Book of Softkeys III

(Compiled from issues 11-15)

You get all of these softkeys: Alien Addition •Alien Munchies •Aligator Mix •Computer Preparation SAT •Cut and Paste •Demolition Division •DLM Software •Einstein Compiler v5.3 •Electronic Arts Software •Escape from Rungistan •Financial Cookbook •Flip Out •Hi-Res Computer Golf II •Knoware •Laf Pak •Last Gladiator •Learning With Leeper •Lion's Share •Mastertype •MatheMagic •Minus Mission •Millionaire •Music Construction Set •One on One •PFS Software •Penguin Software •The Quest •Rocky's Boots •Sabotage •Seadragon •Sensible Speller •Snooper Troops (Case 2) •Soft-Porn Adventure •Stickybear BOP •Suicide •TellStar •Tic Tac Show •Time is Money •Transylvania •Type Attack •Ultima III-Exodus •Zoom Graphix

Plus:
Putting Locksmith 5.0 Fast Copy Into a Normal Binary File
CSaver: The Advanced Way to Store Super IOB Controllers
The CORE Disk Searcher
Modified ROMs
The Armonitor

The Hacker's Ultimate Copy & Deprotection Utility: The Super IOB Collection!

ALL four volumes of Super IOB controllers (through issue #65) in "ONE' package!

COMPUTIST developed the ultimate copy program to remove copy protection from software:

The Super IOB program.

Since the introduction of Super IOB, COMPUTIST has used this flexible program to deprotect (or partially deprotect) dozens of commercial programs with far ranging protection schemes.

Super IOB deprotects disks by using a modified RWTS (the subroutine in DOS which is responsible for the reading and writing of disk sectors) for reading from the protected disk and then using a normal RWTS for writing to the deprotected disk.

This package contains:

•Three disks (supplied in DOS 3.3). Each disk contains at least 60 Super IOB Controllers including the standard, swap, newswap and fast controllers. Also included is version 1.5 of Super IOB, the Csaver program from COMPUTIST No. 13, and a Menu Hello Program that lists the available controllers and, when you select one, automatically installs it in Super IOB and RUNs the resulting program.*

•A reprint of Disk Inspection and the Use of Super IOB, from COMPUTIST No. 17. This article explains how to write your own Super IOB controllers.

•COMPUTIST No. 32, which contains an extensive article detailing the hows and whys of Super IOB v1.5 and at least 5 articles using the new Super IOB program.

Several of the controllers deprotect the software completely with no further steps. This means that some programs are only minutes away from deprotection (with virtually no typing).

The issue of COMPUTIST in which each controller appeared is indicated in case further steps are required to deprotect a particular program.**

*Requires at least 64K of memory.

**Although some controllers will completely deprotect the program they were designed for, some will not, and therefore require their corresponding issue of COMPUTIST to complete the deprotection procedure.

The Super IOB Collection

Volume 1 of the Super IOB collection covers all the controllers from COMPUTIST No. 9 through No. 26. Also included are the newswap and fast controllers from COMPUTIST No. 32. The following controllers are on volume 1:

Advanced Blackjack •Alphabet Zoo •Arcade Machine •Archon II •Archon •Artsci Software •Bank Street Writer •Barrons SAT •Beyond Castle Wolfenstein •BSW //c Loader •Castle Wolfenstein •Computer Preparation: SAT •Dazzle Draw •DB Master 4 Plus •Death in the Carribean •Dino Eggs •DLM Software •Electronic Arts •F-15 Strike Eagle •Fast Controller •Fathoms 40 •Financial Cookbook •Gessler Software •Grandma's House •The Heist •In Search of the Most Amazing Thing •Instant Recall •Kidwriter •Lions Share •Lode Runner •Mastertype •Match Maker •Miner 2049er •Minit Man •Mufplot •Newsroom •Newswap controller •Penguin Software •Print Shop Graphic Library •Print Shop •Rendezvous with Rama •Rockys' Boots •Sargon III •Sea Dragon •Shiela •Skyfox •Snooper Troops •Standard controller •Stoneware Software •Summer Games •Super Controller •Super Zaxxon •Swap Controller •TAC •Ultima I & II •Word Challenge •Xyphus •Zaxxon

Volume 2 of the Super IOB collection covers all the controllers from COMPUTIST No. 27 through No. 38. The following controllers are on volume 2:

Alice in Wonderland •Alphabetic Keyboarding •Alternate Reality •Autoduel •Checkers •Chipwits •Color Me •Conan.data •Conan.prog •CopyDOS •Crisis Mountain •Disk Director •Dragonworld •Early Games •Easy as ABC •F-15 Strike Eagle •Fantavision •Fast controller •Fishies •Flight Simulator •Halley Project •Hartley Software (a) •Hartley Software (b) •Jenny of the Prarie •Jingle Disk •Kidwriter •Kracking Vol II •Lode Runner •LOGO II (a) •LOGO II (b) •Masquerade •Mastering the SAT •Microtype: The Wonderful World of Paws •Microzines 1 •Microzines 2-5 •Miner 2049er •Mist & View to a Kill •Murder on the Zinderneuf •Music Construction Set •Newswap controller •Olympic Decathlon •Other Side •Phi Beta Filer •Pitstop II •Print Shop Companion •RDOS •Robot War •Spy vs Spy •Standard controller •Sundog V2 •Swap controller •Sword of Kadash •Synergistic Software •Tawala's Last Redoubt •Terripin Logo •Threshold •Time is Money •Time Zone •Tink! Tonk! •Troll's Tale •Ultima IV •Wilderness •Word Attack & Classmate •World's Greatest Baseball •World's Greatest Football

Volume 3 of the Super IOB collection covers all the controllers from COMPUTIST No. 39 through No. 53. The following controllers are on volume 3:

2400 A.D. •Alternate Reality: The Dungeon •Address Book •American Challenge •Apple Gradebook V2.6 •Arcade Album #1 •Arcade Boot Camp •Aztec •Bard's Tale II •Beachhead •Beyond Castle Wolfenstein •Black Magic •Blue Powder/Grey Smoke •Borrowed Time •Castle Wolfenstein •Cat-N-Mouse •Catalyst 2.0 & 3.0 •Centipede •Championship Golf •Championship Wrestling •Charlie Brown 123'S •Colonial Conquest •Comprehension Skills •Computer Preparation for the SAT V3 •Coveted Mirror •Coveted Mirror RWTS Capture •CPU68000NV.4Q2 •Cranston Manor •Crime Stopper •Dam Busters •Decimals •Destroyer •Dino Dig •Earth Orbit Station •Encyclopedia Britannica •Escape •Fish Scales •Foundation Course in Spanish •Frog •GFL Football •Goonies •Great American Cross-Country Road Race •Handlers •H.E.R.O •Inferno •Information Master •J-Bird •Jenny's Journeys •Joe Theisman Football •Kindercomp •Kung Fu •Light Simulator •Manuscript Manager •Master Diagnostics IIe •Mastering Math •MECC •MECC Software •MECC Swap •MECC Without RWTS •Missing Links •No Error Check & No DOS •Ogre •One On One •Oregon Trail •Penguin Software •PFS ProDOS •Pitfall II •Puzzles&Posters •Racter •Randamn •Reading Style Inventory •Ring Quest •Science Toolkit •Science Toolkit Rev1 •Skyfox •Snooper Troops •Speed Reader II •Spy's Adventure •Starcross •Super Bunny •Temple Apshai Trilogy •The American Challenge •Translyvania •Troll Courseware •Ultima I •Ultima IV •Wizard's Crown •Word Attack •Word Maze •World Karate Championship •Rings of Zilfin •Zorro

Volume 4 of the Super IOB collection covers all the controllers from COMPUTIST No. 54 through No. 65. The following controllers are on volume 4:

Abracadata/Design Home •Algebra 3 & 4 •American Challenge •Animate side (A) •Animate side (B) •Applewriter Preboot •Aztec •Castle Wolfenstein •Championship Wrestling •Crossword Magic 4.0 •Crypt of Medea •Dam Busters •Dazzle Draw •Death Sword •Deathlord Scenario •Digital Paintbrush Sys •Dondra •Dr Ruth •Dreadnoughts •Elem Social Studies #3 •Elem Social Studies #6 •EOS/Arctic Fox •Estimation •Fraction Factory •Fractions •General Chemistry #8 •GI Joe •Hardball •Lords of Conquest •Matchmaker WGF •Math Blaster •Maxwell Manor •MECC (Apr 89) •Microzine #26 •Mind Over Matter (1984) •Nightmare Gallery •One on One (new) •Path Tactics •Penguin •Pitfall II •Pool 1.5 •Print Shop Data Disk •Project Space Station •PSAT Word Attack Skills •Realm of Impossibility •Roadwar 2000 •Science Prof •Science Toolkit Modules •Science Toolkit •Seaspeller •Songwriter •Space Station •Spell It •Spy Hunter •Sunburst •Super Sun./Gen Manager •Super Sunday •Teachers Rec Book •Tellstar II •Test Maker •The New Oregon Trail •Troll Shop Right •Ultima V (a) •Ultima V (b) •Ultima V (c) •Vocabulary Adv I • Zilfin •

Book of Softkeys ☐ **volume I** ☐ **volume II** ☐ **volume III**

Check the box(s) for the book(s) you want then:

	US/Mexico postage	Canada postage	Foreign postage
One Book	$1.85	$2.28	$5.55
Two Books	$2.40	$3.70	9.19
Three Books	$2.40	$4.32	12.83

add $1 per book _____
plus $.65 handling _____.65
plus the correct postage _____
to get final cost... $ _____

The Super IOB Collection: Includes all four disks with Super IOB version 1.5, COMPUTIST #32, PLUS a reprint of "Disk Inspection and the Use of Super IOB".

☐ US/Canada/Mexico - All four volumes **$30.00** $ _____
☐ Other Foreign - All four volumes **$35.00** $ _____
☐ US/Canada/Mexico - Volume #4 only **$8.00** $ _____
☐ Other Foreign - Volume #4 only **$10.00** $ _____

Total enclosed... $ _____

Name _____
Address _____
_____
City _____ State _____ Zip _____
Country _____ Phone _____
VISA
MC _____ - _____ - _____ - _____ Exp. _____
Signature _____

•Send US funds drawn on US bank. • We ship by UPS so we need a street address. Otherwise, we ship by 1st Class mail which gets there but takes a little longer. • Call (206) 832-3055 to use a credit card or send check/money order to:

COMPUTIST 33821 E Orville Rd Eatonville WA 98328