⚫ RDEX *Features*:

■ **Picture Loader**—by Keith Parker

■ **How To Make DEMUFFIN PLUS**—by Bobby

■ **Convert Print Shop graphics into Print Master graphics**
—by Joseph P. Karwoski

■ **lower case letters For Your Apple II Plus**—by Tim Strelchun

■ **The Product Monitor**—by Jeff Hurlburt

■ **Apple IIc Paddle Fix**—by John B. Walkey

■ **Softkey for** *Daisy Professional:* **'Most-Protected' Award**
—by Terry MacGregor & Joe Walters

■ **DOS EOR MAKER**—by Jean Phillippe Cunniet

⚫ RDEX *Softkeys*:

A2-PB1 Pinball ⚫ Animate ⚫ Bank Street Music Writer ⚫ Boulderdash Construction Kit ⚫ California Games ⚫ Countdown to Shutdown ⚫ Coveted Mirror ⚫ Create with Garfield ⚫ Daisy Professional ⚫ Destroyer ⚫ Donkey Kong ⚫ Expedition Amazon ⚫ General Chemistry Disk #8 ⚫ Graphics Studio ⚫ Green Globs & Graphic Equations ⚫ Kalamazoo Teacher's Record Book 2.0 ⚫ Kids on Keys ⚫ Marble Madness ⚫ Math Blaster ⚫ Maxwell Manor ⚫ Peanut's Maze Marathon ⚫ Petro-Calc ⚫ Police Artist ⚫ Practical Grammar ⚫ Rendezvous ⚫ Ring Quest ⚫ Roadwar Europa ⚫ Roadwar 2000 ⚫ Rocky Horror Show ⚫ Sesame Street Electric Coloring Book Series ⚫ Sesame Street Letters for You ⚫ Sesame Street Numbers ⚫ Seven Cities of Gold ⚫ Snoopy's Reading Machine ⚫ Spy's Adventures In Europe ⚫ Spy's Demise ⚫ Super Sunday Football ⚫ Talisman ⚫ Tellstar II ⚫ Top Draw v1.01A ⚫ The American Challenge ⚫ The Dam Busters ⚫ The Science Professor ⚫ Tubeway ⚫ Vocabulary Adventure I ⚫ Winter Games ⚫ Wizards' Crown ⚫ Zero-Gravity Pinball

IBM RDEX: *Feature:* ● Introduction to IBM Disk Format, Access, and Copy-protection ● Putting Sargon III on harddisk
Softkeys: ■ Prokey 3.0 ■ R:base ■ Time Manager

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:

Attention Apple-users:
Why type those l-o-n-g listings when you can just Upgrade your present sub. to a Mag/5¼"Disk Combo subscription

■ You may upgrade your current subscription to a magazine & disk combination by sending $5.50 ($6.50 foreign) per remaining issue.

Time to renew?

■ Check your mailing label to see if you need to renew your subscription.

☐ Use the order form on this page to renew, and don't forget to send your most current mailing label to help expedite your renewal

Moving soon?

■ If you're moving, let us know at least 30 days in advance.

■ Issues missed due to non-reciept of Change-of-Address may be acquired at the regular back issue rates.

■ Remember, the Post Office does not forward third class mail unless requested. COMPUTIST is not responsible for replacing issues lost while forwarding order is in effect.

April's Fool

Don't be April's "Fool of the month"!

Always make backups. Be an RDEX author. And Subscribe to

COMPUTIST

Notes 'n things......

■ CLOSED FOR VACATION!

NOTICE: The offices of COMPUTIST will be closed **May 4 thru 6, 1988** while Karen (our telephone answerer and chief problem solver) takes a few days of much-needed vacation.

■ Solving Subscription Problems

ATTENTION COMPUTIST Subscribers: Have you been receiving your monthly issues of COMPUTIST? Well, lately, we have heard from a few COMPUTIST subscribers who have not been receiving their issues for up to the last 6 MONTHS!! In a majority of these cases, the subscriber's address was correct and the subscription current, but the issues were still being 'lost in mailing'.

If you've recently had a problem with very late delivery of your COMPUTIST, let us know. We're trying to determine if there is any pattern or reason for the delay.

■ Each issue of COMPUTIST is shipped by the first of the month.

Most subscribers will receive their issue by the end of the third week. So, if more than 30 days have passed or if you are having trouble receiving your monthly issues, don't wait - get in touch with us right away! We'll verify your address and subscription status, and replace UP TO the last 3 months issues of COMPUTIST.

COMPUTIST is not responsible for, and will not replace, issues unclaimed or lost over 90 days of release.

The volunteer customer service department of COMPUTIST...

strives to bring you the quickest and most reliable service. Most orders are shipped within 2 to 3 days of receipt, with delivery normally expected within 2 weeks. If you have a problem or an order is running late, call or write and we'll fix it. Above all, keep a level head.

We lose volunteers when callers are abusive. Their response to such calls is: "Hey, wait a minute. I don't have to listen to that kind of abuse!" and they're gone.

More than anything, COMPUTIST extends it's warmest thanks to all subscribers and contributors. It's your'support that keeps us going.

BUGS

COMPUTIST #51, page 20

Softkey for *2400 AD*. In line 1060, the last POKE statement reads: POKE 474829, 213. Change this to read: **POKE 47829, 13.**

COMPUTIST #52, page 15

Softkey for *Car Builder*. Change 300:4C E3 03 4C D9 03 to 300:20 E3 03 4C D9 03.

COMPUTIST #50, page 23

Softkey for *Oregon Trail*. In the second paragraph somehow some text was left out. It should read:

The address and data epilogs remained DE AA. Using *Controller Writer*, I wrote a controller to copy tracks $03-22. I copied the disk with the controller to an INITialized disk with DOS 3.3 and tried to boot it. My copy just kept rebooting. I examined the HELLO program and saw several REM statements then a last line of CALL 2208. It was evident that I was missing something to make it function properly.

COMPUTIST #54

Table Of Contents

April   **1988**

Features:

Picture Loader........................8
Keith Parker's little 'graphic-grabber' lets you capture 'hidden' pictures from your deprotected software and comes with a track-sector chart on where to find some of them.

How To Make...
DEMUFFIN plus....................11
Bobby's modifications to MUFFIN is reprinted, by popular demand, for those who don't have a copy of the ancient *hardcore COMPUTING* or the first *Book of Softkeys*.

Moving from...
Print Shop to Print Master..........13
Joseph P. Karwoski's easy step-by-step procedure for converting Print Shop graphics into Print Master graphics.

lower case letters...
For Your Apple II Plus..............15
Tim Strelchun's *KEYBOARD ENHANCER* lets you display and type lower case using the **ESC** key as a shift-lock. No hardware needed...

The Product Monitor.................24
Jeff Hurlburt reviews *The Eternal Dagger* and Donald Duck's Playground .adds some quick notes on *Silent Service, Marble Madness, The Bard's Tale IIgs, Mean 18*, and finally warns you to warm up before a cold boot.

Apple IIc Paddle Fix.................28
John B. Walkey's hardware fix shows that not all joystick failures are the fault of the joystick!

Softkey for *Daisy Professional*
'Most-Protected' Award..............30
Terry MacGregor & Joe Walters's teamwork reveals how intricate protection schemes can get. The six different methods used on this package gets it nominated for the 'Most-Protected' Award.

DOS EOR Maker....................37
Jean Phillippe Cunniet's program lets you encrypt data disks using Exclusive ORing with a 'key'.

IBM Feature:

Introduction to...
IBM Disk Format, Access
and Copy-protection................44
David G. Alexander explains IBM floppy formats and some protection schemes used, such as self-modifying code using Interrupt 13.

Readers Data

For serious users of Apple computers: II, II +, //c, //e, IIgs, Macintosh

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 their *DOS 3.3 System Master* disk).

Commands and control keys

In any article appearing in COMPUTIST, 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 specially boxed. An example of both is:

6 ⌐P

Press ⌐6⌐. Next, place one finger on the ⌐·····⌐ key and then press ⌐P⌐. Remember to enter this command line by pressing **RETURN**.

Other special combination keypresses include ⌐RESET⌐ or ⌐⌂⌐RESET⌐. In the former, press and hold down ⌐CTRL⌐ then press **RESET**. In the latter, press and hold down both ⌐CTRL⌐ and ⌐⌂⌐ then press **RESET**.

Special requirements

Special prerequisites for COMPUTIST articles, programs and softkeys are usually listed at the start under:
■ Requirements:

Software recommendations

☞ Applesoft program editor such as *Global Program Line Editor (GPLE)*.

☞ Sector-editor such as *DiskEdit* (in the *COMPUTIST Starter Kit*) or *ZAP* from *Bag of Tricks*.

☞ Disk-search utility such as *The Inspector*, the *CIA* or the *Core Disk Searcher* (in the *COMPUTIST Starter Kit*).

☞ Assembler such as the *S-C Assembler* from S-C software or *Merlin/Big Mac*.

☞ Bit-copy program such as *Copy II Plus, Locksmith* or *Essential Data Duplicator* (EDD).

☞ Text-editor (that produces normal sequential text files) such as *Applewriter II, Magic Window II* or *Screenwriter II*.

☞ *COPYA, FID* and *MUFFIN* from the *DOS 3.3 System Master* disk are also useful.

Super IOB and Controllers

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

Reset into the Monitor

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

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

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

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

Recommended literature:

☞ *Apple II Reference Manual*
☞ *DOS 3.3 manual*
☞ *Beneath Apple DOS*, by Don Worth and Pieter Lechner, from Quality Software
☞ *Assembly Language For The Applesoft Programmer*, by Roy Meyers and C.W. Finley, from Addison Wesley

Keying in Applesoft programs:

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

10HOME:REMCLEAR SCREEN

The LIST will look like:

10 HOME : REM CLEAR SCREEN
...because Applesoft inserts spaces into a program listing before and after every command word or mathematical operator. These spaces usually don't pose a problem except in line numbers which contain **REM** or **DATA** commands. There are two types of spaces: those that have to be keyed and those that don't. Spaces that must be typed appear in COMPUTIST as delta characters (^). All other spaces are there for easier reading. NOTE: If you want your checksums (See **Computing checksums**) to match up, you must key ONLY the ^ spaces after DATA statements.

Keying In 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. A beep means you have typed something that the monitor didn't understand and must, therefore, retype that line.

When finished, return to BASIC with:

3D0G

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

The source code is printed to help explain a program's operation. To key it in, you will need the *S-C Assembler* or you will have to translate pieces of the source code into something your assembler will understand (see table of *S-C Assembler* directives in COMPUTIST #17).

Computing checksums

Checksums are 4-digit hexadecimal numbers which tell if you typed a program exactly as it appears in COMPUTIST.

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 appeared in COMPUTIST #1 and **The Best of Hardcore Computing**. An update to *CHECKSOFT* appeared in COMPUTIST #18.

If the published checksums accompanying program listings and hexdumps do not match those created by your computer, then you typed the program incorrectly. The line where the first checksum differs has an error.

CHECKSOFT instructions:

LOAD *filename*
BRUN CHECKSOFT

Get the checksums with: ⌐&⌐ and correct the program line where the checksums differ.

CHECKBIN instructions:

CALL -151
BLOAD *filename*

Install *CHECKBIN* at an out of the way place

BRUN CHECKBIN, A$6000

Get the checksums by typing the Starting address, a period and the Ending address of the file followed by a ⌐Y⌐.

***SSS.EEE*⌐Y⌐**

Correct the lines at which the checksums differ.

. . . . RDEXed

EXchange

etc... who want all their software backed up and COPYA-able

Keith Parker

Softkey for...

Tubeway
Datamost

■ Requirements

- ☐ 2 blank disks (for part 2 & 3)
- ☐ Sector-editor
- ☐ *COPYA* (from system master)
- ☐ *XFER.BOOT* & *RESTORE* (from COMPUTIST #16)

Tubeway is one of the Tempest type games. Even though it's one of the older games put out by Datamost, in my opinion, it's one of the better Tempest copies. Besides great game play and original screens, it has a very interesting feature, a "soft sound" option. You can turn the sound on, off or on softly. I wish other programs would incorporate this feature. It sure is a blessing when I'm over at a friends house and playing a game on his IIe. Since his IIe doesn't have a volume control, at 12:00 o'clock at night a ⃞G could wake the dead! (It's no fun to play a game without the sound effects.)

I have deprotected this program in two parts. Part 3 is just the addition of the title page at the beginning of the game. However, in order to do part 3, you must first do parts 1 and 2.

Part 1, the *COPYA*-able disk

When I first sat down to deprotect this program, I attempted to copy it with *COPYA*. Boy, talk about a racket! So now knowing that the program used a non-standard DOS, I pulled out *Trax* from *Bag of Tricks* and let it take a look at it. I crossed my fingers, stared at the words "Analyzing Data" on the screen, and hoped that *Trax* would be able to handle it. Of course, it did. It showed that the first byte of the address epilog had been changed. That was the only change on the whole disk, how simple. So knowing the protection used on the program, I pulled out COPYA and told it to ignore epilog and checksum errors. It copied perfectly. I then booted it up. I was alarmed to hear my drive grind. I thought to myself, "Could it be a nibble count?". *The thought of trying to find a nibble count* made a cold chill go down my back. Wait a minute! This program is still trying to read sectors with the altered epilog byte. So I pulled

out my sector editor and read in track $00, Sector $03, and changed byte $91 from DF to DE. It worked!

The program was now completely deprotected. (Incidentally Track $00, Sector $03 is equal to memory addresses $B900 — $B9FF.)

Step-by-Step

☐ **1** Boot your DOS 3.3 system disk.

☐ **2** Tell DOS to ignore checksum and epilog errors and run *COPYA*.

POKE 47426,24
RUN COPYA

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

Trk	Sct	Byte(s)	From	To
$00	$03	$91	DF	DE

You now have a *COPYA*-able disk.

Part 2, The File Copy

Now that I had a *COPYA*-able disk, I thought I might try to capture the game. (Since it is a single load game.) I used *XFER.BOOT* and *RESTORE* to boot the game in auxiliary memory. After the title page came on, I pressed ⃞RESET. I then moved auxiliary to main memory and searched through the code for a keypress routine. (As you know, if you press a key during the title page it'll run the game.) Well, I found the routine and I found the jump to the start of the game at $90C9. I replaced it with a jump to the monitor and then restarted the program. The game loaded and jumped into the monitor. After a little snooping, I found the starting and ending addresses of the game. I then saved it to my work disk and rebooted to try it out. It worked like a charm!

Step-by-Step

☐ **1** With your sector-editor read in track $22, sector $0E of your *COPYA*-able *Tubeway* disk. At byte $CD you should see 4C 00 40, change this to 4C 59 FF (a jump to the monitor). If it's not there, search your disk for the hex string:4C 00 40 AD 1C 91. Change the string 4C 00 40 to 4C 59 FF.

BLOAD XFER.BOOT
PR#3

☐ **2** Insert your *COPYA*-able disk.

Call 768

---

3 Press a key to skip the title page.

4 When the drive stops, press RESET .

```
CALL-151
3F8:4C 00 03
BLOAD RESTORE
900<900.90F8 ⏎Y
A964:FF  Allows large Bfile.
```

5 Save the program.

```
BSAVE TUBEWAY,A$900,L$87F0
```

You now have a file version of *Tubeway*. You can use *FID* to transfer this program to other disks. If you want the title pages and the page flip routine, perform part 3.

Part 3, Title Pages

After completing part 2, I thought to myself, "Some people might want to have the title pages also". So I sat down and produced this softkey. When I was searching through the code in part 2, I found all the routines I needed. All I had to do was to write a little routine to BRUN the file *TUBEWAY*, instead of loading it directly from disk.

Step-by-Step

```
BLOAD XFER.BOOT
PR#3
```

1 Insert your *COPYA*-able disk from part 1, not the file version.

```
CALL 768
```

2 When the title page is loaded and working, press RESET .

```
CALL-151
3F8:4C 00 03
BLOAD RESTORE
2000<2000.95F8 ⏎Y
```

3 Insert the disk with the *Tubeway* file version (from part 2) in the drive.

```
BSAVE PIC1, A$2000, L$2000
BSAVE PIC2, A$4000, L$2000
91D8:4C DC 91
```

(Tell jump routine to go to our loader.)

4 Add the loader routine.

```
91DC: A2 0C BD E8
91E0: 91 20 ED FD CA 10 F7 60
91E8: 8D D9 C1 D7 C5 C2 D5 D4
91F0: CE D5 D2 C2 84
BSAVE FLIPPER,A$913B,L$BA
```

5 Reboot and type in listing 2.

```
10 HOME :D$ = CHR$ (4): PRINT D$ "BLOAD
   PIC1" : PRINT D$ "BLOAD PIC2" : PRINT
   D$ "BLOAD FLIPPER" : POKE 49152,0:
   POKE 49168,0: POKE 49232,0: POKE
   49239,0: POKE 49234,0: POKE 49236,0:
   CALL 37202
```

SAVE STARTUP

That's all there is to it! Now when you want to play *Tubeway*, type:

RUN STARTUP

Softkey for...

Donkey Kong
Atarisoft

■ **Requirements**

☐ COPYA (from system master)
☐ Sector editor
☐ A blank disk

An examination of the disk with *Trax* shows that the epilog bytes have been changed. All that has to be done is to tell DOS to ignore epilog errors and use *COPYA* to copy the disk.

Step-by-Step

1
RUN COPYA

2 At the prompt press ⌷C

3 Delete line 70.

```
70
```

4
```
CALL-151
B942: 18
E003G
RUN
```

Follow the prompts.

5 Using your sector-editor make the following changes:

Trk	Sct	Byte(s)	From	To
$00	$03	$35	?	DE
		$3F	?	AA
		$91	?	DE
		$9B	?	AA

That's all there is to it!

More DOS tricks

Here are a few more DOS tricks to add to the table compiled by Mr. Hart in COMPUTIST #43, page 23:

B989: EA EA	*Ignore address checksum*
B992: EA EA	*Ignore address trailer*
B99C: EA EA	
B920: EA EA	*Ignore data checksum*
B8DF: EA EA	*Ignore the long gap between address and data*

Grab 'Hidden' Hi-res Images with...

Picture Loader

Have you ever wanted to load up a title page from an unprotected commercial program either to print it out on your printer or to put messages such as: "Kids Copy", "Backup Copy", or "Backuped on month/day/year" on the title page? Maybe you would like to save the title pages to disk and create a "slide show" showing all of the programs you've deprotected. There are all sorts of uses for this program.

This program can only be used to load pictures that aren't crunched (to conserve disk space). Most pictures (that aren't crunched) are stored standard (increasing tracks and sectors) or backward (decreasing tracks and sectors). So, I've included both "controllers" in the main program listing. Some programs store their pictures in a different way than what this program can load. So, you can make your own "controller" for that specific program. I'll explain everything in a little while.

Type the Picture Loader program and save it.

SAVE PICTURE LOADER

PICTURE LOADER

```
10 HOME : IF PEEK (768) < > 169 THEN PRINT CHR$
   (4); "BLOAD°DOS°UTILITY"
20 HTAB 14: INVERSE : PRINT "PICTURE°LOADER" :
   NORMAL : HTAB 13: PRINT "BY:°KEITH°PARKER"
30 VTAB 7: INPUT "SLOT,DRIVE-=>" ;SL,D
40 INPUT "STARTING°TRACK,SECTOR-=>" ;T.S
50 INPUT "(R)EAD°OR°(W)RITE°DATA?-=>" ;C$
60 IF C$ = "R" THEN C = 1
70 IF C$ = "W" THEN C = 2
80 IF C$ <> "R" AND C$ <> "W" THEN 50
90 INPUT "READ°OR°WRITE°(F)ORWARD°OR°(B)
   ACKWARD->" ;WA$
100 IF WA$ <> "F" AND WA$ <> "B" THEN 90
110 HTAB 7: PRINT "INSERT°DISK°AND°PRESS°A°KEY"
    ;: GET A$
120 Y = 8192: POKE 49232,0: POKE 49239,0: POKE
    49236,0: POKE 49234,0
130 FOR I = 1 TO 32: GOSUB 160:Y = Y + 256: NEXT
    : IF C$ = "W" THEN TEXT : VTAB 12: HTAB 16:
    PRINT "ALL°DONE." : END
140 GET A$ : IF A$ = CHR$ (27) THEN TEXT : VTAB 13:
    HTAB 9: PRINT "BE°SURE°TO°SAVE°PICTURE." :
    END
150 GOTO 120
160 POKE 784,T: POKE 789,S: POKE 779,D: POKE
    774,SL * 16: POKE 799,Y - INT (Y / 256) *
    256: POKE 804, INT (Y / 256): POKE 794,C
170 CALL 768
180 IF WA$ = "B" THEN GOTO 240
```

```
190 REM STANDARD LOADER
200 IF S = 15 THEN T = T + 1:S = -1
210 IF T = 35 THEN TEXT : VTAB 13: HTAB 13: PRINT
    "END°OF°THE°LINE." : END
220 S = S + 1: RETURN
230 REM BACKWARD LOADER
240 IF S = 0 THEN T = T - 1:S = 16
250 IF T = -1 THEN TEXT : VTAB 13: HTAB 12: PRINT
    "START°OF°THE°LINE." : END
260 S = S - 1: RETURN
```

Checksums

10	- $905B	140	- $99BD
20	- $6B79	150	- $7980
30	- $327A	160	- $EF9D
40	- $0F56	170	- $AC62
50	- $8323	180	- $2CB2
60	- $EE35	190	- $5002
70	- $83C5	200	- $CAFF
80	- $E66A	210	- $22F2
90	- $E7BD	220	- $BDAA
100	- $4F8A	230	- $C07B
110	- $1E56	240	- $0930
120	- $4AE1	250	- $A65A
130	- $1129	260	- $CD96

Now enter the machine language routine.

CALL -151

DOS UTILITY

0300: A9 00 8D EB B7 A9 00 8D	$689F
0308: E9 B7 A9 00 8D EA B7 A9	$39BD
0310: 00 8D EC B7 A9 00 8D ED	$90D4
0318: B7 A9 00 8D F4 B7 A9 00	$DA09
0320: 8D F0 B7 A9 00 8D F1 B7	$7009
0328: A9 B7 A0 E8 20 D9 03 90	$2FD0
0330: 05 AD F5 B7 85 06 60	$41DF

**BSAVE DOS UTILITY, A$300, L$37
3D0G**

How *PICTURE LOADER* works

Here's a line-by-line explanation of how it works.

10	Check to see if DOS UTILITY is loaded or not. If not, load it.
20	Print title and byline.
30	Input slot and drive of the disk containing the picture.
40	Input starting track and sector to read or write. Enter numbers in Decimal.
50	Enter command (Read or Write).
60-70	Change command to a number, so DOS UTILITY can use it.
80	If no command chosen, ask again.
90	Enter which "controller" you want to use.
100	If neither "controller" chosen, then ask again.
110	Final prompt.

120	Turn on Hi-Res page 1 and set loading address to 8192 (S2000).
130	Read or write 32 sectors. Increment loading address by 256 (1 sector), each time a sector is read or wrote. If command is "write" stop after 32 sectors, if not continue.
140	Get key to continue reading. If **ESC** is pressed, quit.
150	Jump to loading routine.
160	Set up DOS UTILITY.
170	Call DOS UTILITY to read or write a sector.
180	If controller equals "backward", jump to the start of the controller.
190	STANDARD LOADER
200	If sector = 15 then increment track. Set sector = -1.
210	If track = 35 (last track) then end. (Real end = 34.)
220	Increase sector by 1 and return to line 130.
230	BACKWARD LOADER
240	If sector = 0 then decrement track. Set sector equal to 16.
250	If track = -1 (beginning track) then end. (Real beginning is 0.)
260	Decrease sector by 1 and return to line 130.

How *DOS UTILITY* works

This program uses DOS's built in IOB to read and write sectors to disk. Here is a table to show how to use the *DOS UTILITY* program.

Dec Addr	What It Does
784	Track number
789	Sector number
779	Drive number
774	Slot number * 16
799	Low order byte of memory bank
804	High order byte of memory bank
794	Command (01=read, 02=write)
6	Error code (See Table 1)

DOS UTILITY sets up the IOB table, so that you can read a specific track and sector. Lastly, it jumps to the vector at $3D9. The carry flag is then checked for an error. If there is an error it's stored at $06.

With this information you should be able to use *DOS UTILITY* for your own purposes.

Using the program

Run *PICTURE LOADER* and, if needed, it will load in *DOS UTILITY*.

The first prompt asks you to enter the slot and drive of the unprotected disk you wish to search (Example: 6,2 (for slot 6, drive 2)).

The second prompt asks you to enter the track and sector you wish to start reading or writing (When looking for a picture, I usually start reading at track 1, sector 0.).

The third prompt asks if you want to read or write data. Enter "R" for read or "W" for write. (NOTE: If you wish to write data, your picture must already be on Hi-Res page 1.)

The last prompt tells you to insert your disk and press a key. If you're reading data, you can press any key after the program has loaded in 32 sectors to continue searching.

If you press **ESC**, the program will stop. After stopping, if you want to save the picture type in the command:

BSAVE PICTURE NAME,A$2000,L$1FFF

Finding a picture

The first thing you need to do is to boot up the disk that you wish to search. Look at the picture carefully! Pick out a big part of the picture, so you'll be able to recognize it easily.

Now run *PICTURE LOADER* and start searching for your picture. Once you think you've found the picture, you may have to read in a sector before or a sector after to get the picture in perfect.

What do you do when the picture has no "foreign" streaks in it, but it looks messed up? Look at any text on the picture. If it looks like it has been flipped end over end, try loading it with the backward "controller".

NOTE: You have to add 2 to the track value and subtract 1 from the sector value, in order for the program to stop on your "old" starting address. Sometimes when you subtract 1 from the sector value this will decrease the track number. I.E. track 01, sector 00 backwards would be track 2, sector 15.

A little help

In order to give you a little help in using this program, I've compiled two lists. One list is for the forward "controller" and the other list is for the backward "controller". Hopefully you have one or more of the programs listed.

For the Forward Controller

	track	sector
The Hobbit title	1	0
AE screens: 1	1	0
2	5	0
3	9	0
4	13	0
5	17	0
6	21	0
7	25	0
8	29	0
Swashbuckler title	7	0
Swashbuckler screens: 1	21	0
2	19	0
3	5	0
Fantavision title	23	0
Sammy Lightfoot title	2	0
more....		

	Trk	Sct
B.C.'s Quest for Tires titles: 1	9	Ø
2	11	Ø
Might and Magic titles: 1	1	Ø
2	1Ø	Ø

For the Backward Controller

	Trk	Sct
The Bard's Tale (play screen)	29	15
Donkey Kong titles: 1	13	7
2	16	15
Donkey Kong screens: 1	19	15
2	22	15
3	25	15
4	28	15
Arcticfox title	17	15
Zaxxon title	16	6
G.I. Joe (either side)	34	15
Wilderness title	28	15
Buck Rogers title	16	15

These locations worked for my versions, but as always, they may not work for your version(s).

Conclusion

Now that you know all about *Picture Loader*, I hope you enjoy it as much as I do. Creating "controllers" for the program is very simple. After all there are only two variables that you have to work with.

(NOTE: The program can use two controllers at the present time. Controller 1 is in lines 19Ø-22Ø and controller 2 is in lines 23Ø-26Ø.) A minimal amount of programming knowledge is required.

Table of variables

SL = slot number
D = drive number
T = track
S = sector
C = sommand (Ø1 or Ø2)
WA$ = forward or backward
A$ = key press (checked once every 32 sectors)
Y = loading address

TABLE 1   Error codes

$1Ø disk write-protected
$2Ø volume mismatch error
$4Ø drive error (Example: I/O error)
$8Ø read error (RWTS will try 48 times to read.)

TABLE 2   IOB table

$B7E9 slot number * 16
$B7EA drive number
$B7EB expected volume number (ØØ is a "wild card" any number is accepted.)
$B7EC track number
$B7ED sector number
$B7FØ low-order byte of data buffer
$B7F1 high-order byte of data buffer
$B7F4 command code
$B7F5 error code
$B7F6 actual volume number
$B7F7 previous slot accessed
$B7F8 previous drive accessed

James W. Heil

Softkey for...

The Science Professor
Bergwall Educational Software

■ **Requirements**

☐ *Super IOB* 1.5
☐ 1Ø disks INIT'ed as Boot
☐ *The Science Professor*

This is a softkey for *The Science Professor* by Bergwall Educational Software, Inc. V2.Ø 1986 series of Uniondale, N.Y..

A teacher friend of mine asked me to see if I could try to make a backup of this very good, but expensive, software since very young students were using it. I gave it a try, since I have been reading COMPUTIST since day one and have previously cracked some programs. This was a new experience, though.

I got out my trusty *CIA* program and attempted to look at the disk. Well, while examining track $11 sector $ØF, I got that dreaded IO Error. I then looked at the track data with the Linguist and saw garbage except the headers looked normal, so why the error?

Well, the epilogs were changed, for one thing. The normal DE AA were changed to C9 AA CF. No problem here. Wrong, IO Error again. Plug in ØØØØ and N into *Tricky Dick* and it read a screen full of garbage again. However, track $11, sector $ØF did have a pattern to it, even though it was garbled.

Upon examining the data with the *Linguist*, I noticed every sector of data started with $FF for the first byte of data. This is a non-valid character right? This needed some more looking into.

So now lets examine track $ØØ since it does boot. Track $ØØ, sector $ØØ to $Ø9 were normal and easily read without error. I read each of these sectors in and wrote them out to a blank but INIT'ed disk. Then I booted this disk and stopped the computer after it chattered a while. I printed out $B6ØØ to $BFFF from

the monitor as a disassembled printout and started comparing the data with a normal printout.

Well, there in the read routine I found a jump $BECA command. At $BECA I found some resembling code but still changed. The $FF was being checked for a negative value. Once found go back and continue reading good data. So at last, I think, I have found their trick.

All I had to do now is INIT a new disk and poke the changed data into my IOB from *Super IOB*, read the protected disk and write out a normal disk. With the changed $C9 in the epilog, and I needed to poke the checksum error routine so it ignored checksum errors, I was on my way. Well, to my surprise it worked with no problem. The disks did not run properly. I had to INIT the disks with a Hello program called Boot.

So the first step in the procedure is to INIT 1Ø disks with Boot. Then enter my controller and save the controller as SCIENCE.CON for all the disks unless you want to do them all at once. Then run *Super IOB* and you have good copies of this wonderful program to let all kids use without having to go gray while worrying about how you are going to afford to replace damaged disks.

The controller is standard, for the most part, with a few exceptions. Line 1Ø1Ø is changed to copy track 3 to 34 and lines 1Ø25,1Ø26,1Ø27 and 1Ø55 are added to put the changed code into the read routine area, to ignore the checksum and at 1Ø55 to normalize the epilogue area.

Controller

```
1000 REM SCIENCE CONTROLLER MODIFIED TK=0 TO
     TK=3

1010 TK = 3:ST = 0:LT = 35:CD = WR

1020 T1 = TK: GOSUB 490

1025 POKE 47505, 201: POKE 48842, 189: POKE
     48843, 140: POKE 48844, 192: POKE 48845,
     16: POKE 48846, 251: POKE 48847, 189: POKE
     48848, 141: POKE 48849, 192: POKE 48850,
     160: POKE 48851, 86: POKE 48852, 189: POKE
     48853, 140

1026 POKE 48854, 192: POKE 48855, 48: POKE
     48856, 251: POKE 48857, 189: POKE 48858,
     140: POKE 48859, 192: POKE 48860, 16: POKE
     48861, 251: POKE 48862, 169: POKE 48863, 0:
     POKE 48864, 72: POKE 48865, 104: POKE
     48866, 76: POKE 48867, 01: POKE 48868, 185

1027 POKE 47426, 24: POKE 47348, 234: POKE
     47349, 189: POKE 47350, 140: POKE 47351,
     192: POKE 47352, 16: POKE 47353, 251: POKE
     47354, 201: POKE 47355, 173: POKE 47356,
     208: POKE 47357, 232: POKE 47358, 76: POKE
     47359, 202: POKE 47360, 190

1030 GOSUB 430: GOSUB 100:ST = ST + 1: IF ST < DOS
     THEN 1030
```

```
1040 IF BF THEN 1060
1050 ST = 0 : TK = TK + 1 : IF TK < LT THEN 1030
1055 GOSUB 220
1060 GOSUB 490 : TK = T1 : ST = 0
1070 GOSUB 430 : GOSUB 100 : ST = ST + 1 : IF ST < DOS
     THEN 1070
1080 ST = 0 : TK = TK + 1 : IF BF = 0 AND TK < LT THEN
     1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "DONE⁀WITH⁀COPY" : END
```

Controller Checksums

1000	– $356B	1050	– $5C28
1010	– $3565	1055	– $A866
1020	– $C21D	1060	– $4BFB
1025	– $DF74	1070	– $43FE
1026	– $DCAF	1080	– $57BF
1027	– $12DF	1090	– $82F5
1030	– $04DE	1100	– $8202
1040	– $525B		

Don Christensen

I used Dudley Brett's softkey for *Kindercomp* from issue #51 and it worked fine.

Softkey for...

Police Artist
Sie-Tech

Softkey for...

Petro-Calc
Gulf Publishing

■ **Requirements**

☐ *DEMUFFIN PLUS* (see ednote following this letter)
☐ *XFER.BOOT & RESTORE* (from COMPUTIST #16)
☐ blank disk

NOTE: This softkey will also make non-working but crackable copies of *Best of Muse* and *ABM*, also by Muse.

[1] Initialize a blank disk with DOS 3.3.
INIT HELLO

[2] Insert disk with crack programs:
BLOAD XFER.BOOT

[3] Insert disk to be cracked:
PR#3
CALL 768

[4] Stop the boot when the prompt appears.
[RESET]

[5] Re-enter monitor, insert cracking disk and get busy.

CALL-151
BLOAD RESTORE
3F8:4C 00 03
6800<B800.BFFF☐Y
BLOAD DEMUFFIN PLUS
B800<6800.6FFFM
803G

Use the = as a wildcard for the filename. That's it. I bet this will work for lots of other programs when I have time to try it.

As far as your including IBM softkeys in COMPUTIST, as a charter subscriber, I say do whatever you have to in order to keep the magazine going and don't get down.

Finally, a big thanks to the volunteer staff. You are doing a great job!

🏃 *A lot of you are using DEMUFFIN PLUS (from the ancient Hardcore Computing Premier Issue or the Book Of Softkeys Volume I), so we're reprinting the directions for making it.* .... RDEXed

---

Bobby

How to make...
DEMUFFIN PLUS

■ **Requirements**

☐ Apple II, II+, IIe with 48K
☐ One disk drive
☐ Blank initialized disk
☐ *MUFFIN* (from DOS 3.3 System Master)
☐ Programmers Aid ROM (or Integer BASIC from DOS 3.3 System Master)

MUFFIN is a program used to transfer 13-sector files to 16-sector disks. To accomplish this, it contains an image of 13-sector RWTS (Read or Write a Track and Sector). It uses this internal RWTS to read the file and then writes to your diskette using the external or resident DOS.

DEMUFFIN is created by changing the jumps in the program so that it uses the resident DOS to read and the internal RWTS image to write. Also, the internal RWTS is changed from a 13-sector image to a 16-sector image.

[1] Boot a DOS 3.3 System Master disk.

[2] Select Integer and load MUFFIN.
INT
BLOAD MUFFIN

[3] Enter the monitor and initialize the Programers Aid Code-Relocation feature.
CALL -151
D4D5G

[4] Tell the relocate routine what we're moving and where it goes.

1900<B800.BFFF☐Y *

[5] Move the first Code segment down to $1900.
1900<B800.BA10☐Y

[6] Move the data segment.
.BC57M

[7] Move the last code segment.
.BFFF☐Y

[8] Make the following patches.
1155:00 1E
115B:D9 03
1197:A0 20
15A0:A0 D2 C5 D3 C9 C4 C5 CE
15A8:D4 A0 C4 AE CF AE D3 AE
15F7:C4 C5
20A0:A9 1E 8D B9 B7 20 FD AA
20A8:48 A9 BD 8D B9 B7 68 60

[9] Save the new *DEMUFFIN* program.
BSAVE DEMUFFIN,A$803,L$1900

DEMUFFIN directions are identical to *MUFFIN*'s. You want to convert the files, so you have to select the disk slot and drive and the file name you want to convert. It is helpful to type = for the file name so that all the available file names will be displayed when you answer Yes to the question: "Do you want prompting?". Now you can choose which files to convert (transfer).

If a file doesn't transfer and you get an I/O ERROR, you'll be returned to the menu. Just repeat all the steps but don't bother converting the files you have already converted, and *bypass the problem file* (it probably isn't needed by the program anyway). Continue to convert as many files as you can.

Richard Granback

Softkey for...

The American Challenge
Mindscape Inc.

■ **Requirements**

☐ Super IOB 1.5
☐ A blank disk

■ **Optional:**

☐ *Locksmith 6.0* (Automatic Boot Tracer)
☐ *Copy II Plus* or sector-editor with disk-search capability
☐ *Controller Writer* (COMPUTIST #16)
☐ *Apple II Reference Manual* by Apple Computer
☐ *Beneath Apple DOS* By Quality Software

I tried the softkey in COMPUTIST #40 for *The American Challenge* on my disk and found Mindscapes copy-protection must have taken a

different tack. Although the program would boot properly, it would display "UNAUTHORIZED COPY" and flounder, after the desired race course was selected.

At this point, I decided to try and remove the copy-protection myself. The following is a description of that process. Those wanting to skip through the hows and whys can go directly to the step-by-step procedure.

The first thing I checked was the disk formatting to see what had been changed and what would be needed to convert it to a standard DOS 3.3 format.

Using the *Copy II plus* nibble editor, and referring to *Beneath Apple DOS* as a guide, I looked at the data, including address and data prolog and epilog marks. From this I was able to determine that Mindscape had not changed the formatting from the previous version, leaving epilogs of FF FF as the only formatting problem to contend with (well within the capabilities of *Super IOB*).

With this information in hand, I used COMPUTIST's *Controller Writer* and selected normal DOS default values for everything except for epilogs of FF FF. Using the resulting controller, *Super IOB* produced a normal DOS 3.3 formatted disk. This disk is easier to work with and also spares my original from any mistakes that might happen.

Now we can tear into the program itself, hopefully leaving the protection scheme dead in the water. In order to know about where in the program to start looking, several observations were needed as to what point the protection might come into play.

Using the softkey from COMPUTIST #40 for additional clues, I first looked at track $ØØ sector $ØB for the bytes Larry listed. As expected, I found them to be the same. I decided to test boot both the original and the copy before making any changes, carefully noting any differences in the operation of the two. To my surprise, the copy ran fine up to the point where the desired race course was selected.

Only after this point did the disk head move back to track $ØØ and display the dreaded "UNAUTHORIZED COPY" message.

Other tests included booting with the copy and then swapping in the original disk just before selecting the course.

I also tried swapping the copy back in immediately after selecting the course, just when the disk head began stepping away from track $ØØ. Both tests resulted in the program operating perfectly through a complete race.

Using these clues, I was able to determine the following about the protection scheme used.

1. Parts of the old protection code on track $ØØ described in COMPUTIST #40 were still there, but no longer used.

2. The new protection code did not come into action until after a race course had been selected.

3. Only track $ØØ was involved in any nibble-count or checksum operation.

At last, the attack could begin. Using *Locksmiths'* Automatic Boot Tracer for ease in tracing the code, the search began at the point where the program waited for keyboard input after the course menu had been displayed.

Using **RETURN** as the key input (to help keep things simple), I followed the code through several subroutines (some of the starting locations are listed here).

$8EB9	*Wait for keyboard input*
$8E3Ø	*Decode the input key*
$8E67	*Restore X & Y registers then JSR to $688B*
$688B	*Start of disk access routines*

In trying to follow the code starting at $688B, I quickly found myself in a maze of subroutines which turned on the disk drive, moved the head to track $ØØ, read in the data, and then proceeded to perform some complicated testing of that data.

Based on the clues obtained earlier, I knew that somewhere in there the elusive code was hiding. Since this was a JSR (a jump to a subroutine and return), and not a simple JMP (jump without returning), something else was actually responsible for deciding what to do with the results of the analysis of track $ØØ. I decided to go back to the JSR instruction and continue from there. The code listed below is what I found.

```
$8E6C  JSR  $688B   Disk check routines.
$8E6F  LDA  $CØE8   Turn off disk drive.
$8E72  BCC  $8E77   Branch if carry clear.
$8E74  JMP  $02F6   Jump to $02F6 (DOOM!)
$8E77  PLA          Program continues.
```

Upon returning from the subroutine, the program turned off the disk drive (accessing location $CØE8) and wound up jumping to $02F6 if the carry bit was set. A quick trace from $02F6 confirmed that this was indeed not the path to enlightenment.

At this point two options were available, bypass the jump to $02F6 only, or try and bypass the entire protection subroutine. I chose the latter in order to skip the delay caused by the track $ØØ analysis. Change the JSR $688B (20 8B 68) to JMP $8E77 (4C 77 8E) where the program would go if everything was alright.

All that remains is to locate the code on the disk and change it. Using the Copy II+ scan for bytes feature, I scanned the disk for the bytes 20 8B 68 and found the dastardly code on track $Ø7, sector $ØE, byte $6C. I changed the code to a more suitable 4C 77 8E and my copy of *American Challenge* worked fine. Later using the *Controller Writer*, I changed the controller to add the sector edit.

Trk	Sct	Byte(s)	From	To
$Ø7	$ØE	$6C	20 8B 68	4C 77 8E

The Procedure

[1] Enter the controller using your favorite method and save it.

[2] Start up *Super IOB 1.5* and merge the controller. Insert the original and blank disks when requested and copy the program.

Controller

```
1000 REM AMERICAN CHALLENGE CONTROLLER
1010 TK = Ø:LT = 35:CD = WR:MB = 151
1020 ST = Ø:T1 = TK: GOSUB 490: RESTORE : GOSUB
     190: GOSUB 210: GOSUB 170
1030 GOSUB 430: GOSUB 100:ST = ST + 1: IF ST < 16
     THEN 1030
1040 IF BF THEN 1060
1050 ST = Ø:TK = TK + 1: IF TK < LT THEN 1030
1060 GOSUB 310: GOSUB 230: TK = T1:ST = Ø: GOSUB
     490
1070 GOSUB 430: GOSUB 100:ST = ST + 1: IF ST < 16
     THEN 1070
1080 ST = Ø:TK = TK + 1: IF BF = Ø AND TK < LT THEN
     1070
1090 IF TK < LT THEN 1020
1100 HOME :A$ = "ALL DONE" : GOSUB 450: END
5000 DATA  213,170,150,213,170,173,255,
     255,255,255
5010 DATA 3 CHANGES
5020 DATA 7,14,108,76
5030 DATA 7,14,109,119
5040 DATA 7,14,110,142
```

Controller Checksums

1000	-	$356B	1080	-	$5DC3
1010	-	$85C6	1090	-	$1A37
1020	-	$FC63	1100	-	$ØC37
1030	-	$AD71	5000	-	$94BØ
1040	-	$71B6	5010	-	$9D77
1050	-	$B973	5020	-	$ØA8D
1060	-	$71EF	5030	-	$5A68
1070	-	$2BF4	5040	-	$Ø3BØ

Sheldon M. Atterbury

† *Softkey for...*

Talisman
Polarware

■ **Requirements**

☐ Super IOB Collection

If you wish to deprotect *Talisman* by Polarware and have the *Super IOB* Collection, just use the controller for *Sword of Kadash* and press "B" when asked for the disk side.

Softkey for...

Bank Street Music Writer
Mindscape

1 Boot your DOS 3.3 system disk.

2 Tell DOS to ignore checksum and epilogs and run *COPYA*.

```
POKE 47426,24
RUN COPYA
```

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

Trk	Sct	Byte(s)	From	To
$00	$05	$4C-4E	20 44 B9	EA EA 60

The back side is unprotected so *COPYA* will work. (I find it strange that they copy-protected a disk that requires their included card, to be plugged into the computer.)

Peter Wilding

Softkey for...

Vocabulary Adventure I

The freeing of programs from the chains of copy-protection is one of the most challenging and rewarding facets of personal computing.

Vocabulary Adventure I is a program which teaches vocabulary to fifth through seventh graders. The game claims to teach the student by including new vocabulary with each description of a room in a 50-room castle. This vocabulary is then tested in a multiple choice quiz. A right answer earns a treasure, a certain number of points, and passage to the next room. A wrong answer, on the other hand, causes the student to miss a few rooms, which decreases his chances at getting the most points. I found the game to quickly become boring, because the only objective was to get as many points as possible and thus get into the Hall of Fame. With no alternate paths through the castle, playing the game more than once is virtually unbearable.

Despite the game's drabness, I will detail what to do in order to crack it so that you may apply this information to cracking other programs as well.

The protection on *Vocabulary Adventure I* involves a change in the Address and Data prologs on the disk. These changes are BA AA 96 (Address Field header), and D5 AA 96 (Data Field header). Notice that the protection attempts to fool copy programs into thinking that the data field is really the address field, resulting in a bad copy being produced.

Many beginners in the field of cracking

(myself included) don't quite know where to begin when they write a controller for *Super IOB*. As a result of this, I have developed a controller which I call the All-Purpose Controller. To use it, simply stick in the changed Address and Data header values and away you go.

But (you say), what if the epilogs (trailers) have also been changed? Well, why not just ignore them completely? Just type the following and you will ignore both the Address and Data epilog fields entirely:

```
CALL -151
B992: EA EA          Ignore address trailer
B936: EA EA          Ignore data trailer
3D0G
```

This way you can copy just about any disk with changed marks.

1 Initialize a disk. This will be your copy to disk.

2 Install the Vocabulary Controller in *Super IOB 1.5*. Don't bother making the two changes mentioned above, they're not needed to crack this program.

3 Run *Super IOB 1.5*. (Don't format the disk.)

That's all. Examine the program (it's mostly written in BASIC!).

Controller

```
1000 REM VOCABULARY CONTROLLER
1010 TK = 3 : LT = 35 : ST = 15 : LS = 15 : CD = WR : FAST
     = 1
1020 RESTORE : GOSUB 190 : GOSUB 210 : GOSUB 490 :
     GOSUB 610
1030 GOSUB 230 : GOSUB 490 : GOSUB 610 : IF PEEK
     (TRK) = LT THEN 1050
1040 TK = PEEK (TRK) : ST = PEEK (SCT) : GOTO 1020
1050 HOME : PRINT "COPY⁴DONE!" : END
5000 DATA 186,170,150
5010 DATA 213,170,150
5100 REM LINE 5000 IS THE NEW ADDRESS HEADER
5110 REM LINE 5010 IS THE NEW DATA HEADER
```

Controller Checksums

1000	- $356B	1050	- $945E
1010	- $2445	5000	- $6C51
1020	- $DAB7	5010	- $F944
1030	- $EF2E	5100	- $B1F1
1040	- $F76F	5110	- $F8CA

Joseph P. Karwoski

Convert *Print Shop Graphics* into *Print Master Graphics*

In September, 1987, I purchased *Printmaster Plus* and found it to have some

advantages over *Print Shop*. One disadvantage was that I could not use my *Print Shop Graphics* with *Print Master*. I have a fair size collection of *Print Shop* graphics.

I decided that there had to be a way of converting my *Print Shop Graphics* to *Print Master Graphics*. To my surprise, it was easier than I thought! Here is a step-by-step way of converting *Print Shop Graphics* to *Print Master Graphics*:

1 Format a disk in ProDOS, use PMDATA as the volume name.

2 Boot *Print Master Plus* and create an Art Lib on your newly formatted disk. I used MYLIB.

3 Boot *Copy II Plus* (Note: When you catalog your data disk, the subdirectory should be MYLIB.AL:B. You must use *Print Master* to create the subdirectory!)

4 Copy your *Print Shop Graphics* to your new data disk - copy them into the subdirectory MYLIB.ALB (Note: *Copy II Plus* will convert the files from DOS 3.3 to ProDOS for you)

5 Rename all the files with a ".GRA" extension. Example:

Print Shop name: *BEAR*
Print Master Plus name: *BEAR.GRA*

That's all there is to it - simple, is it not? Good luck, and have fun converting the files.

Lawrence A. Marks

As a recent subscriber to your excellent magazine, COMPUTIST, I am attempting to locate the following software (mentioned in your publication): *Bag Of Tricks, CIA, S-C Macro Assembler*

Also, I have not been able to purchase the latest version of ProDOS and DOS 3.3 System Utilities. Apple dealers are of little or no help in locating these utilities. Hopefully, you may have some information on how to obtain these Apple utilities.

⚓ Bag of Tricks 2 by Quality Software ($45 plus $2 shipping) and S-C Macro Assembler ($100) can be obtained through the S-C Software Corporation; P.O. Box 280300; Dallas, TX 75228

Regarding The CIA, the software company does not seem to be answering any mail. Our suggestion would be to contact various clubs. Perhaps a member would like to sell their copy. A club is also your best bet for obtaining the latest version of the DOS 3.3 and ProDOS System Master disk. With the exception of a very few, Apple dealers are uniformly unhelpful unless you want to buy some software or hardware at their inflated prices.
...............................RDEXed

Eric Delbridge

Softkey for...

Seven Cities of Gold
Electronic Arts Software

■ **Requirements**

☐ Apple II-ish computer
☐ Sector-editor with 'scan' capabilities (*Copy II Plus*)
☐ Blank disk
☐ Fast copier with 'skip-track' abilities.

When I received *Seven Cities of Gold* (SCG), I searched through my COMPUTISTs for a softkey to no avail. I would have to do this one myself.

Electronic Arts used its old nibble-count and imaging of track $06. This is easily defeated by skipping tracks $05—06 and changing the restart codes so the program will just continue with the game. There are several different versions of SCG, but this crack should work for all of them.

1 Using a fast copier, copy both sides of SCG, ignore errors on tracks $05—06.

2 On side A, scan track $01 for the code in the following table. Scan for each one, one at a time. The format for the changes is 18 60 xx, the xx being the checksum. (It is possible to make a working copy without all the changes listed, but the game will restart after a certain amount of time.)

Trk	Sct	Byte(s)	From	To
$01	see text		AC 69 A0	18 60 DD
			AC 69 05	18 60 42
			20 FB A0	18 60 40
			20 FB 05	18 60 A5
			20 FE A0	18 60 46
			20 FE 05	18 60 AB
			20 03 A0	18 60 4B
			20 03 05	18 60 B0

Playing Tips for...

Beauracracy
Infocom

■ When you are at the airport with an Omnia Gallia ticket, you should go straight to the Omni Gallia desk. But OG has been bought out by Air Zalagasa. To find the Air Zalagasa desk, go through lost-and-found then go north.

■ It doesn't matter how early you get to the desk, there will always be a fat man there who will cause you to miss the plane. No matter, just wait and then exchange your ticket for an Air Zalagasa ticket at the desk.

■ Now go south one move and climb the pillar. You will eventually come to the tower. When the speaker crackles "Air Zalagasa flight 42 requests take off permission", say, "Controller, permission denied." Go back down to the top of the pillar. Examine the speaker. Pull the black wire. Pull the red wire. Connect the red and black wires. Now you will have 11 points and you'll be on the plane.

■ Once on the plane, move to seat 3C. When the stewardess brings you the llama stew, don't eat it, but move to 8D. Then press the light button there, causing the seat in front of 3C to recline, thus spilling the soup so you don't have to eat it. The rest is easy. Make sure you know the order that the postage marks on the mail were in!

Playing Tips for...

Maniac MansionL
Lucasfilm (Activision)

■ The key to the door is under the mat.
■ To open the door that has no knob, get two kids in the front room. Have one stationed at the far right gargoyle, and one by the door. Have the one kid pull the gargoyle, change kids and have the other walk through the door.

Playing Tips for...

Stationfall
Infocom

■ To get into the illegal village (essential to play the game) go to level 7 and look in the wastebasket. Get the crumpled form. Go to level 3, and go to the laundry. Press the form in the presser.
■ Don't keep the presser on too long!
■ Now go back to level 5. Go to the commander's bedroom. Look under his bed. Get the validation stamp. Validate illegal village entry form. Go to the entrance to the military module. Put illegal village entry form in slot. Viola!
■ To change the rank on your ID, go to Shady Dan's in the illegal village. Put ID in machine, turn machine on and type "9". You now have access to the ship's armory.
■ To get the explosive needed to blow the lock on the storage bin in level one, get the spacesuit and magnetic boots (don't have your ID!) and go out the airlock with the thermos. Get explosive and put it in the thermos. Then find the series hyperdiode and the detonator.
■ To know what the dots on the wall of the spaceship mean, go to the alien spaceship, type "Script". Lick dots. "unscript". Go to mayor's office in village. Get language book. Get the paper that flutters out.

"Script" read paper. "Unscript" you can now translate the tastes into English!

Playing Tips for...

Lurking Horror
Infocom

■ Once you have the hand, go to the "Alchemy Dept." door. Knock on the door. Wait until the professor leads you in. Drink the Classic Coke. Turn flashlight on. Show professor the note you got on the great dome. Enter the lab. Wait one turn until the professor, is in his own pentagram. Cut pentagram with knife. Exit pentagram. Move lab bench, open the trap door, go down. Open the trap door up, get ring, wear ring, drop knife, get vat. Put the hand in the vat, go down, then north, then up. Get hand, put ring on hand.

Playing Tips for...

King's Quest II
Sierra-On Line

■ To get the second key, get the nightingale out of Hilga's cave. Cover it with the cloth from Neptune's bottle to keep it quiet if Hilga is in the cave. Find the souvenier shop. Enter the shop, give nightingale, buy brass lamp. Polish lamp. You should now have a flying carpet. Ride flying carpet across the mountains. Go right until you find the snake. Rub lamp, rub lamp. Put bridle on snake. Go right until the cave, and the second key! Fly the carpet back across the mountains. Make sure you unlock the second door and read the third.

■ For the third key, go to one of the green lake scenes. Charon is there, and will ferry you across for a price. Give him a treasure, and get on the raft. Once on the island, drink the potion the pegasus gives you. Enter the castle. Go up the spiral staircase, and find the candle. Light the candle with one of the torches on the wall. Enter the basement. Go to the coffin room. If you have garlic, you can scare the vampire away. Look under the pillow. Behold the third key!

■ Open the third door and walk through it. Take the net you see. Go up as far as you can. Cast net. Let fish go. He will give you a lift across the ocean. Once on the island, find the amulet. Find the crystal tower. Walk up the steps (be careful). Find the princess. Kiss the princess. Hold the amulet and think home. You have completed *King's Quest II*.

That ends my Playing Tips for this month. Don't feel frustrated. It took me almost two months to solve all the games above!

I would like to comment on the *Senior Prom 3.0*. It is the best piece of hardware I

have ever seen, and the on-board utilities work wonders. Clay Harrel has made it easier for the masses to deprotect their own software.

Bill Jetzer

Softkey for...

The Dam Busters
Accolade

■ **Requirements**

☐ Super IOB 1.5
☐ Blank disk

Dam Busters is an arcade strategy game that puts you in the pilot seat of a British Lancaster Bomber. You undertake the most daring R.A.F. bombing mission of World War II—the bombing of the great power dams of Eder, the Sorpe, and the Moehne, deep inside Nazi Germany. Along the way, you must fight off ME-110 night fighters, dodge barrage balloons and spotlights.

The protection used is quite simple. Everything is in normal DOS format, with the exception of track $22, which is not formatted. There is only one routine that checks for the protection. It resides on track $13, sector $0B, at byte $34.

All that is required to deprotect the disk is (1) an RTS (like a RETURN in BASIC) be put at the beginning of the protection routine and (2) the *Super IOB* controller variable LT is set to 34 instead of 35.

Type the controller at the end of the article and save it. Install it into *Super IOB* 1.5 by your favorite method and run it. You now have a deprotected *Dam Busters* disk.

Worth Mentioning

The DOS used on *The Dam Busters* is a version of Beagle Brothers' Pronto DOS with a patch that displays the number of free sectors every time you catalog a disk. I like to use this DOS on disks containing large files because it is so fast. The free sector patch is also nice. To use this DOS on your blank disks, boot the *Dam Busters* disk and break out of the HELLO program by pressing ⌂C right away. Insert your disk and initialize it:

To use this DOS on your normal DOS 3.3 disks that already contain files, you can use *Super IOB* to copy tracks $0 - $2 onto your disk, or use the *Copy II Plus* DOS copy option.

Also, you can use *Copy II Plus* or some other program to undelete four files on *Dam Busters*: FREEUP, DISK, MAIN DATA, and DEMO MODE MCP. The last three are useless, but *FREEUP* might be of interest to you. It will display a track by track bit map of free sectors on the last disk that was cataloged when it is run.

Controller

```
1000 REM DAM BUSTERS CONTROLLER
1010 TK = 0: LT = 34: ST = 15: LS = 15: CD = WR:
     FAST = 1
1020 GOSUB 490: GOSUB 610
1030 GOSUB 490:T1 = TK:TK = PEEK (TRK) - 1:
     RESTORE : GOSUB 310:TK = T1: GOSUB 610
1040 IF PEEK (TRK) = LT THEN 1060
1050 TK = PEEK (TRK):ST = PEEK (SCT): GOTO 1020
1060 HOME : PRINT "COPY°DONE" : END
5000 DATA 1°CHANGES
5010 DATA 19,11,52,96
```

Controller Checksums

1000	– $356B	1050	– $9FE4
1010	– $0715	1060	– $280F
1020	– $1DC5	5000	– $321D
1030	– $ED5C	5010	– $771B
1040	– $2D3E		

Tim Strelchun

lower case letters
For Your II Plus

■ **Requirements**

☐ Apple II Plus
☐ DOS 3.3
☐ Micro Test Clips (optional)Radio Shack CAT NO. 270-370
☐ One 13" length No. 22 wire (optional)

With the arrival of Apple Computer's new line of IIe's a couple of years back, came a lot of teeth gnashing from the II Plus owners who mistakenly bought their computers a month or so too early. Many of these people have seen the IIe's new features so commonly seen on today's computers and wished that they could share in some of its luxuries. One dominate new Apple IIe luxury was its ability to both display and enter lower-case characters.

The Apple II Plus is also capable of this eye-easing feature. This article will talk about how to do this, with its main focus directed towards the entering of lower-case characters using the standard II Plus keyboard.

Displaying Lower Case

The process involved in displaying lower-case text on the Apple II Plus is quite simple. On the motherboard in location A5 is a 16K ROM chip that contains the character set definitions. It is from the memory on this chip that the characters on the text screen originate. This chips, socket is compatable with 2716

EPROMs. This is not true however, for the heavily documented F8 ROM chip. All one has to do to change their character set is to replace this 16K ROM chip in location A5 with a 16K EPROM burned with a copy of the new lowercase character set.

Caution: Only do this on Apple II Plus Revision 7 motherboards or newer ones without RAM Configuration Blocks. If one finds this too difficult (i.e. to burn an EPROM with a data image), a kit can be bought that contains a preburned chip and instructions on how to install it. One such kit may be obtained from the manufacturer listed at the end of this article.

If you are further interested in the process of making your own lowercase chips with customized character sets, a colleague and I have developed a working program to edit the character set to your own liking. Write me at my address listed at the end of this article if you are interested in this customize program or have any questions regarding the installation of the EPROM modification above and or displaying lowercase on the Apple II Plus.

Entering Lower Case

The entering of lower-case characters from the keyboard on an Apple II Plus is where the process becomes a little more complicated. On the Apple II Plus lower-case has many compatibility problems. The monitor, Applesoft Basic, and DOS 3.3 do not recognize anything but the upper-case character set. To further complicate things the II Plus keyboard is not designed to enter lower-case. Therefore, a program must be designed to do these chores and several other unmentioned hangups. As mentioned before, the focus of this article is on this program. I have done all the difficult work so that one may use lower-case entry in most programming situations.

Keyboard Changes & Usage

The machine language program I developed does quite a few chores. It starts by relocating the new input routine and the lower-case patch routines to safe places in memory. The program then modifies the F8 ROM by using the ramcard to connect the new input and patch routines. Once this is accomplished, the program returns control to the system and the new changes are connected.

Once connected, any typing on the keyboard will result in lower-case characters being displayed (if that is possible on your computer). To obtain uppercase use the ESC key. It is now a shiftlock. By pressing it once, you may type another key and it will be in upper-case. Afterwards, the next character will be back in lower-case. By pressing ESC twice in a row all the characters you type will be in uppercase. To turn this 'true' shiftlock mode off, press ESC once more. The first

---

escape mode is provided for those users without a shiftkey modification made to their computers. It is quite simple to make and allows this program to recognize when you press [SHIFT]. By this recognition, it can then make a character uppercase. I will explain later how to make the shiftkey modification to your computer if you do not already have one installed.

There have been several key rearrangements, also. When the shift key is pressed and an N, M, or P is also pressed other characters are usually generated. These other characters have been moved to different key positions. A table showing the key setups will follow. In addition, press [⌐E] to use the normal escape editing feature instead of [ESC]. Also, there is an alternate character mode to obtain the moved conflicted characters and some other characters not previously able to be typed. To switch back and forth between the normal mode and the special alternate mode, press [⌐A].

In the alternate character mode your keypresses act normally except for the following:

key	becomes
(	{
)	}
<	[
>	]
slash	backslash
!	|
N	caret
#	checkerboard
-	tilde
P	@

The majority of the changes done to the system are made directly to the copy of the F8 ROM and the Applesoft ROMs in your RAMcard. Changes are also made to the DOS in memory when the program was run. All these changes allow you to type lowercase input now. The commands from the Monitor program (L,M,V, etc.), from DOS 3.3 (CATALOG, LOAD, BRUN, SAVE, etc.), and from Applesoft Basic (LIST, INPUT, READ, PRINT, etc.), all may be entered in lowercase characters. File names in DOS 3.3 may also be saved using the lowercase and alternate characters.

Whenever you are in DOS 3.3 all that must be done to install the Keyboard Enhancer is to BRUN the file. It would be convenient to have it automatically run from a bootup program so that it would be there upon system initialization. If your bootup program is in BASIC just insert a line that contains:

```
PRINT CHR$(4)''BRUN KEYBOARD ENHANCER''
```

The Keyboard Enhancer will then be installed for use. It will return to your program when it is done with its own chores.

Where It Lives

The Keyboard Enhancer when BRUN loads into $0900. After it has finished executing, you must be sure not to overwrite two pages in memory. The input routine resides in page $03 and the patch routines in page $8F. Both of these pages must be preserved.

Entering the Code

The Keyboard Enhancer is not that long. It takes up less than three pages of memory. Enter the monitor (CALL -151) with DOS loaded and type the enhancer hex dump. After you have finished that task, save your work with:

BSAVE KEYBOARD ENHANCER, A$0900, L$220

That is all there is to entering something that has the potential to be very useful.

Shiftkey Mod

To make this modification on your computer, you must have a newer Apple II+ circuit board. A newer II+ has a piggy back board under the "7" through "*" keys. If this is not so on your computer do not perform this mod.

Take your 13" No. 22 gauge wire and strip 1/4" of insulation from each of its ends. Fiddle with one of the Micro Test Clips to see how it works and then solder one end of the the 13" wire to it. Now, turn off the power to your Apple and unplug it.

Looking towards your computer from the front, find the multi-pin connector that attaches the piggy back board to the keyboard. Clip the Micro Test Clip to the second pin from the far right (directly above asterisk key). Now if you have a joystick in your 16 pin game port socket, remove it.

Run the 13" wire to the game port and stick the unused end into the socket in the fourth hole on the right side (looking down with keyboard towards you).
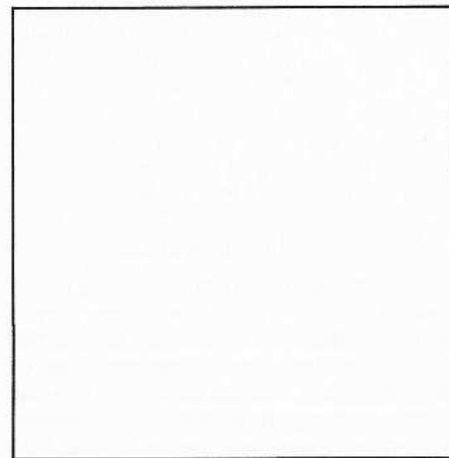
Soldering is not needed on this wire. If you had a joystick previously in the socket, you may put it back in. Don't worry, the wire is small enough to fit in with the joystick.

You may obtain a lower-case EPROM (product name is LCA-2) from:

Dan Paymar
91 Pioneer Place
Durango, CO   81301

My current address is:
Tim Strelchun
7180 SW 189th Ave.
Beaverton,   OR   97007

KEYBOARD ENHANCER

```
0900: 8D 81 C0 A2 00 BD 00 D0      $3063
0908: 9D 00 D0 E8 D0 F7 EE 07      $A7CE
0910: 09 EE 0A 09 D0 EF 8D 83      $F16B
0918: C0 A2 C9 BD F1 09 9D 00      $EF47
0920: 03 CA E0 FF D0 F5 A2 66      $9B09
0928: BD BA 0A 9D 00 8F CA E0      $2512
0930: FF D0 F5 A0 0F A9 EA BE      $FB34
0938: 8B 09 9D 00 FD 88 10 F7      $92C0

0940: A0 0C BE 9B 09 B9 A8 09      $967E
0948: 9D 00 FD 88 10 F4 A9 EA      $787C
0950: 8D B0 FF 8D 0F A1 8D 10      $40A2
0958: A1 A9 20 8D AD FF 8D E8      $A7BE
0960: 9F 8D 6D D5 8D A8 D5 8D      $177A
0968: BE D5 8D E9 D5 8D 0B D6      $FAEE
0970: A2 13 BD B5 09 8D 82 09      $D2F9
0978: BD C9 09 8D 83 09 BD DD      $A31D

0980: 09 8D FF FF CA 10 EB 8D      $EF05
0988: 80 C0 60 14 1B 1C 1D 1E      $C05E
0990: 1F 20 2B 2C 2D 7E 7F 80      $2B07
0998: 81 82 83 11 12 13 21 22      $2D37
09A0: 23 24 25 26 27 28 29 2A      $819F
09A8: 20 4A 8F 48 20 00 03 68      $E322
09B0: 91 28 AD AD 03 AE AF E9      $F3C5
09B8: EA 0D 0E 6E A9 EA 6F AA      $295A

09C0: EB BF C0 0C 0D A3 30 A4      $4698
09C8: 31 FF FF 9F 9F A1 A1 D5      $A000
09D0: D5 D5 D5 D5 D5 D5 D5 D6      $FBF9
09D8: D6 FB FD FB FD 00 8F 07      $82E7
09E0: 8F 15 8F 2C 2C 2C 8F 8F      $F5ED
09E8: 8F 27 8F 3A 8F 60 60 8F      $50C5
09F0: 8F E6 4E D0 02 E6 4F AD      $F9CF
09F8: 63 C0 8D AB 03 AD 00 C0      $BBAF

0A00: 10 EF 8D 10 C0 C9 9B F0      $C81E
0A08: 0B C9 81 F0 1D C9 85 F0      $5D5A
0A10: 2A 4C 4C 03 AD AA 03 F0      $00E2
0A18: 0B C9 01 F0 07 A9 00 8D      $09C5
0A20: AA 03 F0 03 EE AA 03 4C      $3AD7
0A28: 00 03 AD AC 03 D0 04 A9      $4D96
0A30: 01 D0 02 A9 00 8D AC 03      $09D0
0A38: 4C 00 03 A9 9B 8E AE 03      $C0B2

0A40: 8D AD 03 AD AB 03 10 05      $C43E
0A48: AD AA 03 F0 15 AD AD 03      $8190
```

```
0A50: A2 02 DD AF 03 F0 05 CA        $384E
0A58: 10 F8 30 0F BD B2 03 4C        $1E2C
0A60: 7A 03 AD AD 03 C9 C0 90        $A773
0A68: 02 09 20 8D AD 03 AD AC        $15BA
0A70: 03 F0 15 AD AD 03 A2 09        $D200
0A78: DD B5 03 F0 05 CA 10 F8        $F4AE

0A80: 30 06 BD BF 03 8D AD 03        $DC72
0A88: AD AA 03 C9 02 F0 05 A9        $F08C
0A90: 00 8D AA 03 AD AD 03 AE        $A932
0A98: AE 03 60 00 00 00 00 00        $7F81
0AA0: DE DD C0 CE CD D0 A8 A9        $F4B7
0AA8: BC BE AF A1 CE A3 AD D0        $20C1
0AB0: FB FD DB DD DC FC DE FF        $A07A
0AB8: FE C0 B9 00 02 C8 4C 0A        $D022

0AC0: 8F 20 A4 A1 C9 E1 90 06        $B7BB
0AC8: C9 FB B0 02 29 DF 60 20        $A974
0AD0: 0A 8F A2 0A DD 40 A9 F0        $DE26
0AD8: 03 CA D0 F8 68 68 4C 18        $DDFA
0AE0: A1 BD 01 02 10 11 A5 0E        $37DF
0AE8: F0 16 C9 22 F0 12 A5 13        $204D
0AF0: C9 49 F0 0C BD 00 02 08        $A2B7
0AF8: C9 61 90 02 29 5F 28 60        $C8ED

0B00: BD 00 02 60 C9 C0 90 0E        $9DAC
0B08: C9 E1 90 06 C9 FB B0 02        $3D5F
0B10: 29 DF 38 E9 80 60 38 E9        $CC2B
0B18: 40 60 20 0C FD 4C 0A 8F        $B033
```

J. Aufderheide

■ **Requirements**

☐ FID
☐ blank disk

Here are some softkeys on some great children's software:

Softkey for...

Create with Garfield
DLM

DLM's *Create With Garfield* has been on the Most Wanted List for quite some time. The copy I have is 2 years old and may be different than the one currently on the market. A search through past COMPUTISTs shows that DLM is fond of changing epilogs. This one is no different. The disk seems only to have altered address and data epilogs of AADE. So, to deprotect it, do the following:

1 Initialize a disk with a fast DOS (ProntoDOS is great) and delete the hello file.
INIT GARFIELD
DELETE GARFIELD

2 Kill some epilog and checksum errors:
CALL -151
B942:18
B925:18 60
B988:18 60

3 Get out your System Master.
BRUN FID

Put your *Create With Garfield* disk in drive 1, and your newly initialized disk in drive 2. Copy all files using the wildcard (=). When *FID* asks you for a filename, press =. It will ask if you want prompting. Press N and the program will copy all the files from the *Create With Garfield* disk to your copy.

4 Quit out of *FID*. Put your copy in drive 1 and load the file *GARFIELD*.
LOAD GARFIELD
LIST

Ah ha! This little program has some unneeded code! The program should look like this:

```
10 ONERR GOTO 100
20 POKE 1012,0
30 HGR2
35 PRINT ""MAXFILES 1''
40 PRINT ""BLOAD START''
50 PRINT ""BRUN MENU''
100 CALL -1438
```

Take out lines 10, 20, and 100, and save the program, it should work just fine.
10
20
100
SAVE GARFIELD
You are done!

Softkey for...

Peanut's Maze Marathon
Random House

Softkey for...

Snoopy's Reading Machine
Random House

These two can be defeated in almost the same way as *Garfield*. They have address epilogs of AA DE.

1 Initialize a disk with a fast DOS (ProntoDOS is great) and delete the *HELLO* file.
INIT HELLO
DELETE HELLO

2 Kill some epilog and checksum errors and run *FID* from your DOS 3.3 System Master.
CALL -151
B942:18
B988:18 60
BRUN FID

3 Put your Random House disk in drive 1, and your newly initialized disk in drive 2.

Copy all files using the wildcard (=) character for the filename.

Put your originals away, and use your programs without fear. Has anyone ever listed all the confirmed ways that original disks crash?

Stanley Planton

Softkey for...

Practical Grammar
Intellectual Software

One of the media personnel from a local school system brought me a box of 18 English grammar disks from Intellectual Software and asked my help in making backups of the disks. Unfortunately, some of the disks had already been damaged by students, as seems inevitable when schools spend lots of money on protected software, then put it into the hands of the kids. I was able, however, to come up with a softkey that seems to work with the rest.

The people at Intellectual were certainly busy with their copy-protection schemes! The DOS is a slightly-modified DOS 3.3, with changed address epilog bytes in tracks $00—$02, the first byte of the address header was changed to B7 on most of the rest of the disk, there were checksum errors, either deliberate or accidental on most tracks, and there is at least one track of "garbage", usually track $13, that consists mostly of FF bytes. All of this meant that using *COPYA* or *SUPER IOB* might require a fair amount of work, so I decided to *FID* the files over to a standard disk, and to write out the instructions so that the school media personnel could do most of the work themselves.

1 Boot your DOS 3.3 System Master disk. Place a blank disk in the drive and initialize it with the name *BTUT*.
INIT BTUT
DELETE BTUT

2 Tell DOS to ignore the first byte of the address header and address checksum errors and use *FID* (from System Master disk) to copy all the files.

POKE 47444,41
POKE 47445,0
POKE 47498,0
BRUN FID

3 Use the wildcard (=) for the file names, answer N on the prompting question.

Assembly Line Softkeying

That's it. Most or all of the disks in the series, those that have not already been trashed by the kids, should now be in DOS 3.3 format, unprotected.

If you have a number of these disks to work on, you will probably want to make an "assembly line": when you get the first disk INITed with the *BTUT* file, use a fast copier, such as *Locksmith* Fast Copy to copy this disk to enough blanks for the rest of the series. You can then get *FID* running as above, and switch originals in and out of drive #1 and the copies of the INITed disk into drive #2 as *FID* finishes the preceding disk.

Softkey for...

Kalamazoo Teacher's Record Book 2.0
Hartley Courseware Inc.

■ **Requirements**

□ Apple II, II Plus, IIe, IIc
□ Blank disks
□ *Super IOB 1.5*
□ *Copy II Plus* or equivalent sector-editing and file-moving utilities
□ *Dos-Up* or equivalent DOS mover
□ System Master

A friend, who is a teacher, brought me his new gradebook program and asked me how to make a backup. As I have successfully copied Hartley programs before with Dave Stanton's softkey from COMPUTIST #32, I dug out *Super IOB 1.5*, keyed in the suggested controller, and let *Super IOB* "do its thing" to the Hartley Master Disk.

When I put a standard DOS on the copy, however, the program would load partially, then crash. Hmmm, it appeared that the gang at Hartley had done something a little different, so it was time to investigate further.

Reasoning that there might be something in their DOS, in addition to a change of epilogs from DE AA to a DA AA pattern that might be causing problems.

The next step was to change the IOB controller to pick up every track, not just those from $03 on, so that their DOS could be examined.

Using the controller below, and looking at the copy with the track mapping and sector editing utilities of *COPY II Plus*, it became apparent that Hartley was using a version of *Diversi-DOS*, or another DOS that let track $02 be used by files, since every available sector, including those of track $02, was in use by some file or another.

This was the cause of the failure of my first attempt to unprotect the disk: copying a standard DOS to the softkeyed disk stepped all over track $02, resulting in a massive loss of file data.

Remembering that there had been another COMPUTIST that dealt with adding space to a disk for large files, I dug out COMPUTIST #27, and looked at Eric Ondler's alternate

softkey for *Microzines*. Yes, it might be possible to create a disk with a slightly modified DOS 3.3 that would allow file storage in tracks $02 and $11 (just to be safe) to replace Harley's track $02, which would be lost when DOS was put onto the softkeyed disk. Since a lot of space would not be required, it would not be necessary to add an extra track, so only some of Mr. Ondler's procedures would be needed.

I INIT'ed a fresh disk, then used *COPY II Plus'* sector-editor to make some changes to tracks $02 and $11, based on the edits in COMPUTIST #27:

Trk	Sct	Byte(s)	From	To
$11	$00	$40	$00	$FF
$11	$00	$41	$00	$E0
$11	$00	$7D	$00	$3E
$11	$06	$01	$11	$00
$11	$06	$02	$05	$00
$02	$01	$92	$69	$A9

These edits should make enough room for the *Record Book Master Disk* files. I then booted the DOS disk that I had previously INITed and modified in the last step, put it in drive #2, put my System Master in drive #1, BRAN FID, removed the System Master and put the copy of the Harley disk that I had made earlier with *Super IOB 1.5* into drive #1. All these steps might not be strictly necessary, but I wanted to be absolutely certain that FID would be operating under the rules created by the edit, and not under standard DOS 3.3. I then chose the wildcard = and no-prompting options, and copied the files from the IOB'ed disk to the INIT'ed and modified disk. They all fit, so I proudly booted the disk and watched developments. Needless to say, the copy crashed, in a spectacular fashion that seemed to load an APPLESOFT program into a graphics screen, flashing a scrambled BASIC program on the monitor.

What now? An examination of Hartley's HELLO program and MAIN program turned up a reference to a binary file, *DDMOVER*, which was BRUN early in the file loading process. *DDMOVER* seemed to move *Diversi-Dos* up in memory, for reasons clear only to the folks at Hartley, and which was causing programs under standard DOS 3.3 to load into graphics screens... The new task before me was to somehow get rid of *DDMOVER* while preserving its function of getting DOS out of the way of the files. I had a *Wildcard* routine available, *DOS-UP*, that is a binary file that can be BRUN to move DOS out of the way of a *Wildcard* load, so I thought "Why not try *DOS-UP* instead?" I used *COPY II Plus* to DELETE *DDMOVER*, to COPY *DOS-UP* from a *Wildcard* utility disk to my semi-softkeyed *Grade Book 2.0* disk, and to RENAME *DOS-UP* as *DDMOVER*. That way, any references to *DDMOVER* would get *DOS-UP* instead.

The next test boot of the softkeyed disk

provided the message "DOS NOW IN HIGH MEMORY" as *DOS-UP* was BRUN under its new alias, and the disk continued to load properly. Data disks could be formatted with the *Record Book 2.0* softkeyed disk, and records stored onto them, so the program was finally in an unprotected DOS 3.3 *COPYA* format!

The Supplementary Master disk in the package can be softkeyed with the *Super IOB* controller below without other changes. Data disks seem to be copyable with *COPYA*.

Controller

```
1000 REM HARTLEY GRADEBOOK
1010 TK = 0 : LT = 35 : ST = 15 : LS = 15 : CD = WR : FAST
     = 1
1020 RESTORE : GOSUB 170 : GOSUB 490 : GOSUB 610
1030 GOSUB 230 : GOSUB 490 : GOSUB 610 : IF PEEK
     (TRK) = LT THEN 1050
1040 TK = PEEK (TRK) : ST = PEEK (SCT) : GOTO 1020
1050 HOME : PRINT "ALL DONE" : END
5000 DATA 218,170,218,170
```

Controller Checksums

1000	– $356B	1040	– $D41C
1010	– $2544	1050	– $F08B
1020	– $73B0	5000	– $96FB
1030	– $4629		

Step-By-Step

1 Boot a System Master and INIT a blank disk.

HELLO
DELETE HELLO

2 Enter the above controller into *SUPER IOB 1.5* and copy the Master Disk.

3 Copy the Supplementary Master disk with *Super IOB* and the same controller.

Put the Supplementary Master original and copy away, so you don't get them confused with the Master disk.

4 Use a sector editor to make the edits to tracks $02 and $11 of the disk you INITed in Step 1.

5 Copy the files from the IOB copy of the Master Disk that was made in Step 2 to the edited disk from Step 4. Don't bother copying the *DDMOVER* file, or DELETE it if you did copy it.

6 Copy *DOS-UP* to the disk you are working on, and change its name to *DDMOVER*. If you have some other DOS-moving utility instead of *DOS-UP*, it might work as well.

7 Boot the disk. All going well, you should get a perfectly functioning *COPYA*-able version of *Teacher's Record Book 2.0*, as well as a *COPYA*-able Supplementary Master.

Gary Rohr

Franklin ACE 2xxx Users

As the owner of a Franklin personal computer, I am always finding software and/or hardware which works super on an Apple personal computer, but refuses to work on my Ace.

This is unfortunate for me because I cannot obtain the benefits of many valuable products; and for vendors, because many will never receive a single dollar from me.

Add to this growing list of products: *Senior Prom*.

† Softkey for...

Math Blaster

Davidson & Associates

I have a pet peeve regarding software which works on Apple computers, but not on my Franklin Ace 2200. This is another one of those programs. My goal is to try to find out why this program runs fine all the way through the sign on screens and demo screens but refuses to run the actual program.

While I have not, as yet, figured out why this program will not run, I have managed to remove the protection so that I can continue my research.

I noticed that this program was on the "Most Wanted" list, and thought I'd share what I know so far — maybe someone will be able to let me know how to get it to run on my Franklin.

The Protection

The first step in determining why a program won't run, is to be able to have a copy to experiment with.

First step - make a copy - or so I thought. My copies made with *COPY II Plus* or *Locksmith* failed to boot. The screen fills with C's - I get a beep and a re-boot. Using a nibble editor, I looked at various tracks and found normal ADDRESS and DATA prologs; however, the epilogs had been altered to FFs.

I initialized a diskette with a *HELLO* program, deleted it, then set up a *Super IOB* controller to copy the original tracks $3—$22 changing the strange epilogues to normal ones.

I had some problems with read-errors during the copy, so I poked DOS in a few places to ignore checksums and read errors.

This allowed the copy to run successfully.

Since the original program disk would display the DOS prompt briefly, I thought I would find a catalog; however, the CATALOG command just displayed trash. I tried to boot the disk — it wouldn't work.

Next, I tried to capture the RWTS for use in a swap controller and replaced the protected DOS with a normal DOS. This wouldn't work because their DOS was so heavily modified. Within the write routine they jump to another routine outside of that which is normally used for the RWTS ($B800—BFFF) and *Super IOB* comes to an abrupt stop.

After looking through my stack of COMPUTIST issues, I found a softkey for another Davidson & Associates program. I read through it to see what protection they used on that program. The one thing that caught my eye, was the fact that they altered the translate tables.

I grabbed my trusty *Beneath Apple DOS* to find the locations of the translate table and compared the *Math Blaster!* version to a normal DOS. Sure enough, there was one byte altered in both the read and write translate tables. I altered my first IOB controller to swap the proper/improper translate bytes and successfully copied the protected tracks $3—22.

I was now able to do a CATALOG and load programs from the disk - one was even named *HELLO*. Great! I booted the de-protected disk and got an error message - I believe it was the infamous "PROGRAM TOO LARGE". After more attempts at trying to get the program to run, I decided that their DOS was required.

I modified my *Super IOB* controller to copy all tracks and then searched the DOS tracks for the FF FF epilogue bytes and replaced them with normal DE AA bytes.

Still, the diskette failed to boot successfully. By comparing the *Math Blaster!* DOS with a normal DOS I found the problem to be a jump on track $00 sector $00.

Using *Diskedit* you can read a normal DOS sector and write it to the unprotected *Math Blaster!* diskette, or you can change the 6C FE BB at offset $4A to 6C FD 08.

Softkey #1

1 Add the following controller to *Super IOB* and when prompted, specify yes for the format with volume number 254.

Controller

```
1000 REM MATH BLASTER CONTROLLER
1010 TK=0 : ST=0 : LT=35 : CD=WR
1020 T1=TK : GOSUB 490 : RESTORE : GOSUB 260 :
     POKE 47426,24 : GOSUB 190 : GOSUB 210 :
     GOSUB 170
1025 POKE 47786,0 : POKE 47166,171 : POKE
     48224,254
1030 GOSUB 430 : GOSUB 100 : ST=ST+1 : IF ST<DOS
     THEN 1030
1040 IF BF THEN 1060
```

```
1050 ST=0 : TK=TK+1 : IF TK<LT THEN 1030
1060 GOSUB 2330 : GOSUB 490 : TK=T1 : ST=0
1065 POKE 47786,170 : POKE 47166,255 : POKE
     48224,255
1070 GOSUB 430 : GOSUB 100 : ST=ST+1 : IF ST<DOS
     THEN 1070
1080 ST=0 : TK=TK+1 : IF BF=0 AND TK<LT THEN
     1070
1090 IF TK<LT THEN 1020
1100 HOME : PRINT "DONE WITH MATH BLASTER COPY"
     : END
5000 DATA213,170,150
5010 DATA213,170,173
5020 DATA255,255,255,255
```

2 Using *Diskedit* change the following bytes:

Trk	Sct	Byte(s)	From	To
$00	$00	$4B-4C	$FE BB	$FD 08
$00	$02	$9E	$FF	$DE
		$A3	$FF	$AA
		$A8	$FF	$EB
$00	$03	$35	$FF	$DE
		$3F	$FF	$AA
		$91	$FF	$DE
		$9B	$FF	$AA
$00	$04	$29	$AA	$96
		$AA	$00	$AA

Softkey #2

(Optional - either should work)

1 Use DOS 3.3 to initialize a disk with *HELLO* then delete *HELLO*.

INIT HELLO
DELETE HELLO

2 Use the controller with *Super IOB* except change line 1010 in the controller so the starting track number is $03. (TK = 3 instead of TK = 0.)

3 Run *Super IOB* copying the original *Math Blaster!* diskette to the diskette initialized in step #1 (Specify NO to format question).

4 After the copy is complete, change the *Super IOB* controller to copy just DOS from the original diskette (TK = 0 : LT = 3). Leave all other values in line 1010 the same

5 Using a new output diskette, run *Super IOB* to copy the *Math Blaster!* DOS (Specify YES to the format question - specify volume # 254)

6 You should now have two partially deprotected *Math Blaster!* diskettes, one which contains the programs, and another which contains the DOS.

7 Using *Diskedit*, read the following sectors from the second diskette (the *Math Blaster!* DOS) and write them back out to the first diskette which you initialized with a normal DOS.

Track	Sectors
$00	$01-$02
$00	$06
$00	$0A-$0B
$01	$02
$01	$04
$01	$07
$01	$09
$02	$02-$0F

8 Check track $00, sector $02 for the following pattern of bytes, which represent the epilog byte checks.

A9 FF 20 B8 B8 A9 FF 20 B8 B8 A9 FF B8 B8

...and change to:

A9 DE 20 B8 B8 A9 AA 20 B8 B8 A9 EB B8 B8

Note: I found the pattern starting at byte $9D

9 Check track $00, sector $04 at approximately byte $09 for:

B3 B3 A0 E0 B3 C3 C5 B3 A0 E0 B3 C3 C5 B3 A0 E0 B3 B3
C5 AA A0 82 B3 B3 C5 AA A0 82 C5 B3 B3 AA 88 82 C5 B3
B3 AA 88 82 C5

...and change to:

A9 AB 8D 4B B8 A9 FF 20 DF BC 20 EC BC A9 AA 8D 29 BA
60 60 A9 FF 8D 4B B8 A9 DE 20 DF BC A9 AA 20 EC BC A9
96 8D 29 BA 60

10 This should complete the deprotection — the completely deprotected version is now on diskette #1. (The second diskette is no longer required)

11 Note: The data diskette is not protected and can be copied using any "normal" copy utility (i.e., *Super IOB* with a FAST.CON controller)

Softkey for...

A2-PB1 Pinball

?

Why another softkey for A2-PB1 Pinball?

As the owner of a Franklin personal computer, I sometimes feel as though I'm a social outcast. I bought my Franklin because, like many people, I thought it was an Apple compatible. Well, maybe it is. I do have some Apple software which runs perfect. On the other hand, I have an almost equal amount of software for the Apple which either doesn't work at all, or doesn't work entirely correct.

I work in a technical support department, supporting IBM mainframes and debugging system software. Occasionally, I've had to rely on assistance from a vendor's technical support department to supply fixes for their products.

Imagine my surprise when I first called a software vendor's technical support department to find out what I needed to do to get their software to run on my Franklin, and heard, "It doesn't run on a Franklin," - I know that! Why

do you think I called technical support?? How do I get it to run on a Franklin? "It doesn't run on a Franklin!!" — Is someone working on this problem? Can I expect help soon?? "It doesn't run on a Franklin!!!!"

Although unspoken, I could sense the voice at the other end wanting to say, "And I don't care - and I hope it never does". After dealing with a few vendors, I decide - fine - if they want to restrict their market share to only Apples and leave out the "so-called" compatibles - to heck with them!

I decided that before I tried obtaining any more software that I'd find out first which software would run on my Franklin. Who should know? Well, Franklin Computer Corporation - of course. So, I sat down and wrote a nice letter explaining the problem I was having and asked if they could supply me with a list of software which would run on my Franklin - or a list of which software I should stay away from.

Several weeks later, I received notification that they did indeed have a list of compatible software, and it's updated monthly. But would they send me one?? Of course not - "See your local dealer" was how I believe the letter was phrased.

Of course the dealer doesn't know which software is compatible - they say to consult a software retailer - and the ones that deal in Apple software won't hardly admit that they even know what a Franklin is...!

Finally I decided - this is ridiculous - I've written software - I've fixed software - I'll try to fix it myself. That's when I discovered that the world of PCs is not really another world at all - more like "welcome to the twilight zone."

Protected software?? I'm not leasing it - I bought it!! I don't want to sell it - I just want to be able to use it!!!

Then I discovered COMPUTIST, what a great magazine! However, I was to find that the problems were to continue. While attempting to use the published softkey to de-protect *Choplifter*, so I could try and get it to run on my Franklin, I discovered that I couldn't use the method presented (i.e., move the ROM boot code to RAM - modify it - and initiate the boot from the modified RAM code).

Obviously, something's different. I figured it was going to take awhile to try and determine how to do this, and that if I was going to know whether or not the process was working - I'd better try using a game that normally runs on my PC. So, I looked for a game which I had, and for which there was a softkey that used the ROM to RAM method of de-protection — *A2-PB1 Pinball*.

After several days of trying every possible combination of soft-switches I could think of, I finally gave up on this method of de-protecting the *A2-PB1 Pinball*. I was about to give up altogether when I started wondering how many

people tried, and couldn't de-protect this game with the published softkey. Then I decided to come up with an alternate method for us poor unfortunates.

The Protection

Using my own method of boot code tracing, which involves modifying the readable sector(s) to jump to the monitor after loading new sectors so I can analyze what has been loaded, I discovered that the first thing the boot process does is to copy page 8 of memory ($800-8FF) to page 3 ($300-3FF) and then jumps to the page 3 code.

I'm relatively new to 6502 machine language, so I may be mistaken, but it appeared as though bytes were being read from disk - checking bytes against a table - then storing selected bytes into an area of memory normally occupied by the RWTS.

I decided that this was going to be a pain. There had to be an easier way, especially since someone had already researched this to develop the original softkey. So, I went back to the softkey published in COMPUTIST #22, by Mr. Leathlean. From here, I learned that the program is loaded into memory at $900—9100, as well as, $300—3FF and $800—8FF. The trick was going to be in determining when the data in memory was valid and could be captured.

After timing different phases of the boot process (a full boot requires 22 seconds on my Franklin), I tried capturing pages of memory at different times and then combining them into a single BRUNable module (as was described in the original softkey) and attempting to execute it.

The Softkey

Initialize a new diskette:

NEW
INIT HELLO
DELETE HELLO

Insert original (protected) copy in boot drive (make sure it is write-protected) and initiate the boot process. Carefully time the boot process and eighteen seconds into the boot interrupt it with your favorite method.

I have found on my Franklin, that holding down [RESET] flipping open the drive door - releasing the keys - then pressing [RESET] once again as soon as an attempt is made to read the disk is usually enough to interrupt the boot process.

Now enter the monitor and display part of page 8 memory.

CALL -151
850.85F

You should see:

```
0850= 0D 4C 13 08 00 00 00 00
0858= 00 00 00 14 00 00 00 00
```

If not, restart the boot and try altering the elapsed time allowed before interrupting the boot.

Move page 8 to a "safe" memory location:

9200<800.8FFM

Warm boot the original diskette:

C600G

Wait until boot completes, and the game screen is displayed. Then interrupt the game with any method you prefer (i.e., `RESET` process described above)
Enter the monitor:

CALL -151

Move page 3 to a "safe" memory location:

9100<300.3FFM

Insert your newly initialized diskette into the boot drive and perform a warm boot:

C600G

When DOS prompt (Applesoft) appears, enter the monitor:

CALL-151

Move page 8 back to it's original memory location:

800<9200.92FFM

And make the following edits:

080D:20 DA 91
91DE:91
91E6:60
A964:FF

And exit the monitor:

`C`

Save *A2-PB1 Pinball* to diskette:

BSAVE A2-PB1 PINBALL,A$80D,L$89F3

Cold boot the new disk to clear memory (i.e., turn power off and back on) - then load your new "un-protected" version of A2-PB1 Pinball into memory:

BLOAD A2-PB1 PINBALL

Enter the monitor:

CALL -151

And move page 8 to a "safe" area of memory:

9200<800.8FFM

Insert the original (protected) diskette into the boot drive - and perform a warm boot:

C600G

Carefully time the boot process - and interrupt the boot after 20 seconds into the process using your favorite method.

Then enter the monitor:

CALL -151

Insert the new unprotected diskette into the boot drive and initiate a warm boot:

C600G

Enter the monitor:

CALL -151

Move page 8 to it's original memory location:

800<9200.92FFM

Alter DOS to allow a save of greater than 32K:

A964:FF

Exit the monitor:

`C`

And save your completely un-protected version of A2-PB1 Pinball to diskette:

BSAVE A2-PB1 PINBALL,A$80D,L$89F3

In Summary

While my Franklin PC does have many disadvantages - it does have it's advantages as well, such as the ability to format diskettes with 40 tracks, as well as, the standard 34 track format. The speed of the Franklin FDOS 5.0 is also a plus as can be seen with the following timings:

Boot time

Original protected version 22 seconds
Unprotected version using DOS 3.3 37 seconds
Unprotected version using FDOS 5.0 8 seconds

Thomas Purifoy

A.P.T. for...

Might and Magic
New World Computing Inc.

Carole Fox's APT for *Might and Magic* in COMPUTIST #51 left out a few bits of information that are rather important. Since I don't own a IIgs, I was not able to try out her method of changing decimal to hex in BASIC.

However, I did notice that on my version of *Might and Magic*, experience, gold, hit points, gems, and spell points are written with a special formula onto the disk.

They are calculated together using the numbers assigned to certain bits (called nibbles). Rob Hall pointed out in COMPUTIST #40 that *Wizardry* uses a similar type formula to store experience points.

Might & Magic sometimes writes two bytes onto disk that are for exactly the same thing, the byte is just repeated. Therefore, just put the same byte twice, where it is neccessary (all attributes, experience level, spell level, and hit points use this "double writing").

Experience: (use Carole Fox's table to match up bytes)

byte	nibble	place	
$27	1st	16's	(161)
	2nd	1's	(160)
$28	1st	4,096's	(163)
	2nd	256's	(162)
$29	1st	1,048,576's	(165)
	2nd	65,536's	(164)
$2A	1st	268,440,000's	(167)
	2nd	16,777,216's	(166)

i.e. You have 1,694,834 experience points, thus:

```
byte $27 = $72 (7x16)(2x1)
where 7x16 = 112   and 2x1 = 2
so... 13x40 96 = 53,248

byte $28 = DC (13x40 96)(12x256)
where 12x256 = 30 72 and 1x10 48576 = 10 48576
so... 9x65,536 = 589,824

byte $29 = 19 (1x10 48576)(9x65536)
so... total = 1,694,834
```

For those of you who are not math whizzes try entering $0F in the right-hand most position. Since the value are actually stored in reverse nibble order, a small encoded number at the right becomes a large decoded number.
............................*RDEXed*

Gold and Gems are stored in exactly the same way as experience, except they only use three and two bytes, respectively. But, they only use nibbles up to 4,096 and 1,048,576.

Spell Points and Hit Points are a type of mixture. They use the special formula and the double writing. Therefore, if you have 1,034 hit points, the bytes you see would look like this: (bytes $33-38).

```
hex                         text
30- 00 49 00 0A 04 0A 04 0A  @ I@+@+@+
38- 04 D8 94 00 0D 11 12 00  @X.@MQQ@
```

I hope that this clears up any misunderstandings that anyone might have on character editing *Might and Magic*. Go out there and find out what's really "Behind VARN", literally.

(?) *Lords of Conquest*, by Electronic Arts, is a wonderful RISK take-off and is very challenging and fun. But, like most Electronic Arts games, it's protected.

I copied the disk with *Copy II Plus* 5.1 fast disk copy and got the usual read error on track 6. However, when I searched the disk for the tell-tale nibble count strings (4C 69 A0, 20 03 A0, etc.) I didn't find anything.

I cataloged the copy and received a list of all the files. *Copy II Plus* said that the file "LOC" was the HELLO file and I started to investigate.

I won't go into details, but it just got complicated from there. There seems to be four different times that the protection routine is accessed: one at boot-up, one at the title screen, one when production is over, and one when development starts.

If anyone can give me any help on deprotecting this, I would really appreciate it.

Brian Troha

A request

For those people who submit IIgs softkeys, please include some disassembly around the edit you made.

Also, if possible please give a brief description of why the edit works (why you did what you did). Why? Because IIgs programs are being constantly upgraded and the protection routines may move around.

A quick note

You should compare the codes listed in the softkeys for *Destroyer* by Epyx, *Top Draw*, by Styleware, and the *Graphic Studio*, by Accolade.

I have seen four different versions of this same protection routine, implemented in four different ways, by the forementioned companies and Electronic Arts.

You can usually find the disk check by searching for A2 20 A0 00 and A2 21 A0 01.

I found the string on five different programs by three different companies. This leads me to believe that they are using a commercial protection scheme.

You might want to keep this in mind when trying to softkey other programs that copy, but give no read errors. Although the methods used to compare the checksums range in technique, all the values are within one of each in the four different versions I have encountered. The values are:

For track $20, between 1F40 and 2008 (2009 for Styleware).

For track $21, between 1E78 and 1DB0 (1E79 for Styleware).

I hope someone may use this information in the future.

Softkey for...

Graphics Studio
Accolade

■ **Requirements**

☐ 3½"disk-copier
☐ 3½"disk-editor
☐ *Copy II Plus v8.1*

The *Graphics Studio* is another IIgs program which uses the new protection routine. The protection seems to be a nibble-count on tracks $20 and $21 of side one of the 3½"disk (again!).

I searched the disk for A2 20 A0 01 and found it on block $98 starting with byte $1A8. Here is what the code looks like (on the disk):

```
1A5:  08              PHP
1A6:  E2 30           SEP #30
1A8:  A2 20           LDX #20
1AA:  A0 01           LDY #01
1AC:  8B              PHB
1AD:  4B              PHK
1AE:  AB              PLB
1AF:  20 9D D1        JSR D19D
1B2:  AB              PLB
1B3:  B0 23           BCS 1D8 (+23)
1B5:  8A              TXA
1B6:  8F 33 00 00     STA 000033
1BA:  98              TYA
1BB:  8F 34 00 00     STA 000034
1BF:  A2 21           LDX #21
1C1:  A0 01           LDY #01
1C3:  8B              PHB
1C4:  4B              PHK
1C5:  AB              PLB
1C6:  20 9D D1        JSR D19D
1C9:  AB              PLB
1CA:  B0 0C           BCS 1D8 (+0C)
1CC:  8A              TXA
1CD:  8F 35 00 00     STA 000035
1D1:  98              TYA
1D2:  8F 36 00 00     STA 000036
1D6:  A9 00           LDA #00
1D8:  8F 02 00 00     STA 000002
1DC:  8F 03 00 00     STA 000003
1E0:  28              PLP
1E1:  6B              RTL
```

Working my way backwards, I found the call and the checksum comparisons are on block $31 and looks like this (on the disk):

```
4E:  AD 33 00        LDA 0033
51:  38              SEC
52:  E9 40 1F        SBC #1F40
55:  70 03           BVS 5A (+03)
57:  49 00 80        EOR #8000
5A:  10 34           BPL 90 (+34)
5C:  AD 33 00        LDA 0033
5F:  38              SEC
60:  E9 08 20        SBC #2008
63:  F0 07           BEQ 6C (+07)
65:  70 03           BVS 6A (+03)
67:  49 00 80        EOR #8000
6A:  30 34           BMI 90 (+34)
6C:  AD 35 00        LDA 0035
6F:  38              SEC
70:  E9 B0 1D        SBC #1DB0
73:  70 03           BVS 78 (+03)
75:  49 00 80        EOR #8000
78:  10 16           BPL 90 (+16)
7A:  AD 35 00        LDA 0035
7D:  38              SEC
7E:  E9 78 1E        SBC #1E78
81:  D0 03           BNE 86 (+03)
83:  82 61 00        BRL E7 (+61)
86:  70 03           BVS 8B (+03)
88:  49 00 80        EOR #8000
8B:  30 03           BMI 90 (+03)
8D:  82 57 00        BRL E7 (+57)
```

Checking the code at E7:

```
E7:  A9 01 00        LDA #0001
EA:  85 F0           STA F0
EC:  82 25 FF        BRL 14 (-DB)
```

Now back up to 14:

```
14:  AD 98 00        LDA 0098
17:  D0 03           BNE 1C (+03)
19:  82 D3 00        BRL EF (+D3) SKIP OVER CHECK
1C:  A5 F0           LDA F0
1E:  F0 03           BEQ 22 (+03)
20:  82 CC 00        BRL EF (+CC) ;SKIP OVER CHECK
```

When looking at the the above code, the important instructions are LDA #0001 and STA F0. Then the code branches up to 14 and checks to see if F0 holds anything but zero. From 14 on you can see that 0098 should hold a value of 0000 and (as I said before) F0 should hold a value of 0001. With that in mind I came up with the following patch:

```
14:  9C 98 00        STZ 0098      STore 0 in 0098
17:  EA              NOP
18:  EA              NOP
19:  A9 01 00        LDA #0001
1C:  85 F0           STA F0
1E:  EA              NOP
1F:  EA              NOP
20:  82 CC 00        BRL           Skip Check
```

As you can see the above code stores the right values in the right places and then branches past the disk check. You now have an unprotected backup of the *Graphics Studio* that can be moved to a hard disk.

Step by step would be:

[1] Copy the 3½"disk

[2] Make the following edits (on the copy):

Block	Byte(s)	From	To
$31	$14	AD 98 00 D0 03	9C 98 00 EA EA
		82 D3 00 A5 F0	A9 01 00 85 F0
		F0 03	EA EA

[3] Write the block back to the copy.

Softkey for...

Marble Madness
Electronic Arts

■ **Requirements**

☐ 512K Apple IIgs
☐ 3½"disk-copier
☐ 3½"disk-editor
☐ *Copy II Plus v8.1*

The IIgs version of *Marble Madness* has improved sound and graphics, plus you can use the mouse or play with two players. When making a copy, I received no read errors. So I assume this is one of the many programs that is being protected by using on nibble count on tracks $20 and $21 of side one.

The game runs under ProDOS 8 and uses *DOS8.SYSTEM* as the *HELLO* program. I loaded *DOS8.SYSTEM* into memory and started checking things out.

After much looking I came across a JSR to $37ØØ and a BCS (Branch Carry Set) at $2ØF3, so I changed the JSR to NOP NOP SEC and started the file with **2000G**. I was then able to play the game.

The only thing left was to find 20 00 37 on the disk and make the change permanent.

Step-By-Step

| 1 | Copy the 3½"disk |

| 2 | Make the following edits on the copy: |

Block	Byte(s)	From	To
$381	$F3	20 00 37	EA EA 38

| 3 | Write the block back to the copy. |

Softkey for...

Top Draw v1.Ø1A
Styleware Inc.

■ **Requirements**

□ 512K Apple IIgs
□ 3½"disk-copier
□ 3½"disk-editor
□ *Copy II Plus v8.1*

The softkey in COMPUTIST #48 did not work for my copy of *Top Draw*. However, using the listed 'from' bytes I was able to find the protection routine. When I found the 'from' bytes, I noticed some text like: "about to check", "got a real one!", and "Please insert the original *Top Draw*....".

First, my copy had bad blocks from $2EF through $2F3, so I thought that *Top Draw* would check for the errors. However, after checking the protection, Styleware appears to be doing a nibble-count on tracks $2Ø and $21 of side one.

I loaded block $393 in and found the following:

```
008: 20 10 42     JSR 4210
00B: EE CØ 42      INC 42CØ
00E: AD C2 42      LDA 42C2
011: DØ 13         BNE Ø26 (+13)
013: 22 D3 56 ØØ   JSL ØØ56D3
017: AB            PLB
018: 6B            RTL
```

Later in the same block:

```
0E8: C2 30         REP #30     16 bit wide Accum.
ØEA: AD BE 42      LDA 42BE
ØED: DØ 1D         BNE 1ØC (+1D)
ØEF: AD BA 42      LDA 42BA
ØF2: C9 BØ 1D      CMP #1DBØ
ØF5: 30 15         BMI 1ØC (+15)
```

```
ØF7: C9 79 E1      CMP #1E79
ØFA: 10 19         BPL 1ØC (+1Ø)
ØFC: AD BC 42      LDA 42BC
ØFF: C9 40 1F      CMP #1F40
102: 30 08         BMI 1ØC (+Ø8)
104: C9 09 20      CMP #2009
107: 10 03         BPL 1ØC (+Ø3)
109: 9C C2 42      STZ 42C2
10C: 60            RTS
```

The only important instruction seems to be the STZ (STore Zero). With this in mind, I went back up to the first portion of the code and changed the LDA 42C2 to a STZ 42C2 and the BNE 26 to NOP NOP.

After searching the whole disk, I did find the protection routine a second time, but it starts on block $52Ø and continues onto block $521.

Now you no longer need a key disk and may upload the programs to a hard disk.

The step-by-step method would look like:

| 1 | Copy the 3½"disk. |

| 2 | Make these edits (on the copy): |

Block	Byte(s)	From	To
$393	$ØE	AD	9C
	$11	DØ 13	EA EA
$520	$154	AD	9C
	$157	DØ 13	EA EA

| 3 | Write the block to the copy. |

Softkey for...

Destroyer
Epyx

■ **Requirements**

□ 512K Apple IIgs
□ 3½"disk-copier
□ 3½"disk-editor
□ *Copy II Plus v8.1*

Having already cracked *Winter Games* and *World Games*, I thought that *Destroyer* would use the same type of protection. As it turned out, Epyx did use the same disk-check but a different type of flag system for a pass or fail.

I looked at the two other Epyx protection routines and found they started out in the same way: 08 E2 30 A2 20 A0 01.

I booted up *Copy II Plus v8* and scanned the disk for A2 20 A0 01 and found the following on block $3D:

```
5D: E2 30        SEP #3Ø      8 bit wide Accum.
5F: A2 20        LDX #2Ø
61: A0 01        LDY #Ø1
63: 20 B1 00     JSR ØØB1
66: 8E D5 00     STX ØØD5
69: 8C D6 00     STY ØØD6
6C: A2 21        LDX #21
6E: A0 01        LDY #Ø1
70: 20 B1 00     JSR ØØB1
```

```
73: 8E D3 00     STX ØØD3
76: 8C D4 00     STY ØØD4
79: 22 9C Ø2 ØØ  JSL ØØØ29C
7D: C2 30        REP #3Ø     16 bit wide Accum.
7F: AD D5 00     LDA ØØD5
82: C9 40 1F     CMP #1F40
85: 90 1E        BCC A5 (+1E)
87: C9 08 20     CMP #2008
8A: BØ 19        BCS A5 (+19)
8C: AD D3 00     LDA ØØD3
8F: C9 BØ 1D     CMP #1DBØ
92: 90 11        BCC A5 (+11)
94: C9 78 1E     CMP #1E78
97: BØ ØC        BCS A5 (+ØC)
99: A9 06 00     LDA #0006   Set flags for pass
9C: 8D B9 Ø1     STA Ø1B9
9F: A9 07 00     LDA #0007
A2: 8D BF Ø1     STA Ø1BF
A5: 22 A8 ØØ E1  JSL E1ØØA8
```

Again, like the two Epyx programs before, *Destroyer* is using a nibble count on tracks $2Ø and $21 of side one. When making the code skip the disk check and set the flags to pass, the accumulator is switched from 8 bits to 16 bits.

So I chose two 2 byte instructions that wouldn't change (when loaded in and relocated) for the patches. I change the SEP #3Ø (E2 30) to BRA 97 (80 38) and the BCS A5 (BØ ØC) to REP #3Ø (C2 30).

With these edits, the program will now boot and you can play the whole game from an unprotected backup.

The protection routine is found in the file *SYSTEM.SETUP* in the SYSTEM/SYSTEM.SETUP subdirectory.

Step by step

| 1 | Copy the 3½"disk |

| 2 | Make these edits (on the copy): |

Block	Byte(s)	From	To
$3D	$5D	E2 30	80 38
	$97	BØ ØC	C2 30

| 3 | Write the block back to the copy. |

Softkey for...

Winter Games
Epyx

■ **Requirements**

□ 512K Apple IIgs
□ 3½"disk-copier
□ 3½"disk-editor
□ *Copy II Plus v8.1*

The protection on *Winter Games* is the same as on *World Games*. The entire protection code is in the file WNG.SYS16, and follows the same pattern as *World Games*.

First $E12AF3 is loaded with the value of 0072, then a disk check is run. If the check passes, nothing happens (except for returning of course). However, if the check fails then $E12AF3 gets loaded with a different value.

Again, if you would like to check out the actual disk check it starts at byte $A5 on block $61E. All you need to do is stop the check from ever being made, then you would have an unprotected copy.

Here is what I found on the disk:

Block $61E:

```
5E: A9 72 00    LDA #0072
61: 8F F3 2A E1 STA E12AF3
65: 20 60 00    JSR 0060
```

Block $423:

```
BE: AF F3 2A E1 LDA E12AF3
C2: C9 72       CMP #72
C4: D0 FC       BNE C2 (-4)   An endless loop.
```

So with the above information, a softkey becomes fairly easy. Simply change the JSR instruction (20) to a LDA instruction (AD).

Step-by-step

`1` Copy the disk and make the following edits:

Block	Byte(s)	From	To
$61E	$65	20	AD

Optional:

Block	Byte(s)	From	To
$423	$BE	AF F3 2A E1 C9 72 D0 FC	A9 72 8F F3 2A E1 EA EA

Jeff Hurlburt

The

PRODUCT MONITOR

Ratings

✪✪✪✪✪	SUPERB
✪✪✪✪	EXCELLENT
✪✪✪	VERY GOOD
✪✪	GOOD
✪	FAIR
✊	POOR
✊✊	BAD
✊✊✊	DEFECTIVE

The Eternal Dagger

Strategic Simulations

✪✪✪✪

■ **Requires:**

☐ 48K Apple II series
☐ one 5¼"drive
☐ second drive is recommended

When a plague of demon-spawned monsters from the Middle World threatens dear old Arghan, the Fellowship is quick to establish a magical link with Ssur, the last friendly MW wizard. No, they're not going to fight the monsters; the portal they have opened is for you and your seven brave comrades. It's a one-way ride with no guarantees save one: win or lose. You will not soon return from the quest for the *Eternal Dagger*.

To put things in perspective, it appears that this sequel to *Wizard's Crown* began as two separate games. Neither, evidently, being quite large enough, they were combined, with the result that *The Eternal Dagger* can deliver over one hundred hours of play and two distinct scenarios.

You begin at Ssur's tower on a continent-sized island swarming with undead warriors, evil sorcerors, ogres, giants, etc.— all in the service of the Necromancer. Since there are no 'friendlies', they've all been transformed into monsters. With the only city, Greenbay, in ruins, your party must live off the land. Weapons and armor come from enemy carcasses and hidden caches; food is obtained by hunting or prayer. To win this scenario, you must take out a super dragon, recover the Holy Morningstar, and polish-off the Necromancer. Whereupon, Greenbay is restored. Now, you can board a ship to Elven Isle and really get down to business! (i.e. secure the Eternal Dagger and close the portal into the Demon World.)

Altogether, your party will explore three scrolled-map Ultima-type continents, there's also a Drarven Isle, and battle through five multi-level 'dungeons'. In the latter, all party members become visible (and separately controllable) as you explore rooms and passages packed with furniture, statues, bookcases, treasure, and vital implements. Also, there are guards to overcome, clever puzzles to solve, and even a princess to rescue!

In one of three thriving cities, you can unload booty, obtain free healing, purchase weapons/armor enhancements, or make use of the alchemy lab. (Juicy hint: a high-ALCHEMY sorceror can conjure a small fortune producing the right kind of spells.)

Upon any encounter, you have the choice of tactical combat or a quickie "computer resolution". Thanks to souped-up magic and other new features, you will find the latter alternative much less attractive than in *Wizard's Crown*. Mainly, tactical combat is just too much fun.

Your fighters, priests, thiefs, (sorcerors, rangers, etc.) may be 'gated-in' from *Wizard's Crown*, newly created, or come from a 'pre-developed' group. As in *Wizard's Crown*, each character may combine two or more occupations (e.g. priest-ranger); and you spend "experience" points to develop a host of skills and attributes. Add an excellent system for weapons differentiation and enhancement, and you have one of the best character development schemes anywhere.

Well documented and easy to 'get-into', *The Eternal Dagger* offers decent sound and attractive maps, but falls a bit short in the frills area (e.g. no music, no hires text). Still, for hard-core questers and newcomers alike, the game is an extraordinary, consistently entertaining, adventuring experience— a legitimate candidate for "Best Map-Maze Adventure Ever'.

Donald Duck's Playground

Sierra

✪✪

■ **Requires:**

☐ 64K Apple II series
☐ one 5¼"drive
☐ Joystick

For computists at a loss for software to entertain young visitors, Sierra has come to the rescue. Including an illustrated manual, *Donald Duck's Playground* aims to teach addition, subtraction, and change-making as Donald earns and spends money to build a playground for his nephews.

Guiding Donald along Main Street, you can check-in for work sorting produce, go to the airport and load packages, or visit the train station to direct the Amquack Special as it makes pickups and deliveries. Across the way are shops run by Minnie, Mickey, and Goofy, just the places to select and purchase the ladders, swings, slides, and other equipment you'll need. Purchased items are automatically sent, just down the street, to the playground site, a three-level, 3x5 grid with space for fifteen

pieces. Of course, you can visit at any time, to adjust equipment placement just so and watch one of your nephews give everything a good tryout. (At this time, you control the nephew's movement.)

Featuring good hi-res artwork and animation, plus some nifty little tunes, *Donald Duck's Playground* looks like it should be fun for the targeted 7-11 age group; and, sure enough, it is. Loaded with instructional value, this one is a guaranteed entertainer.

Fast Frames

Electronic Art's *Marble Madness*

If you think it's easy to guide your fearless marble through a cartoon world of Escher landscapes and Rube Goldberg contraptions infested by kama-kazi steelies and other baddies— well, you're half right.

In the IIgs release of *Marble Madness* (512K, $34.95 from Electronic Arts), levels I—III are a breeze. Thereafter, it's time to strap on your Arcade Ace helmet.

Having tried *Marble Madness* on a couple smaller machines, there's no question that detailed super-resolution displays and IIgs-powered 'marble music' make a world of difference in player enjoyment. It's too bad EA doesn't complete the parlor-to-PC retrofit.

At present, you get only about one minute per level. This amount of time is not nearly enough to try out all the interesting gadgets and high scores are not saved to disk.

Even a simple GAME SAVE would help; now you must repeat all the easy levels just to get another shot at the toughies.

Micro Prose's *Silent Service*

For the combat simulation devotee who craves realistic, long-play strategy and tactics, it's hard to imagine a better choice than the IIgs release of *Silent Service* from Micro Prose (512K, $39.95).

The game offers both short "practice" sorties and a choice of several full-scale South Pacific missions with ranks and scores, based upon performance, saved to disk.

Documentation is thorough; and, except for combining sonar and radar functions in a single "map" display, attention to technical detail borders on the obsessive— down to a choice of steam or electric torpedoes.

With good speed, excellent graphics, and pounding sound to deliver WWII-movie-class realism, all you need to do is turn up the amp, turn down the lights, and DIVE!

Updates, Etc.

The Bard is Back, Really

As *The Bard's Tale*/IIgs programmers discovered, when Apple warned developers to employ only toolbox calls or documented ROM hooks to access firmware, it wasn't kidding. Well, a new ROM-compatible version of the Electronic Arts classic has, at last, been released.

The package I received is distinguished by an orange spine sticker.

More Courses

Accolade's *Mean 18*, undoubted number one in IIgs golf, has acquired several new courses. Supplied in *Famous Course Disk* supplements (Volumes II-IV, three courses per diskette, $19.95 each) are Turnberry, Inverness, Muirfield, Doral, and other favorites.

The Mutating Computer

When Apple replaced first, the video chip, then the ROM, many IIgs owners, no doubt, began to wonder when the design would be finished.

Not to worry; with talk of an impending motherboard swap offer (due around mid-year), it is apparent that Big Green has hit upon the marketing coup of the decade, the "mutating computer"! Basically, every two years or so, the MC makes a sort of 'evolutionary hop', pacing new developments in hardware and firmware. (According to rumor, the next 'hop' will boost processor speed and slash ProDOS /16 boot time.)

At $200-$300 per swap, the cost vis-a-vis a 'new model' is relatively minor; in return for which MC (IIgs) owners enjoy continually upgraded performance— not a bad stratagem.

Too Cold to Boot?

Here's a wicked little puzzle, which is likely to be of more than passing interest to a few IIgs types.

On a cool autumn morning, a few weeks after getting the ROM upgrade, you turn on your computer. This time, instead of booting, it plays around with the drives and then 'locks-up' with a screen full of garbage. If you wait twenty or thirty seconds, turn off the machine, and try again; everything works fine! What's the problem?

If you answered "a bad video chip", give

yourself a pat on the nose. Evidently, some of the upgrade video IC's malfunction at low room temperature, but function normally after just a little warm-up. (We zeroed-in on the culprit using Radio Shack circuit coolant.)

Due to slight differences in the new vs. old ROM reset sequence, the bug is less likely to show until the new ROM is installed.

If you have experienced similar, inexplicable cold-start boot-up problems, get your dealer to try a video chip swap.

Bill Jetzer

Softkey for...

Animate
Broderbund

■ **Requirements**

☐ 6 blank disk sides
☐ ProDOS disk containing the files *PRODOS* and *BASIC.SYSTEM*
☐ Super IOB 1.5

Animate is an excellent new product released by Broderbund. It allows anyone to create professional quality double hi-res graphics animation. With the easy-to-use drawing and editing capabilities of the program, you can easily create simple animation sequences, such as a man walking or a bird flying.

You can also use a variety of characters and backgrounds already included. You could make a show disk for your creations, or use the Watch command included with *Animate*.

The protection used on the program disk is difficult to figure. On the front, a sequence of bytes is searched for between tracks and stored in memory. Later, these bytes are used to check if the disk is a copy or not. The back appears to have almost the same protection with a little something extra that I couldn't figure out.

All that is needed to deprotect the front of *Animate* is to change two branch instructions to keep searching for the sequence of bytes instead of rebooting.

The back side had some similar protection, but it was actually part of another routine that also checked for protection. I wasn't sure what to do about it, so I just changed some branch instructions so that the routine was never executed, and it worked fine.

To deprotect *Animate*, enter the controllers at the end of the article and save them. Install the "SIDE A" controller into *Super IOB 1.5* and run it to copy the front side. Install the "SIDE B" controller into *Super IOB 1.5* and run it to copy the back side.

The Animation Library, Background Library, Demo Show disk and Programmers Aid disks are all unprotected, so you can make

a copy of those with any whole disk copier.

I had hoped to be able to quit to ProDOS from *Animate*, but several attempts proved futile. Since you have to boot the disk to use *Animate* anyway, it's not so serious.

To use *SHOW.SYSTEM*, however, you can start from ProDOS, and having to reboot after using it each time is quite annoying. To make *SHOW.SYSTEM* quit to ProDOS, follow these steps:

1 Insert the front side of the deprotected *Animate* disk.

BLOAD SHOW.SYSTEM,A$2000,TSYS

2 Change a few quit messages.

```
4658:F4 EF A0 D0 F2 EF C4 CF
4660:D3 A0
4BF0:D1 D5 C9 D4 A0 A0
```

3 Replace the old quit routine with the new one.

```
4C06:20 00 BF
4C10:65 0C 7C 04 00 00 00 00
4C18:00 00
```

4 Save the file back to disk.

UNLOCK SHOW.SYSTEM
BSAVE SHOW.SYSTEM,A$2000,TSYS

If you want to have the same patches on the included demo show disk, insert a copy of the demo show disk and repeat step 4.

Controller A

```
1000 REM ANIMATE (SIDE A)
1010 TK = 0 : LT = 35 : ST = 15 : LS = 15 : CD = WR : FAST
     = 1
1020 POKE 47426,24 : GOSUB 490 : GOSUB 610
1030 POKE 47426,56 : GOSUB 490 : T1 = TK : TK = PEEK
     (TRK) - 1 : RESTORE : GOSUB 310 : TK = T1 :
     GOSUB 610
1040 IF PEEK (TRK) = LT THEN 1060
1050 TK = PEEK (TRK) : ST = PEEK (SCT) : GOTO 1020
1060 HOME : PRINT "COPY*DONE" : END
5000 DATA 2*CHANGES
5010 DATA 0,3,65,0
5020 DATA 0,3,79,244
```

Controller Checksums

1000	– $356B	1050	– $9114
1010	– $2544	1060	– $FEE3
1020	– $B5FB	5000	– $AFE0
1030	– $BB1C	5010	– $CF6F
1040	– $C168	5020	– $F032

Controller B

```
1000 REM ANIMATE (SIDE B)
1010 TK = 0 : LT = 35 : ST = 15 : LS = 15 : CD = WR : FAST
     = 1
1020 POKE 47426,24 : GOSUB 490 : GOSUB 610
1030 POKE 47426,56 : GOSUB 490 : T1 = TK : TK = PEEK
```

```
(TRK) - 1 : RESTORE : GOSUB 310 : TK = T1 :
     GOSUB 610
1040 IF PEEK (TRK) = LT THEN 1060
1050 TK = PEEK (TRK) : ST = PEEK (SCT) : GOTO 1020
1060 HOME : PRINT "COPY*DONE" : END
5000 DATA 3*CHANGES
5010 DATA 2,0,137,240
5020 DATA 2,0,138,1
5030 DATA 15,11,51,18
```

Controller Checksums

1000	– $356B	1060	– $FEE3
1010	– $2544	5000	– $FE60
1020	– $B5FB	5010	– $50BC
1030	– $BB1C	5020	– $E1C8
1040	– $C168	5030	– $6DC6
1050	– $9114		

Brian Symonds

I read with interest your comments under the letter from Larry Newby in COMPUTIST #52. I am one of the "dozens" who have previously sent to you an Appleworks database file that could be used as an index for the COMPUTIST. I have sent Mr. Newby a copy of my file at no charge and ordered a copy of his. Perhaps if everyone who has such a file could send a copy to Mr. Newby, he could amalgamate them and we would then have the "ultimate" file (if he is willing). I will send a copy of my file free of charge to anyone who can prove that they have a similar file. My file does contain copying methods where known and protection and deprotection methods where known. For anyone else wishing a copy my previous offer still stands — send $5 and a disk and I will send it to you.

4637 Willingdon Ave.
Powell River, B.C.
V8A 2N1
Canada

The purpose is not to make money, but not to lose it either. Hats off to Mr. Newby for his generous offer, especially considering the time and work involved.

How about taking a poll to see how many of your readers would like to be able to submit letters in Appleworks word processor files or at least in ProDOS text files?

I'm very sorry if I offended you. I most certainly didn't mean too. We received so many letters about Appleworks databases, not to mention the readers who sent copies of their disks, that printing them all would have taken several pages. We kicked it around and decided to print the most reasonable offer (criteria was cost and content, and the decision admittedly arbitrary) and simply mention the rest.

It probably isn't necessary to take a poll. A great many of you already send your letters on ProDOS disks in Appleworks files. We simply

---

convert them to text and transfer them to DOS 3.3. The reason we ask for DOS 3.3 text files is that we have an old (very old) Corvus hard-disk network system that is tied into a typesetter. The system does not support ProDOS so all files have to be in DOS 3.3 format.

If you convert your Appleworks letter to text (with the print-to-text-file option) and transfer it to a DOS 3.3 disk, it's just one file. For us to do all the conversions would mean dozens of files and would be time-consuming. It's tough enough getting local volunteers to stop by and do some editing. So if you have to use ProDOS, it's okay. But if you've got the time, give us a hand. The more work you do the easier it is for us and if you think about it, it makes you one of the volunteers, even if you never make it to Tacoma. . . . . RDEXed

---

Don McClelland

Softkey for...

Sesame Street
Electric Coloring Book Series
Polarware

■ Requirements

☐ Advanced *COPYA*
☐ Disk Search Utility
☐ Sector Editor
☐ Blank Disks

The *Sesame Street Coloring Book* series is a cute series of programs that allow children to color pre-made pictures and learn the alphabet or numbers from 1 to 10. Each disk in the series can be purchased for $13.00 to $20.00 depending on your retail outlet. Polarware, formerly Penguin Software, unfortunately protected the disks. The disks are in ProDOS format. Both programs use a loader to put the pictures on screen. The pictures are not stored as files so you will not be able to move the programs to a mass storage device.

Softkey for...

Sesame Street Numbers
Polarware

The disk is formatted with the standard address and data marks, $D5 AA 96 and $D5 AA AD respectively. This makes things easy. First, copy the disk with COPYA. This will produce a copy that will not work.

At the second screen, the program does what appears to be a nibble count. To defeat the nibble count, we must find where the program accesses the disk drive. I searched the disk for the bytes $BD 89 C0. These bytes access the disk drive and turn it on. The bytes were

found on track $00 sector $0E, track $04, sector $0B and track $12, sector $0D.

Analyzing the code on each of the track/sector locations, it appeared that track $12, sector $0D was the nibble routine. The code looks for the sequence of bytes D5 AA BB then counts the number of FF's then turns the drive off. As the routine exits, it sets the Carry Flag if an error occured and clears the Carry Flag if all went well.

So we have a couple of options. We can clear the Carry Flag and Return to the main program before the disk access or we can let it do the nibble count then clear the carry. I chose the first option just to save on disk access. I accomplished this by putting 18 60 at track $12 sector $0D. Now you have a deprotected disk.

1 Copy the disk with *COPYA* or any other Fast copy program.

2 Edit the disk.

Trk	Sct	Byte(s)	From	To
$12	$0D	$00	?	18 60

Softkey for...

Sesame Street Letters for You
Polarware

This disk uses ProDOS as its operating system, but has the familiar Penguin Address Marks. Even tracks (0,2,4,6, ect.) have the normal address marks of D5 AA 96. Odd tracks (1,3,5,7, ect.) have the altered address marks of D4 AA 96. On all tracks, the address trailers are modified from DE AA to FF FF. Also on further investigation, track $02 is not formatted, making *COPYA* print a "READ ERROR" and stop.

Advanced COPYA was used for its ability to continue copying on read errors. This allows it to copy right across track $02.

✍ *For those of you who don't have Advanced COPYA, (I've never heard of it either) a simple change to COPYA will make it ignore errors. After the binary file COPY.OBJ0 is loaded, stop COPYA and change the byte at $3A1 from 60 to 18 (POKE 929,24). Restart COPYA with* **RUN 80**.........................*RDEXed*

1 Boot a DOS 3.3 disk and run Advanced *COPYA*.

2 Stop the program and make some changes.

[C]
CALL -151
B993:00	*ignore address epilogue*
B99D:00	*ignore address epilogue*
B957:00	*ignore 1st byte of prologue*

[C]
RUN

That's all there is. Remember that ProDOS and DOS 3.3 are exactly the same at the disk level. The major difference is that a ProDOS Block is made up of two DOS 3.3 Sectors. This allows you to utilize your favorite DOS 3.3 deprotection utilities on ProDOS formatted 5.25" disks.

J. Nissel

Softkey for...

Spy's Adventures In Europe
Penguin/Polarware

1 Boot your DOS 3.3 system disk.
CALL-151
B954:29 00
B988:18 60
RUN COPYA

2 Copy side 1 of your disk. Use normal disk copy for side 2. That's it.

Softkey for...

Tellstar II
Scharf Software Systems

1 Boot your DOS 3.3 system disk.
2 Put a blank disk in the drive.
FP
INIT TELLSTAR

3 Boot the original *Tellstar II* disk and after your drive reads the first 3 tracks reset into the monitor.

4 Move the RWTS to a safe location.
1900<B800.BFFFM

5 Put your copy disk in the drive and boot it.

6 Place your *Super IOB* data disk in the drive and save the RWTS .
BSAVE RWTS.TELLSTAR II, A$1900, L$800

7 Install the controller listed below into *Super IOB* and copy the original to the disk that you initialized with *TELLSTAR*.

8 When asked, enter N, so as not to format the disk while running the controller.

Controller

```
1000 REM TELLSTAR II CONTROLLER
1010 TK = 3 : ST = 0 : LT = 35 : CD = WR
1020 T1 = TK : GOSUB 490 : GOSUB 360 : ONERR GOTO 550
1030 GOSUB 430 : GOSUB 100 : ST = ST + 1 : IF ST < DOS
     THEN 1030
1040 IF BF THEN 1060
```

```
1050 ST = 0 : TK = TK + 1 : IF TK < LT THEN 1030
1060 GOSUB 490 : TK = T1 : ST = 0 : GOSUB 360
1070 GOSUB 430 : GOSUB 100 : ST = ST + 1 : IF ST < DOS
     THEN 1070
1080 ST = 0 : TK = TK + 1 : IF BF = 0 AND TK < LT THEN
     1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "COPY^DONE" : END
10010 IF PEEK (6400) < > 162 THEN PRINT CHR$
     (4) "BLOAD^RWTS.TELLSTAR^II.A$1900"
```

Controller Checksums

1000	–	$356B	1060	–	$90D8
1010	–	$3565	1070	–	$98DD
1020	–	$E1E8	1080	–	$7422
1030	–	$F7E9	1090	–	$2DB2
1040	–	$D35A	1100	–	$1A55
1050	–	$EB5B	10010	–	$7CB4

You can put fast DOS on this copy if you want to.

Softkey for...

Super Sunday Football
Avalon Hill Games

1 Boot your DOS 3.3 system disk.
2 Put a blank disk in the drive.
FP
INIT HELLO

3 Boot your original disk and after your drive reads the first 3 tracks, reset into the monitor.

4 Move the RWTS to a safe location.
1900<B800.BFFFM

5 Put the copy disk you just initialized in the drive and boot it.

6 Place your *Super IOB* data disk in the drive and save the RWTS.
BSAVE RWTS.SUPER SUNDAY, A$1900, L$800

7 Install the controller listed below into *Super IOB* and copy the original to the copy disk. When asked, enter N, so as not to reformat the disk.

If your drive head bangs while reading some of the sectors on this disk don't worry about it, the copy should still come out ok. This is only true while reading the sectors, not while writing them.

Controller

```
1000 REM SUPER SUNDAY FOOTBALL CONTROLLER
1010 TK = 3 : ST = 0 : LT = 35 : CD = WR
1020 T1 = TK : GOSUB 490 : GOSUB 360 : ONERR GOTO 550
1030 GOSUB 430 : GOSUB 100 : ST = ST + 1 : IF ST < DOS
     THEN 1030
```

```
1040 IF BF THEN 1060
1050 ST = 0 : TK = TK + 1 : IF TK < LT THEN 1030
1060 GOSUB 490 : TK = T1 : ST = 0 : GOSUB 360
1070 GOSUB 430 : GOSUB 100 : ST = ST + 1 : IF ST < DOS
     THEN 1070
1080 ST = 0 : TK = TK + 1 : IF BF = 0 AND TK < LT THEN
     1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "COPY°DONE" : END
10010 IF PEEK (6400) < > 162 THEN PRINT CHR$
      (4) "BLOAD°RWTS.SUPER°SUNDAY.A$1900"
```

Controller Checksums

1000	– $356B	1060	– $90D8
1010	– $3565	1070	– $98DD
1020	– $E1E8	1080	– $7422
1030	– $F7E9	1090	– $2DB2
1040	– $D35A	1100	– $1A55
1050	– $EB5B	10010	– $255B

[8] Boot your DOS 3.3 system disk.

[9] Put the copy you just made into the drive and enter the following:

FP
10 D$ = CHR$(4)
20 PRINT D$'BRUN BOOT"!
SAVE HELLO

You can put fast DOS on this copy if you want to.

Softkey for...

Maxwell Manor
Avalon Hill Games

[1] Boot your DOS 3.3 system disk.

[2] Put a blank disk in the drive.

FP
INIT HELLO

[3] Boot your original disk and after your drive reads the first 3 tracks, reset into the monitor.

[4] Move the RWTS to a safe location.

1900<B800.BFFFM

[5] Put the copy disk you just initialized in the drive and boot it.

[6] Place your *Super IOB* data disk in the drive and save the RWTS.

BSAVE RWTS.MAXWELL MANOR, A$1900,
L$800

[7] Install the controller listed below into *Super IOB* and copy the original to the copy disk. When asked, enter N, so as not to reformat the disk.

Controller

```
1000 REM MAXWELL MANOR CONTROLLER
1010 TK = 3 : ST = 0 : LT = 35 : CD = WR
1020 T1 = TK : GOSUB 490 : GOSUB360 : ONERR GOTO 550
1030 GOSUB 430 : GOSUB 100 : ST = ST + 1 : IF ST < DOS
     THEN 1030
1040 IF BF THEN 1060
1050 ST = 0 : TK = TK + 1 : IF TK < LT THEN 1030
1060 GOSUB 490 : TK = T1 : ST = 0 : GOSUB 360
1070 GOSUB 430 : GOSUB 100 : ST = ST + 1 : IF ST < DOS
     THEN 1070
1080 ST = 0 : TK = TK + 1 : IF BF = 0 AND TK < LT THEN
     1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "COPY°DONE" : END
10010 IF PEEK (6400) < > 162 THEN PRINT CHR$
      (4) "BLOAD°RWTS.MAXWELL°MANOR,A$1900"
```

Controller Checksums

1000	– $356B	1060	– $90D8
1010	– $3565	1070	– $98DD
1020	– $E1E8	1080	– $7422
1030	– $F7E9	1090	– $2DB2
1040	– $D35A	1100	– $1A55
1050	– $EB5B	10010	– $8D2D

You can put fast DOS on this copy if you want to.

John B. Walkey

Apple IIc Paddle Fix

At the Lethbridge Community College, I am coordinating a Computer In-service Training Program. Amongst other things, my job involves teaching school teachers to use computers. In working with teachers, I have found some interesting hardware problems. One that has stumped me for some time is that several of the Apple IIc computers in the schools are no longer able to read the x-axis from the joystick input. The joystick works fine up/down, but not at all left/right. The same joystick tested on another computer works fine. This led me to suspect the computer as the source of the problem. In one school, 4 of 11 computers have this problem.

Upon examination inside the Apple IIc, there are 3 IC's at the rear near the input/output (I/O) jack for the joystick/mouse port. One chip has PDL 556 screened on the PC board. The 556 is a dual timing chip and seemed like a good candidate to be responsible for the problem. I cut out the existing chip and socketed in a good 556, but the problem persisted. The second most likely candidate is the 74LS251 IC which is a data selector/multiplexer. Replacing this chip fixed the problem.

The information which follows describes how to fix the Apple IIc paddle problem. The following procedure will void any Apple warrantee. If you are inexperienced with soldering, desoldering or handling integrated circuits then get some help with this fix.

To begin, the case will have to be opened. Figure 1. represents the back of the Apple IIc. There are 2 small plastic clips indicated at points 1, which will need to be carefully pried open. A medium flat headed screwdriver will be needed for the job. Gently pry these open, first on one side then on the other. The second set of clips can be found by using the flatheaded screwdriver to pry the case apart at points 2. There are 3 clips in the groove between the smooth case bottom and the upper ribbed section. Gently pry in the groove to separate the upper and lower case.

With the back off, the six Phillips headed screws on the bottom (four under the keyboard and two at the back by the handle) may be removed. The four remaining screws on the bottom hold the disk drive in the case. Remove them as well.

Place the computer upright and gently pry the case apart along the split around the side of the case. Lift off the top. The keyboard may now be unplugged and removed from the board. Next, the disk drive may be unplugged and removed. Also, unplug the speaker jack.

The motherboard is held in by about ten Phillips screws and these need to be removed. With all the screws out, the motherboard can be removed from the case.

Locate the 74LS251 chip at the right-rear of the motherboard. This is the chip that needs to be replaced. Figure 2. indicates the location of the 74LS251 chip.
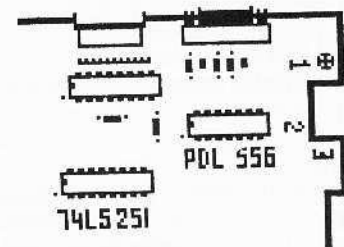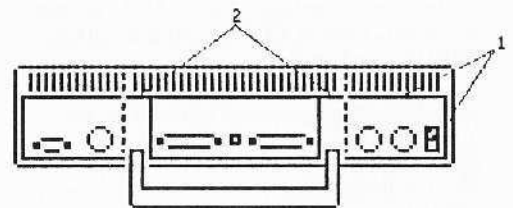
With the motherboard out, the 74LS251 chip must be cut out. A fine pair of wire cutters works well. Care must be taken not to damage other components on the board while cutting the chip out. When the chip is cut out desolder the pins left in the board. Do not overheat the printed circuit board by holding the soldering iron on the board too long. A medium Wattage, 23 or so, soldering iron should be used with a low-static suction desolder tool as opposed to desoldering braid. After the board is clean, solder in a low profile 16 pin socket. The new 74LS251 chip may be inserted observing the pin 1 location. The screened dot on the motherboard corresponds to pin 1 on the IC.

Reassemble the computer in the reverse order it was taken apart.

To test the repaired system, you may use this BASIC program.

10 PRINT PDL(0), PDL(1):GOTO 10

A final point relates to the cause of this problem. With the older Apple II+ and Apple IIe computers the game I/O port served a unitary purpose. However, with the Apple IIc the game port also serves as a mouse port and a switching operation sets the function. On older Apples, plugging-in or unplugging an I/O device would not ordinarilly cause any damage. By contrast, the Apple IIc may be damaged by plugging-in or unplugging I/O devices with the power on. To avoid this quite tricky repair job, I would suggest not plugging-in or unplugging I/O devices with the power on if you are using an Apple IIc.

Mike Neuliep

Softkey for...

Ring Quest
Penguin Software

The first thing I do after purchasing a disk is try to deprotect it. And the first step in trying to deprotect is to see if the disk is in a normal format. I soon found that this was not the case because *COPYA* refused to copy. I used the nibble-editor from Copy II+ and found that the formatting was similar to that of other Penguin products. The data epilogs were changed from the standard DE AA to DA AA. All the even track numbers are formatted somewhat normally, but on the odd number tracks the data headers had been changed from the standard D5 AA 96 to D4 AA 96. With this in mind we can set out to deprotect the disk.

One easy way to get a protected disk in a normal format is have it read the strangely formatted disk and write in a normal format. *Super IOB*'s RWTS Swap Controller depends heavily upon this. Another way to do it is to change the part of DOS that reads so it will read in the protected format and write in the normal

format. That's how I deprotected *Ring Quest*.

1 Boot the DOS 3.3 System Master and run *COPYA*.

```
RUN COPYA
□C
70
CALL-151
```

2 Now change the part of DOS that reads the disk so it'll read the protected disk.

```
B934:C9 DA
B990:C9 DA
```

The old values were C9 DE for both of them. We changed them to C9 DA so we can read Penguin's altered data field epilogs.

3 Now tell DOS that the even tracks have a normal address header.

```
B954:4A 49 6A D0 EF
```

4 Clear the carry codes. This tells DOS that if you run into an error,then don't worry about it.

```
B925:18 60
B942:18 60
B988:18 60
```

5 Now run the program and copy both sides of the disk.

```
D566G
```

6 When that's done sector edit side one.

Trk	Sct	Byte(s)	From	To
$00	$03	$42	?	18

7 Write-protect both sides of the disk. If you don't, the program will refuse to show the graphics in the game and then self-destruct.

Anyone wishing to discuss copy protection, in general, or anyone who has scored more than 250 points on *The Hitchhiker's Guide to the Galaxy*, can write me. I always blow myself up when putting the interface in the vending machine. Can anyone help?

Larry D. Hough

More Franklin Ace notes
Protected IIc utilities

I share a subscription with a friend of mine and let me first say that I'm enjoying the magazine a little more each issue.

After reading the last issue, my son and I were able to work with his *Might and Magic* characters and change them around. This was our first experience in working with the innerworkings of a program. It proved to be quite interesting. After working with this one program, it brought up another question.

(?) I have a Laser 128 and would like

to try and use the Apple IIc utility disk that my son has for his machine. We both use the *Copy II Plus* that I bought when I purchased the Laser 128; however, after seeing what can be accomplished with the various programs, I wanted to inquire about his utility disk.

When I boot the IIc Utility disk, I get the following notation and then am unable to move anywhere. "The System Utilities program is designed to work only with the Apple IIc."

One thing that bothered me was the so-called software piracy. I didn't think that what I am trying to accomplish is in any way piracy. It's more a matter of being inquisitive and delving into the programming part of computing.

Any comments that you could add would be most appreciated.

Does anyone know how to circumvent the protection on the IIc utility disk. It sounds a lot like the problem with ProDOS and the Franklin Ace computer.................RDEXed

---

Alan T. Zak

Modification for...

Super Boulderdash
First Star

A note on the *Super Boulderdash* softkey. On my version of *Super Boulderdash* from First Star Software, the softkey in COMPUTIST #50 worked — to a degree.

Unfortunately, the graphics become scrambled up slightly and the game is pitiful to play. To solve this, you will need the following:

■ **Requirements**

☐ COMPUTIST #50
☐ *Super Boulderdash* backed up from softkey in COMPUTIST, Issue #50
☐ *Copy II Plus* sector-editor

Trk	Sct	Byte(s)	From	To
$01	$0C	$00-02	4C 69 05	18 60 DD

You may decide that this modification is not worth merit if you enjoy playing specific levels. I have discovered that levels K, M—P and possibly others do not work. The entry door just flashes, and Rockford does not appear.

All in all, if you want graphics on *Super Boulderdash*; then this mod is for you (as long as you can stand a few unplayable levels).

The best softkey for Boulderdash I & II is in COMPUTIST #38. Try it before you try any of the others.................RDEXed

---

Terry MacGregor & Joe Walters

Softkey for...

Daisy Professional
Stone House Systems

The Most-Protected Award nominee!

■ Requirements

- ☐ *Daisy* v2.2 or *Daisy* v2.4
- ☐ System Master
- ☐ A slave disk with the *HELLO* program deleted.
- ☐ A master disk with a hello program named *DAISY* which has been deleted.
- ☐ *Copy II Plus*

Overview

Daisy Professional is a very fine statistical mathematics package, published by Rainbow Computing, Inc., that runs on all the Apple family computers. It is written in Applesoft BASIC and assembly language and occupies most of a disk.

The Applesoft programs provide overall program control and the user interface. Assembly language subroutines provide *Daisy* improved real-time, computation and specialized disk I/O.

Despite these specialized routines, *Daisy* still suffers from real-time performance problems. These problems can be virtually eliminated through the use of a hard/RAM disk and an accelerator card. Unfortunately, copy-protection prevents *Daisy* from being a better performing program.

For this reason and to learn more about DOS 3.3, I undertook the deprotection of *Daisy*.

While the program itself is probably of minimal interest to the reader of COMPUTIST, what I learned will be of significant interest.

The copy-protection techniques employed by Rainbow Computing are worthy of the attention of any hacker worthy of the name.

I spell out, in great detail, what happens at each stage of the game. In addition, I make available a great deal of disassembled and commented code.

The disassembled code is too extensive to print but is included in the Library Disk that is associated with this issue. . . . . . . RDEXed

Back Ground

Daisy's protection scheme turned out to be "synchronized tracks", nibble-count, self-modifying code (two versions), and checksum counts (again, two versions). As far as I know, the self-modifying code and the nibble-count techniques are a first.

I believe that *Daisy*'s copy-protection is better than anything that has appeared in COMPUTIST over the last several issues.

Initially, I thought this would be an easy job because I was able to make bootable copies using either *Locksmith 5.0* rev C or *Copy II Plus 5.5* and specifying synchronized tracks. This idea was reinforced when I was able to make an error free (but non-bootable) copy using *COPYA* from the DOS 3.3 system master disk. This demonstrated that no altered DOS marks hanky panky was going on. In retrospect, I was only partially correct.

As a starting point, I assumed that DOS was not involved (almost true) in the protection scheme. I decided to disassemble the two programs BLOAD'd (*DAISY:PARM0* and *DAISY:PARM1*) and the one that was BRUN (*DAISY:PARM2*) by *Daisy*'s hello program. This turned out to be the right approach.

The Chase

DAISY:PARM1 turned out to be data which couldn't be disassembled (wrong, stick around) so I moved on to *DAISY:PARM2*.

The first suspicious code I examined turned out to be the standard method specified by Apple to determine if you have an 80-column card (The method is documented in "Design Guidelines" which should be available, for no charge, from your local Apple dealer).

Continuing with my code-tracing, I discovered a copy-protection technique using values stored in page zero. Location $EC has $8D added to it. This location is then checked by *Daisy* to see if it is the same as the contents of $CF. If different, a transfer to $C600 (the PR#6 location) ensues. If this test passes, the boot process continues.

Continuing with my code-tracing, I found that location $7000 is modified from AD 00 E0 (LDA $E000) to AD 40 E0 (LDA $E040). Program execution continues to a point where the program performs a JMP ($7000). JMP ($7000) is an indirect transfer which takes us to $40AD which is in the middle of previously loaded *DAISY:PARM1*.

From here, there is another indirect transfer, JMP ($03FF). At this point *Daisy* capitalizes on the well-known 6502 hardware bug where the processor picks up the low byte of the destination from $3FF and the high byte from $300, rather than the high byte from $400.

At $300 DOS begins its vector table. The table contains (somewhat) fixed values. At location $300 is a JMP instruction which has an opcode of $4C while $3FF contains $FF, so we end up at location $4CFF. (As a side note, version 2.2 of *Daisy* will not work with

any Apple using the 65C02 processor. The new *Daisy* publishers (Stonehouse) corrected the problem in version 2.4 by replacing the two indirect transfers with a direct transfer to $4CFF.)

At $4CFF, I found executable code but was stymied on how it got there. As far as I can tell nothing has been loaded at $4CFF during the boot process, except the DOS image that was later moved up in memory. My focus then shifted towards understanding how *Daisy* had placed executable code at $4CFF.

I remembered COMPUTIST #5. In that issue was an article called "Getting On The Right Track." The author of the article recommended marking your disk drive cam so you could see which tracks were being accessed. This seemed like a good idea, so I did it and then booted a *COPYA* version of *Daisy*. I chose this copy because I knew it would fail to boot. I saw that tracks $00, $02, $01, $13, and $00 were accessed during boot, at which time the boot failed.

Getting smarter, I used the track/sector map utility of *Copy II Plus* to see what programs were on track $13. Holy Krakowicz!! Track $13, sectors $03, $04, and $05 were busy (not free) and yet no program on the disk claimed them.

Puzzled, I disassembled DOS BOOT2, which is the RWTS and the code that moves DOS to its final resting place. I used the dissassemble of BOOT 2, the Merlin editor, and a normal DOS listing and found out that a portion of DOS that was labeled "UNUSED" when compared to my disassembled copy was quite different. Looking more closely, I saw that the protection scheme placed an address into the RWTS control block that was for the "pointer to a client data buffer." The code then transferred to another unused hole in DOS where they finished setting up the RWTS parmlist to read track $13, sector $03 into address $4C70. RWTS was then called.

Armed with this information, my next logical step was to use *Inspector/Watson* to produce a BLOADable copy of track $13, sectors $03, $04, and $05 with a loading address of $4C70. I then used Merlin to disassemble it.

Track $13 Sector $03

($4C70 — Executable Code)

The first sector promptly stored "LDA #$06" into the DOS-cold start routine. That is the instruction that normally lives there. Looking at the contents of that address before the write, I saw there was a JMP to the first DOS hole previously discussed.

The protection scheme was not taking any chances, it was covering its tracks as quickly as possible.

Next the protection scheme changed the "JMP to just read sector" into a RTS so that when the next sector was read in, RWTS would return to the sector $03 code. When RWTS returned, a JSR to $6000 ensued (track $13, sector $04).

Track $13 Sector $04

($6000 Executable Code)

The $6000 code does several self-modifications. The first instruction decrements (DEC) the first byte of the next instruction making it a valid INC instruction. Execution of this instruction turns it back into an illegal instruction. After that, program execution continues for a couple of instructions and then branches into the middle of a JSR, JMP, illegal instruction, etc. The trick here is to disassemble up to the BNE/BEQ instruction and then take the next byte(s) as "who cares?" HEX values until we reach the destination of the transfer where we again start the disassembly. In some cases the HEX values are truly "who cares," in other cases the branched over byte(s) are constants used by the program.

This nonsense continued with little useful work except an EOR of a byte at location YY. This changed a RTS into a LDA as is shown in the program fragment below.

```
         LDY #$29   Loop counter
LOOP     TYA        Y to the A
         BCC XX     Always
         HEX 20     This looks like JSR $0059
XX       EOR $6000,Y This decodes $6929-69FF
         STA $6000,Y using a moving key,
         INY        the Y register (see loop).
         BNE LOOP
         TYA        Zero to A
         BPL YY
         HEX $4C    JMP $2960
YY       LDA #$00
```

The EOR's start at $6029 and continue through $60FF. Looking at this program fragment, one sees that the protection scheme performs a conversion of track $13, sector $04 data to executable code by using a "moving key" (the loop counter).

This scheme is atypical; hence, rendering the *Locksmith* function that allows one to "exclusive-OR" a number with the entire sector useless.

The converted code was the "synchronized track" code I for which I had initially started looking. It turns out that it is only half way synchronized. When the protection scheme finds the wanted sector $00, on track $00, it goes into a mode where it merely reads the data field starting at the last nibble of the data prologue for 343 nibbles. At that point, it transfers into another loop where it keeps reading the data from the disk without checking for the presence of a nibble for 527μs. It then

looks one time for a nibble. If a nibble can not be found, then try again in 7μs. If one is found either time, it is checked for the value C9 (test pass case).

If it fails then it self destructs all of memory except $C000-$C0FF and $0100-$01FF by writing it full of $04's. Finally, the protection scheme jumps to $FA62 which is the system reset routine where it will look as if power was just applied (because of $04 in the power up byte) and so we will boot the disk again.

Assuming the test passes, we transfer back to the top of this routine ($6000) where we go through the whole mess one more time. However, remember that the second EOR sequence puts memory back to its original state. This in essence, covers up all remains of protection scheme code. When completed, the protection scheme falls out of the EOR loop and executes a $60 (RTS) at location YY.

Program execution resumes at the $4C70 code which invoked the track $13, sector $04 code. From here, the protection scheme loads in track $13, sector $05 into $5600 and transfers there.

Here's another observation gleaned from this code-tracing centers around the nibble checked for (C9). It is an invalid disk-nibble in that it has two occurrences of adjacent zeroes (1100 1001) in the same word. The fact that this worked implies that the Apple disk drives are better than we have been told! In the following code fragment the code protection scheme arrives at the sought C9 nibble.

Register X contains the slot of the boot drive times (∗) 16 ($60).

```
LOOP     INY            Y is zero on entry
         CPY #$30
         EOR $C08C,X    Strobe card for nibbles
         BCC LOOP       Carry from CPY not EOR
         LDA $C08C,X    Now for real
         BPL GETBYTE
CKBYTE   CMP #$C9
         BNE DEATH
         LDA $C088,X    Turn off drive motor
         JMP $6000      From the top
GETBYTE  NOP
         NOP
         LDA $C08C,X    Last chance
         BMI CKBYTE     Maybe success
DEATH    EQU ∗
```

Track $13 Sector $05

($5600 Executable Code)

When the sector $04 code returns, the sector $03 code first performs some magic with location $CF. The correct answer is 78 and I have no idea why (see the assembly language file for track $13, sectors $03-05 for good guess).

The sector three code then loads sector $05 into $5600 and transfers there. The same old stuff awaits us. This time, the first instruction

is an ASL that enables a following ROR that puts it back to illegal. The same tricks of BNE/BEQ into the middle of instructions continue until we reach the EOR loop where we uncover a checksum-count routine that makes sure that the synchronized tracks routine has not been trifled with. If the routine checks out okay, location $1E is set to 93. Next we go back to the top where the code is hidden once more and we fall into the RTS (ho hum) back to the sector $03 code.

More Track $13 Sector $03

Program execution picks up with the following track $13, sector $03 code which sets up constants. This will be used by subsequent programs to validate the boot process.

```
LDA $1E   Contains $93
STA $97
EOR $CF
STA $EC   Contains $EB
```

The program then zaps both DOS patches by copying disjoint DOS routines over them. The protection scheme then zaps memory from $4C70 through $4CFE. Following this, it then transfers back to the restored instruction (restored as the first action of the track $13, sector $04 code) in the DOS cold start routine.

A little later the *Daisy* HELLO program will be executed and *DAISY:PARM0* and *DAISY:PARM1* will be loaded and *DAISY:PARM2* will be BRUN. As described in the very beginning, $4CFF will be entered.

The $4CFF routine will load track $02, sector $06 into $5600 and go there. This routine is an exact copy of the track $13, sector $05 code that performed a checksum-count of the synchronized tracks routine. The checksum-count, previously described, is then repeated. This is done to prevent one from defusing the code loaded from track $13, sectors $03 through $05. Assuming this second identical test passes, the protection scheme then returns to the $4CFF routine where we execute the following code:

```
LDA $CF   C($CF) = $78 (Magic #1)
EOR #$BD  A = $C5
CLC
ADC $1E   C($1E) = $93 (Magic #2)
STA $1B   A = $58
```

The value of location $1B is a function of the two magic numbers and the constant BD. What does it mean? I don't know. I think it is a confusion factor (mine, not their's), under the assumption you can never discover the contents of $CF and $1E which would not allow you to form the contents of $1B.

The program continues executing the following code:

```
         LDX $05    Number of pages to decode
AA       SEC
         LDA ($08),Y Here we are decoding the
```

```
SBC $CF        program DAISY:PARM1
EOR $1B
STA ($08),Y
INY
BNE AA
DEX
BPL AA
```

What this is doing is decoding *DAISY:PARM1* using two of the magic numbers from above. This is a different decoding technique than the one used twice before. This time, the EOR key remains constant. However, the value it works on is a function of the magic number in $CF. After it finishes, it copies page $03 over most of this code and then returns to BASIC.

Continual Copy-Protection Ploy

After a successful boot of *Daisy* is made, the protection scheme is continually active during program operation. This separate scheme is put into place during boot.

When run, *DAISY:PARM2* POKEs $3E71 (*DAISY:PARM1*) into the ampersand vector address of page zero. This address will be entered by any Applesoft program that has an ampersand (&) as a command. Each of the *DAISY:?* programs has an initial line of "& RESTORE" which transfers control to this ampersand vector. If you look at several of these programs, you will see the comment "restore variables." This is the method *Daisy* uses to pass variables between Applesoft programs.

On entry, the job of passing variables between Applesoft programs is addressed. Right in the middle, page zero location $1B is checked to see if it contains 58. If it does not, a routine is entered that will pull (PLA) the first byte of the return address off of the stack. Then the code continues. This means that when an exit is made from this routine, program execution begins at the wrong address and will run wildly and unpredictably through memory.

Continuing with the mind games, they check location $4F for non-zero in bits 7 through 4. If it is non-zero, they will not kill you this time. Rest assured, these bits are zero on an initial boot.

Flaws in Rainbow's Protection

Three major flaws were discovered during my deprotection efforts. These were bugs not found during Rainbow's testing of the protection scheme.

As you saw, they covered their tracks into the $4C70 code yet they never prevented me from messing with it. While messing around, I overwrote $4C70 code to see what the called routine returned in terms of page zero locations, etc., writing that sector back and booting the disk one more time.

Something prevented them from producing a real synchronized tracks protection scheme, which allowed almost any decent nibble-copy program to make a bootable copy.

Finally, they busied sectors $03, $04, and $05 of track $13 in the track sector map. If they would have left them idle, it would have taken me much longer to catch on.

Commentary & Observations

On Copy-Protection

The silly part of this entire exercise is that *Daisy* is a mathematics package that depends very much on the paper documentation that accompanies it. If the disk was copyable by any copy program and one gave it to every Tom, Dick, & Harry that came down the pike, they wouldn't know what to do with it.

The annoying part is that I could copy this program using the first two copy programs of choice, *Copy II Plus* and *Locksmith 5.0*. However, I still could not move it to a hard or a RAM disk where it would do me the most good. I think the message that needs to be transmitted to the providers/owners of software is that "thieves will always steal your wares while honest owners will suffer your copy protection.

The Softkey

As always, use a **copy** of the original disk or be prepared to light a candle in the name of the late, lamented original! In fact, use a copy made with your favorite copy program. The copy will not boot; however, that is not a problem.

1 Boot a DOS 3.3 System Master disk.

2 Remove the DOS 3.3 System Master and insert a blank disk.

INIT DAISY
DELETE DAISY

This is a slave disk. The DOS from a slave disk, when it is booted, goes directly to its normal location for a 48K Apple without screwing up the memory in between $9600 and $800.

3 Repeat steps one and two for another blank disk (use the back if you wish).

4 Remove the 2nd slave disk and reinsert the DOS 3.3 System Master.

BRUN MASTER CREATE

5 When the drive stops, put the second slave disk from above in the drive and follow the screen prompts to turn it into a master disk with a hello program called *DAISY*. Be sure you label these disks correctly as slave and master. The master will become the liberated *Daisy*, the slave will help us do our job.

From here on *master* will refer to this disk and not the DOS 3.3 system master.

6 Remove the master disk and put in the copy of *Daisy*.

LOAD DAISY
102 POKE 27,88: POKE 30,147 :POKE 207,120 :POKE 215,147: POKE 236,235
SAVE DAISY

When the disk is booted the POKE's put the magic numbers generated by the copy-protection scheme into memory for any subsequent programs that may wish to check for them.

7 Using a program, such as *Copy II Plus*, find the start address of *DAISY:PARM1* and it's length. My version loads at $3E71 and is $3A3 long (xx & yy).

8 Using a sector-editor, find A9 01 8D on track $13, sector $03. Change these bytes to 4C 41 9E. This causes the initial copy-protection steps to be bypassed.

9 Next find A2 00 8E and change these bytes to 4C 27 4D This jumps over the final checksum-check.

10 Find 10 F7 60 and change these bytes to 4C 59 FF. This will cause a jump to the monitor when the program *DAISY:PARM1* has been decoded. Finally, write the sector back to track $13, sector $03.

11 Boot the copied protected disk and wait for the monitor prompt (*). *DAISY:PARM1* is now decoded in memory.

12 Place the slave disk (NOTE: slave) in drive 1 and boot it with 6⌷P

13 When the drive stops, put the master disk in the drive.

BSAVE DAISY:PARM1, Axx, Lyy

Where xx and yy are the loading and length attributes obtained in step 7.

14 Copy all files except *DAISY:PARM1* to this disk from the protected disk. You are now done with the protected disk. Put it away in a safe place.

15 Use your sector-editor on the nearly liberated disk to find the byte sequence 6C 00 70 for *Daisy 2.2* and 4C FF 4C for *Daisy 2.4*. Change the first byte, (6C/4C) to 60 and write the sector back. The file is *DAISY:PARM2* if your sector-editor has file-following capabilities. You are done.

Boot your liberated disk

Moving *Daisy* to a Hard/RAM disk is left as an exercise to the reader, as is the replacement of normal DOS with a speed up DOS of your choice. Hint, *Diversi-DOS* almost always works on anything!

Review of the Copy Protection

There were six different copy protection techniques used to protect *Daisy*.

The combination synchronized track, nibble count routine is the primary protection scheme. The heart of this routine cannot be found using a sector editor since the code is "hidden" until the EOR loop uncovers it.

If you would somehow circumvent the nibble count routine the plan was for the second nibble count routine to catch on and reset in a totally different way from the SYNC tracks routine, thus confusing you even more.

If you remove all the code on track $13, sectors $03 through $05 and track $02 sector $06, the absence of the "magic" numbers in page zero locations would be detected by *DAISY:PARM2* which then branched to $C600 which causes the disk to reboot.

If you bypassed those checks, the program *DAISY:PARM1* would still be encoded (garbage) and would not run.

If you fixed the above problem (probably the way we did), *DAISY:PARM1* would discover the missing page zero magic number and you would boot once more. This time the cause of the boot would be very hard to discover since you would transfer wildly on exit.

Finally, the "magic" numbers generated during the boot process are probably checked for by crucial programs other than DAISY:PARM1 during program execution. This is why I took the chicken's way out and put them into the POKE's in the hello program.

All in all, because this protection used six different copy protection techniques, I would have to give this protection scheme a 10 (or zero, depending on where you stand on this issue).

Edward Teach

Attn: Bud Myers.

Since I submitted the last softkey for *Award Maker Plus*, I think I should be one of the first to comment on the letter from Bud Myers.

Bud, you have missed the whole point of COMPUTIST. The idea is the sharing of information, not simply deprotection.

For example, read the other articles on page 32 of COMPUTIST #52. What you will find is that the softkeys for *Color Print Shop* and *Empire I & II* did not work for everyone. What they are saying is while the specific patches did not work, try this different way.

Further, while my article looks like I was stressing that the other softkeys did not work, the emphasis was supplied by the RDEXed. Is it so unheard of that a software company might vary the protection a bit every 10,000 disks or so?

So Bud, I hope you keep reading and enjoy the mag for what it is.

Bud Myers

My apologies for my cranky letter regarding copy-protection removal which you published in RDEX for February. My thanks for Mr. Teach's technique published on the same page as my letter. It works like a charm! At last I can allow *Award Maker Plus* to circulate, secure in the knowledge that when (not if) something happens to the working copy, I can quickly provide another.

The reason I need to deprotect *Word Attack* is that, against my better judgment and all common sense, I allowed a teacher to sign out the master copy. She ruined it. She is willing to pay for a replacement, but unless and until I can back it up it seems futile to allow her to do so.

Could you ask your more advanced readers to provide the rest of us with more methods of backing up protected programs, whether the backup process removes the protection or not?

Mr. Meyer's letter arrived a few days before the influx of answering mail. Many of the replys were somewhat harsh. I thought that printing the Award Maker softkey right after the original letter was sufficient answer but many of you thought otherwise. Thanks Bud for shaking up our readers and reminding them to write. I hadn't heard from some of them in quite a while. A little dissention can do wonders for the old blood pressure...........RDEXed

---

Charles Taylor

Softkey for...

Green Globs & Graphing Equations
Sunburst

■ **Requirements**

☐ Super IOB
☐ Swap controller
☐ Initialized disk (no hello)

This can be deprotected by using the swap controller and *Super IOB*, copying tracks 3-22 to a disk initalized with *HELLO1*. A previous COMPUTIST softkey stated that the swap controller wouldn't work with 1986 Sunburst programs but it works with this one.

The swap controller is used so often maybe you should consider putting it on the inside of the front cover with Checkbin, etc.

1 Boot the disk and enter the monitor (reset or whatever).

2 Move the RWTS to a safe place.

```
1900<B800.BFFFM
```

3 Boot a slave disk and save the RWTS to your disk with COMPUTIST stuff.

```
C600G
BSAVE RWTS.SUNBURST, A$1900, L$800
```

4 Load and merge the swap controller with *Super IOB* then make sure that line 1010 of the swap controller reads TK=3 and not TK=0 and make sure that line 10010 BLOADs *RWTS.SUNBURST*.

5 Copy the program disk to a disk initialized with *HELLO*.

Bug in the Softkey for...

Masquerade
Phoenix Software

There is a bug in the *Masquerade* softkey (COMPUTIST #35, page 25) that I don't recall seeing a correction for:

shown as	should be
952:A5	952:A5
942:09	942:09
945:09	945:05

A few more comments on this crack: You can use *COPYA* to convert this disk to normal formatting for the first part of the softkey. (I had trouble with the *Super IOB* controller.)

1 Modify DOS to accept a D4 or a D5 in the address prologue and to ignore epilogues and checksums.

```
CALL -151
B954:4A C9 6A D0 EF
B942:18
```

2 Insert the disk with *COPYA* then load and modify *COPY.OBJ0* to stop copying after track $11.

```
BLOAD COPY.OBJ0
3A1:18
302:12
35F:12
```

3 Now we can load *COPYA* and run it.

```
LOAD COPYA
RUN 80
```

This modified version of *COPYA* will quit when it gets to track $12. Note that the changes in step 2 are to *COPY.OBJ0* in memory, so you have to load *COPYA* and run the line after 70 to keep *COPYA* from reloading *COPY.OBJ0* and undoing your work. Changes to $B954 and $B942 are to the memory image of DOS so be sure to reboot when you are through copying.

Softkey for...

Expedition Amazon
Penguin Software

Softkey for...

Coveted Mirror
Penguin Software

■ Requirements

☐ Two double sided blanks
☐ DOS 3.3 System master
☐ Apple II series
☐ Program disks
☐ *Copy II Plus* or some utility for changing boot programs and copying DOS.

These two excellent programs from Penguin are protected and deprotected by similiar schemes. The protection is the usual Penguin protection: the first byte of the address prologue alternates from D5 on even tracks to D4 on odd numbered tracks. This is easily defeated by using the method of Robert Muir in COMPUTIST #38.

Use *FID* from the system master to copy the files from the boot side of *Expedition Amazon* and use *COPYA* to copy the second side of *Expedition Amazon* and both sides of *Coveted Mirror*.

Coveted Mirror has a slight secondary protection. It normally runs *SETUPA* as the boot-up program. *SETUPA* then runs *SETUP*. We circumvent that by running *SETUP* as the boot-up program.

For... *Expedition Amazon*

[1] Boot the DOS 3.3 system master.

[2] Place a blank disk in the drive.

FP
INIT HELLO
CALL-151
B954:4A C9 6A D0 EF
B942:18
3D0G

[3] BRUN FID and copy all files from the boot side to the initialized disk.

[4] Run *COPYA* and copy the other side of the disk. Don't reboot.

For... *Coveted Mirror*

[1] Without rebooting, run *COPYA* and copy both sides. (For some reason *FID* would not make a working boot side.)

RUN COPYA

[2] Copy DOS from the System Master or your favorite fast DOS to the boot side of the copy.

---

[3] Change the boot-up program to *SETUP*. (I used *Copy II Plus* for steps 2 and 3.)

† *Softkey for...*

Boulderdash Construction Kit
Epyx

The protection on these two is identical to *Street Sports Baseball*, COMPUTIST #50, page 36.

A.P.T. for...

Expedition Amazon

Customize Your Expedition

One of the many advantages of unprotected software is that one can make changes to the program. Since *Expedition Amazon* is a lengthy and difficult game, I have developed some APTs.

The adventurers' progress is stored in two text files, *EXPA* and *EXPB*, depending on whether you are expedition A or B. Two other text files appear to be back-ups.(*EXPAEXPA* and *EXPBEXPB*) Accordingly, they can be loaded into a word-processor and changed.

The word-processor must be one that reads and writes text files. *Applewriter* will do just fine. The important thing is money and levels. Don't overdo it or the game will crash.

The table below will assist you in customizing your expedition.

Field Assistant	1	1 for Alive or 0 for Dead, I think
	9	Level (LVL)
	7	Energy Level (EL)
Medic	1	
	9	LVL
	8	EL
Radio Operator	1	
	9	LVL
	8	EL
Guard	1	
	9	LVL
	8	EL
	11	Hours
	0	Days
	1657	Treasure Value
	1	Sectors Mapped
	179117	Money
	229	Sq Meters Mapped
	0	
	8	Med Kit
	2	Tools
	2	Lamp
	1	Battery
	2	Rope

	4	Mosq Net
	1	Short Wave
	1	Loran
	1	Auto Rifle
	157	Bullets
	1	Grenades
	4	Food
	2	Boat
0000006000000		Sectors visible
0000006000000		(the first one is fully
0000000000000		mapped

Softkey for...

General Chemistry Disk #8
COMPress

■ Requirements

☐ *Super IOB 1.5*
☐ Apple II
☐ 1 blank disk side
☐ DOS 3.3 System Master

When I first got this program I went to my COMPUTIST Appleworks database and found a softkey in COMPUTIST #29. Unfortunately, the protection discussed in that softkey was completely different than the protection on my version.

After scanning a few tracks with the *Copy II Plus* nibble editor, I realized that the *Chemistry* disk had the same format as *Ultima IV*. I tried *Super IOB* with Mike Roetmans' *Ultima IV* controller (COMPUTIST #28, correction in COMPUTIST #32) and I was rewarded with a working copy without having to make sector edits.

[1] Boot the DOS 3.3 system master.

[2] Put a blank disk in the drive and type:

INIT HELLO

[3] Install this controller in *Super IOB 1.5* and run it.

Controller

```
1000 REM GENERAL CHEMISTRY #8
1010 TK = 3 : LT = 4 : ST = 15 : LS = 15 : CD = WR
1020 POKE 47405,24 : POKE 47406,96 : POKE
     47497,24 : POKE 47498,96
1030 POKE 47829,3 : T1 = TK : GOSUB 490 : GOSUB 210
1040 GOSUB 190 : GOSUB 610
1050 TK = TK + 1 : LT = LT + 1 : IF PEEK (BUF) < MB
     AND TK < 35 THEN 1040
1060 POKE 47405,208 : POKE 47406,19 : POKE
     47497,208 : POKE 47498,183 : POKE
     47829,213 : GOSUB 230
1070 TK = T1 : LT = 35 : GOSUB 490 : GOSUB 610 : IF
     PEEK (TRK) = LT THEN 1090
1080 TK = PEEK (TRK) : ST = PEEK (SCT) : LT = TK + 1 :
     GOTO 1020
1090 HOME : PRINT "COPYDONE,˚DOS˚NOT˚COPIED."
     : END
```

```
5000 DATA 213,170,181
5010 DATA 215,170,151
5020 DATA 213,170,150
5030 DATA 213,170,151
5040 DATA 215,170,150
5050 DATA 215,170,151
5060 DATA 221,170,158
5070 DATA 221,170,159
5080 DATA 213,170,181
5090 DATA 223,170,158
5100 DATA 223,170,159
5110 DATA 221,170,158
5120 DATA 221,170,159
5130 DATA 223,170,158
5140 DATA 223,170,159
5150 DATA 213,170,150
5160 DATA 213,170,181
5170 DATA 213,170,151
5180 DATA 215,170,150
5190 DATA 215,170,151
5200 DATA 213,170,150
5210 DATA 213,170,151
5220 DATA 215,170,150
5230 DATA 215,170,151
5240 DATA 213,170,181
5250 DATA 221,170,158
5260 DATA 221,170,159
5270 DATA 223,170,158
5280 DATA 223,170,159
5290 DATA 221,170,158
5300 DATA 221,170,159
5310 DATA 223,170,158
5320 DATA 213,170,181
5330 DATA 223,170,159
5340 DATA 245,170,182
5350 DATA 245,170,183
5360 DATA 247,170,182
```

Controller Checksums

1000	- $356B	5140	- $59D2
1010	- $3189	5150	- $8F2A
1020	- $C562	5160	- $AFA5
1030	- $545E	5170	- $8041
1040	- $DDB4	5180	- $0796
1050	- $A5C8	5190	- $1269
1060	- $044B	5200	- $95A5
1070	- $B732	5210	- $8E5D
1080	- $045C	5220	- $1DBE
1090	- $0EB7	5230	- $4CFD
5000	- $47E9	5240	- $EAF8
5010	- $C5B9	5250	- $3B74
5020	- $8750	5260	- $F464
5030	- $0BA3	5270	- $35E1
5040	- $0D61	5280	- $E4C4
5050	- $8BAD	5290	- $598C
5060	- $D0F6	5300	- $7A80
5070	- $397D	5310	- $67A1
5080	- $CC35	5320	- $2132
5090	- $96F7	5330	- $6F01
5100	- $4E87	5340	- $0EF1
5110	- $58CC	5350	- $9339
5120	- $A176	5360	- $B6AC
5130	- $0A8D		

Softkey for...

Zero-Gravity Pinball
Avant-Garde Creations

■ Requirements

☐ *MUFFIN* (from DOS 3.3 system master)
☐ Formatted disk with at least 173 free sectors
☐ Apple II (48K or more)

Zero-Gravity Pinball is a computer pinball game with a new twist — the ball will continue in the direction it is hit until it hits something else. There is no tendency to return to the bottom of the screen. Ten flippers along the sides of the screen are selected with a paddle control, one at a time. The ball is kept from going out the top and bottom of the screen by means of a force field (controlled by the keyboard and space bar.) If this sounds complicated, it's because it is.

At first glace the copy protection seems simple. It is 13-sector with the address and data epilogs changed from DE AA to DF AA. Copying this disk with any program that attempts to copy the whole disk will result in numerous read errors, however. This is because the sectors not containing data are formatted without data headers.

I decided that a modified *MUFFIN* was the way to go. I loaded *MUFFIN* and searched it for the byte DE, then changed DE to DF. This modified *MUFFIN* was now able to read the files on the *Zero-Gravity Pinball* disk and write to a standard formatted disk.

1 Boot DOS 3.3 system master and load *MUFFIN*.

BLOAD MUFFIN

2 Enter the monitor.

CALL-151
1A56:DF *read altered data epilog*
1AB2:DF *read altered address prolog*
803G *starts MUFFIN*

3 Copy all files to a formatted disk.

Softkey for...

Spy's Demise
Penguin Software

■ Requirements

☐ 1 blank disk
☐ DOS 3.3 system master with *FID*

Mr. Pollak's boot code trace for *Spy's Demise* was interesting, but it wouldn't work on my disk. I do thank Mr. Pollack for the information in his article (COMPUTIST #25, page 7).

Since *Spy's Demise* could be cataloged

after it was converted to standard DOS, I used *FID* to copy the files. Then I loaded two of the files into memory and continued with the code entry in Mr. Pollack's softkey.

1 Boot the system master.

2 Place a blank disk in the drive.
POKE 40514,52 *allows binary hello*
INIT SPY'S DEMISE

3 Place the system master in the drive.
CALL-151 *Enter the monitor*
B954:4A C9 6A D0 EF *read penguin DOS*
B942:18 *ignore epilog errors*
BRUN FID

4 Copy *TITLE.PIC* and *DEMISE* to the initialized disk then boot the system master disk.

5 Place the copy in the drive.
BLOAD TITLE.PIC
BLOAD DEMISE

6 From here on we use Mr. Pollak's softkey.
CALL-151
3F04:AD EC B7 8D
3F08:EE 80 AD ED B7 8D EF 80
3F10:AD 50 C0 AD 52 C0 AD 55
3F18:C0 AD 57 C0 4C 00 60
8E00:00
8E01<8E00.8EFEM
80D4:20 00 BD 60
BSAVE SPY'S DEMISE, A$3F04, L$4FFB

Softkey for...

Roadwar 2000
Strategic Simulations

Softkey for...

Wizards' Crown
Strategic Simulations

Softkey for...

Roadwar Europa
Strategic Simulations

Softkey for...

Kids on Keys
Spinnaker

■ Requirements

☐ *COPYA* (from DOS 3.3 system disk)
☐ 1 blank disk

Roadwar 2000 is sort of a combination sequel to the computer game *Autoduel* and the Mad Max movie series. You have to chase all over the country, scavanging for food, gas and

other supplies on your quest to find an antidote to the national epidemic.

The copy-protection is identical to the *Wizard's Crown* protection (alternating D5 and D4 in the address prolog and non-standard address epilogs. (See COMPUTIST #42.) Being too lazy to type in the *Wizard's Crown* controller, I used COPYA with a few DOS changes to deprotect this game.

```
  1  ] Boot your DOS 3.3 system master and
```
alter DOS.

Call-151

B954:4A C9 6A D0 EF *allow D4 or D5 in addr prolog*
B988:18 60 *ignore addr epilog errors*
3D0G
RUN COPYA

This procedure also copies *Wizard's Crown*, *Roadwar Europa*, and *Kids on Keys* by Spinnaker.

Softkey for...

Rendezvous
Peachtree

■ **Requirements**

□ Apple II
□ One blank disk side
□ COPYA
□ Program line editor (optional, but very useful)
□ A way into the monitor

The two previous softkeys for *Rendezvous* were not effective on my version, so I was forced to do my own deprotection. *Rendezvous 1.0* can be copied by COPYA and the files are listable from normal DOS.

The copy begins to boot, but the screen clears and the copy reboots after the rocketship comes on the screen. Since rebooting is usually caused by a JMP C600 somewhere on the disk, I scanned the disk with my *Copy II Plus* sector editor for the machine code 4C 00 C6 (JMP C600).

To my delight, the sector editor found the JMP C600 on track 1A, sector B, along with the following code:

```
1C75 A6 34    LDX 34
1C77 BD 8C C0 LDA C08C,X
1C7A 10 FB    BPL 1C77
1C7C C9 C3    CMP $C3
1C7E D0 F7    BNE 1C77
1C80 A9 00    LDA $00
1C82 85 00    STA 00
1C84 85 01    STA 01
1C86 BD 8C C0 LDA C08C,X
1C89 10 FB    BPL 1C86
1C8B C9 C3    CMP $C3
1C8D F0 14    BEQ 1CA3
1C8F E6 00    INC 00
```

```
1C91 D0 F3    BNE 1C86
1C93 E6 01    INC 01
1C95 D0 EF    BNE 1C86
1C97 4C 00 C6 JMP C600
1C9A EA       NOP
. . . .
1CA2 EA       NOP
1CA3 A5 01    LDA 01
1CA5 C5 36    CMP 36
1CA7 D0 EE    BNE 1C97
1CA9 A5 00    LDA 00
1CAB C5 37    CMP 37
1CAD 90 E8    BCC 1C97
1CAF C5 38    CMP 38
1CB1 B0 E4    BCS 1C97
1CB3 60       RTS
```

I thought all I had to do was replace the JMP C600 with a RTS (return) to have an unprotected copy. Wrong! It rebooted anyhow, and there was no other JMP C600 on the disk. I experimented with the code in this area to no avail. It was certainly the protection code, but I would have to get around it in another way.

The Catalog and Track-Sector map of *Copy II Plus* revealed that track 1A, sector B (containing the reboot code) was contained in the binary file "RGO", which loaded at location $1B00, and is $04F0 in length.

The basic program *RENDEZVOUS* BLOADs *R3DH*, BLOADs *RGO*, then CALLs 7168, which is a jump to $1C00. (#7168 = $1C00)

Eliminating this CALL solved the rebooting problem, but the program hung-up after the main menu. Apparently, the code at location $1C00 did something besides a disk check.

I studied the code beginning at $1C00 and found the following:

```
1C00 A2 00    LDX #$00
1C02 BD 00 1B LDA $1B00,X   load accum with
                            page $1B00
1C05 9D 00 0A STA $0A00,X   store it at page
                            $0A00
1C08 E8       INX
1C09 D0 F7    BNE $1C02
1C0B A9 A2    LDA #$A2      load accumulator
                            with A2
1C0D BD 00 08 STA $0800     store it at $0800
1C10 A9 70    LDA #$70      load accumulator
                            with 70
1C12 8D 01 08 STA $0801     store it at $0801
1C15 A9 B5    LDA #$B5      load accumulator
                            with B5
1C17 8D 02 08 STA $0802     store it at $0802
1C1A D0 01    BNE $1C1D
. . . .
1C2B D0 48    BNE $1C75     branch to disk
                            check
```

Note that the LDAs and STAs are changing some machine code around $800. Back to the catalog, the only program loading at $800 was *R3DH*. *R3DH*, therefore, was the file being

modified by the CALL 7168. After the disk check and the changes were made to *R3DH*, the menu runs. *R3DH* was called by CALL 2048 (#2048 = $800) in the *LIFTOFF* basic program (and others) after the main menu. Now it was clear that I had to get the modified *R3DH* onto my COPYA copy. Then the CALL 7168 in the file *RENDEZVOUS* could be eliminated, along with the disk check.

The rest was straight-forward. I booted my original disk. When the main menu came up, I reset into the monitor with my *Replay* card and BSAVEd *R3DH*, onto my COPYA copy. The *R3DH* that I saved was the modified *R3DH*, since I had allowed the program to boot past the disk check and *R3DH* changes, to the main menu. Then I loaded the *RENDEZVOUS* file and eliminated the CALL 7168 from line 2.

Lacking a means of resetting into the monitor, you might try making a nibble copy of *Rendezvous*, loading the basic program *EDU-WARE*, and removing the three pokes which prevent the normal reset. These are the POKES 1010, 1011, and 1012 in line 1. Then you can reset into the monitor with the reset key. I had some difficulty with I/O errors when doing this, however.

Cookbook style

```
  1  ] Use COPYA to make a copy of
```
Rendezvous.

```
  2  ] Unlock and delete R3DH from your
```
copy.

UNLOCK R3DH
DELETE R3DH

```
  3  ] Unlock and load Rendezvous.
```

UNLOCK RENDEZVOUS
LOAD RENDEZVOUS

```
  4  ] Edit line 2. Remove the CALL 7168.
```

```
  5  ] Save the modified RENDEZVOUS.
```

SAVE RENDEZVOUS

```
  6  ] Boot the original and reset into the
```
monitor at the main menu.

```
  7  ] Enter BASIC with:
```

3D0G

```
  8  ] Put your copy into the drive and:
```

BSAVE R3DH, A$800, L$12FF

The following are optional steps.

```
  9  ] If you haven't already done so, you
```
will probably want to remove the POKES 1010, 1011, and 1012 in *EDU-WARE* to allow a normal reset.

```
  10 ] While RENDEZVOUS is in memory
```
(step 4) you might as well delete the BLOAD RGO. It isn't used any more.

```
  11 ] You can also delete RGO during step
```
2 if you also perform step 10.

Jean Philippe Cunniet

DOS EOR MAKER

DOS EOR MAKER (a.k.a. DEM) is a short program used to protect data on disks. DEM will modify DOS 3.3 so that the RWTS will exclusive-OR (EOR) all data read from or written to the disk with any hex value you have chosen. It's a simple way to code data on disk for protection. The data in memory is the same as always but on the disk, it's very different. Catalog a DEM disk with normal DOS to see what I mean. The disk format is normal and can be copied with *COPYA* or any other normal disk copier. Only the data is encoded.

The heart of the coding routine is a EOR #$xx command where xx is a hex value (location $BCF9:xx) from $00 to $FF. A value of $00 is for normal DOS since EOR-ing with $00 doesn't change the original value. You could also modify the code into whatever complicated routine you want.

The main routine is put in DOS at location $BCDF-BCFF (normally unused). The RWTS is patched at $B7B7 to call $BCDF ($B7B7:20 DF BC instead of $B7B7:20 00 BD). Another patch is made to DOS at $B799 so that if you use the INIT command, the DOS won't be EORed when written to the disk ($B799:20 00 BD instead of $B799:20 B5 B7).

💥 *You may have some problems with these patches depending on the equipment in your system. Other programs sometimes take advantage of the "unused areas" in DOS to store their own routines. Some hard disks patch the jump code at $B7B7 and are adverse to sharing. Also, patching the INIT code to bypass the normal entry at $B7B5 may not be a good idea. While it is necessary to turn of the encoding during an INIT, calling the RWTS direct (instead of thru $B7B5) will not set the interupt disable. If any interupts are generated, they will interfere with the write routines which are timing dependent not to mention what terrible things may be happening to your hard drive if it didn't intercept the INIT call at $B7B5. A better method would be to remove the patch at $B7B7 for the INIT or another subroutine (in some other "unused area" of DOS) that sets the EOR byte to $00 and restores the original value after the INIT is completed. . . . . .RDEXed*

Just enter and save the BASIC program.

SAVE DOS EOR MAKER

When run the program will install the patch and ask what value you wish to use to encode data. When the program exits by printing ''DOS EOR READY'', all the patches will be in place.

Disks initialized now can be used as any DOS 3.3 disk but you can't CATALOG it from another DOS.

To transfer files from one DOS to another, you should:

1	Select the source DOS.
2	Load the file into memory.
3	Select the destination DOS.
4	Save the file on the destination DOS.

If your file is a 'A' file (BASIC), it's very easy. If it is a binary file (type 'B'), you have to know the starting address and the length of it before you save it. To find that, type, after loading the binary file:

CALL-151
AA60.AA73

Locations AA60.AA61 will give you its length (low byte first). Locations AA72.AA73 will give you its starting address (II). Then, select destination DOS and type:

BSAVE *filename*, **A$** *Address*, **L$** *Length*

If your file is a 'T' file, you will have to play with the READ and WRITE commands.

Of course, you could modify the *Super IOB* program so that the disk in Drive 1 is read with a normal DOS (POKE 48377,0) and the data is written in drive 2 encoded (POKE 48377,xx). Good luck protecting your data.

DOS EOR MAKER

```
10 TEXT : HOME : D$ = CHR$ (4) : INVERSE : PRINT
   "DOS-EOR-MAKER" : PRINT "JPH'CUNNIET" :
   NORMAL
20 FOR X = 1 TO 33 : READ A : POKE 48350 + X, A : NEXT
   : POKE 47032,223 : POKE 47033,188 : POKE
   47002,0 : POKE 47003,189
30 VTAB 12 : PRINT "ZERO'='NORMAL'DOS" : PRINT
   "(DEFAULT'VALUE=" PEEK (48377) ")" : VTAB
   11 : PRINT "EOR'VALUE'FOR'DOS'(1/255):" ; ;
   INPUT A$ : A = VAL (A$) : IF LEN (A$) < 1 THEN
   A = PEEK (48377)
40 IF A > 255 THEN 30
50 POKE 48377,A
60 PRINT : PRINT "DOS'EOR'READY!" : PRINT : PRINT
   "TYPE:'POKE'48377,0'FOR'NORMAL'DOS'OR" :
   PRINT : PRINT "POKE'48377," PEEK (48377)
   "'TO'RESTORE'ENCODING"
70 DATA 132, 72, 133, 73, 160, 8, 177, 72, 133,
   6, 200, 177, 72, 133, 7, 32, 244, 188, 32,
   4, 189, 160, 0, 177, 6, 73, 120, 145, 6, 200,
   208, 247, 96
```

Checksums

10	– $B8C3	50	– $ABC8
20	– $F716	60	– $2174
30	– $68B9	70	– $51E5
40	– $4C50		

RDEX END

COMPUTIST back issues

53 March 1988 ⬤ ■Softkeys: •2400 A.D. •Age Of

Adventure •Apple's Core II •Arcade Boot Camp •Arctic Fox •Aztec •Ballblazer •Bard's Tale IIgs •Blue Powder Gray Smoke •California Games •Championship Wrestling •Colonial Conquest •Comprehension Skills I.II •Conquering Whole Numbers •Coordinate Math •Countdown To Shutdown •Dataquest: The World Community •Destroyer •Dream House •Dream Zone •Earth Orbit Station •Equation Math •Forecast: Your At-Home Weather Station •Fraction Concepts Inc •Fraction Munchers •Fraction Practice Unlimited •GBA Championship Basketball •Genesis •GFL Football •Ghost Rider •Goonies •Grade Manager v2.3 •Great American Cross-country Road Race •Hardball IIgs •Ikari Warrior •Jenny's Journeys •Kid Niki Radical Ninja •Kung-Fu Master •Learning To Tell Time •Leisure Suit Larry •Let's Learn About Money •Let's Learn About The Library •Letters For You •Lords Of Conquest •Magic Spells •Math Blaster •Money Works •Maps & Globes: Latitude & Longitude •Marble Madness •Microzine 18,19,20,21,22,23 •Mist •Morning Star Math •Movie Monster Game •Multiplication Puzzles •Multiscribe v3.0c •Murder On The Mississippi •Music Made Easy •Mystery Sentences •Number Munchers •Numbers Count •Odell Lake •Operation Frog •Opposites Attract •Oregon Trail v1.4 •Phonics Prime Time: Blends & Digraphs •Phonics Prime Time: Vowels I, II •Puzzles & Posters •Quotient Quest •Reader Rabbit •Reading Style Inventory •Realm Of Impossibility •Sesame Street 'Crayon' series •Shanghai •Sons Of Liberty •Space Quest v2.2 •Story Book: Pixelworks •Story Tree •Subtraction Puzzles •Super Huey •Super Wordfind •Tass Times In Tonetown •Those Amazing Reading Machines III, IV •Timothy Leary's Mind Mirror •To Preserve, Protect and Defend •Tower Of Myraglen •Troll's 'MicroCoarseware' series •Webster: The Word Game •Word Munchers •Words At Work: Compound It •Words At Work: Suffix Sense •World Games •World's Greatest Baseball Game •World Karate Championship •Writer Rabbit •Zoyon Patrol ⬤ ■APTs: •Buck Rogers •Ikari Warrior •Kung-Fu Master •Leisure Suit Larry IIgs •Marble Madness •Realm Of Darkness •Rings Of Zilfin •Space Quest IIgs •Super Boulder Dash ⬤ ■Playing Tips: •2400 A.D. •Donkey Kong •Infiltrator •Space Quest IIgs •Spy Hunter •Swashbuckler •Thexder •Ultima II • ⬤ ■Features: •Modify Super IOB to read/write every other track •APT for Rings Of Zilfin: Turn yourself into a lean, mean fighting Machine •More Softkeys for M.E.C.C. software (1987) •How To Use The Electronic Art's RWTS •APT for Realms Of Darkness: Realm's Wrecker! •Putting Super Boulder Dash onto a hard disk IBM■ Softkeys: •EasyWriter 1.0, II •Zork III

52 February 1988 ⬤ ■Features: •The Product Monitor

•Unprotecting The Unprotectable: MacIntosh Softkeys! •A.P.T. Cornucopia •APT:Alternate Reality-Dungeon: Create A Super-human •Softkey for SSI's RDOS disks: I.ProDOS RDOS, 2.RDOS Transfer Utility •Making Cracked II Plus Disks Work On The IIc ⬤ ■Softkeys: •Apple Gradebook v2.6 •Award Maker Plus •Black Cauldron •Black Magic •California Games •Car Builder •Color Print Shop •Computer Ambush •Concepts In Science •Disney's Comic Strip Maker •Elite •Empire I, II •European Nations & Locations •Fooblitsky •Grid Designer •H.E.R.O. •Ikari Warriors •Infiltrator II •Le Francais par Ordinateur •Little Computer People's House on a Disk ★Main Street Filer •Master Diagnostics IIe ★MegaFiler ★MegaMerge •Microzine 23 •Might & Magic ★Millionaire •Mindplay software •Music Construction Set •Nibbler •Operation Market Garden •Phantasie ★Planetfall •PrintMaster Plus •Print Shop •Questron •Regatta •Ring Quest •Ringside Seat •Rings Of Zilfin •Shanghai •Silent Service •Snooper Troops •Spy's Adventure in N. America •Super Print •Tass Times In Tonetown •Think Quick ★Transylvania •Ultima I re-release •Where in the USA is Carmen Sandiego •World Games ★Zork I (★MacIntosh softkey) ⬤ ■A.P.T.s: Alternate Reality: The Dungeon •Arctic Fox •Bard's Tale II •Beyond Zork •Black Magic •Cavern Creatures •Drol •Goonies •Ikari Warriors •Zorro ⬤ ■Playing Tips: •Beyond Castle Wolfenstein •Championship Lode Runner •Conan •King's Queen II •Lode Runner •Lurking Horror •Station Fall •Ultima IV •Zork I BM■Softkeys: •Lotus 1-2-3 •Flight Simulator •PFS Report I BM A.P.T.S: •Bard's Tale

51 January 1988 ■Features: •The Crypt-arithmetic Helper •Using

EDD IV to Modify Tracks And Sectors •Bard's Tale APT: Dungeon Mapper Revisited •RAMfactor mod for Laser 128 •Ultima IV APT edit-tables •The Product Monitor •Get Better Sound by using the cassette jacks •Making A Fast Boot Disk •Might & Magic APT edit-tables ■Softkeys: •2400 AD •Aliens •Alphabet Zoo •Amnesia •Bag Of Tricks •Bard's Tale I •Bard's Tale II •Battle Cruiser •Beach-head II •Below The Root •Black Magic •Body Awareness •Bridge 4.0 •Carriers At War •Catalyst 3.0 •Centipede •Championship Boxing •Championship Wrestling •Chessmaster 2000 •Combining The Elements •Commando •Creative Contraptions •Einstein Compiler •Fat City •Fight Night •Flight Simulator v2.0 •Fun with Direction •GBA 2-On-2 Championship Basketball •GraphicWriter v1.1RA •Growing Up Small •House-on-a-disk •Intrigue •Jet •Jungle Hunt •Kindercomp •Knowing Numbers •Kung-fu Master •Law Of The West •Learning Well series •Letters And Words •Little Computer People •Make Your Own Murder Party •Manic Mansion •Master Diagnostics •Movie Maker •Music Construction Set •Pinball Construction Set •Pitstop •Print Shop Graphics Library Holiday •Print Shop IIgs •Rendezvous •Shapes And Patterns •Silent Service •Sorcerer •Spy vs Spy I & II •Stargate •Stellar 7 •Stickybear ABCs •Stickybear Drawing •Stickybear Numbers •Stickybear Printer •Stickybear Printer Library I & II •Stickybear Townbuilder •Super Boulderdash •Temple Of Apshai Trilogy •Tomahawk •Thexder •Walt Disney's Card And Party Shop •Walt Disney's Cartoon Maker •Wings Of Fury •Word Maze •World's Greatest Baseball Game •Zork III ■A.P.T.s: Bard's Tale •Lode Runner •Might & Magic •Ultima IV •W. Disney's Card And Party Shop •Wizardry III •Wizardry IV ■Playing Tips: •Autoduel •King's Quest •Manic Mansion •Summer Games •Tass Times In Tonetown •Thexder •Where In World is Carmen Sandiego?

50 December 1987 ■Features: •Super Boulderdash APT-writer

•Softkeys to Activision/ MECC/ and PFS ProDOS/ software •Double F-8 ROM space w/o motherboard surgery •Ace-Apple bimodal Switch •Using Sider hard drives 3½", 800K drives, & 5¼" drives in DOS 3.3 ■Softkeys: •Aliens •Alter Ego •Alternate Reality •Amazing Reading Machines •Amazon •American Challenge •Arcade Album #1 •Arithmetic Critters •Award Maker •Baseball Database •Bard's Tale II: Destiny Knight •BC's Quest for Tires •Bop & Wrestle •Champ. Boxing •Champ. Wrestling •Clock Works •Commando •Computer Prep for SAT •Conflict In Vietnam •Counting Critters •Crisis Mountain •Dataquest 50 States •Deluxe Paint II •Dino Eggs •Disney Card & Party Shop •Disney Comic Strip Maker •Draw Plus •Eidolon •Electric Crayon ABCs •Expedition Amazon •Facemaker •First Letter Fun •Fish Scales •Fun From A—Z •Game Maker •GBA Champ. Basketball •GFL Champ. Football •Graphicwriter 1.0R/1.1R •Great Road Race •Hacker II •Hardball •Infiltrator II •Instant Music •James Bond 007: A View To A Kill •Jenny's Journeys •Kung Fu Master •Little People •List Handler •Manic Mansion •Mastery Arithmetic Games •Market Place •Master of Lamp •Math Rabbit •Microzine #17 •Might and Magic •Mission In Solar System •Moebius •Music Construction Set •Music Studio •Number Munchers •Paint With Words •Paintworks Plus •Path Tactics •pfs:File •pfs:Graph •pfs:Plan •pfs:Report •pfs:Write •Phonics Prime Time •Portal •Principal's Assistant •Print Shop ProDOS 8 v1.4 •Print Shop Holiday Edition •Quickflash! •Reader Rabbit •Realm of Impossibility •Robot Odyssey I v.2.0 •Rocky Horror Show •Rocky's Boots v4.0 •Saracen •Shanghai •Silent Service •Skylab •Sound Tracks •Speedy Math •Spindizzy •Street Sports Baseball •Sub-Mission •Super Boulderdash •Tass Times in Tonetown •Thexder •Top Fuel Eliminator •Word Handler •Word Munchers •Words at Work •World Karate Champ. •Writer's Choice: Elite •Zardax v5.2.1

49 November 1987 ■Features: •Eliminate some ProDOS erroneous

error messages •Date/time without a clock card •Sector surgery: recover lost files •Generating Applesoft programs 'on-the-fly' •Product Monitor reviews •PLUS: How to convert List Handler files into standard text files •How to make GRAPHIC.GRABBERv3 run on the IIgs •Laser 128 'absolute' RESET ■Playing Tips for: Bard's Tale II, Conan, Donkey Kong, Hacker I, Hard Hat Mack, Orbitron, Print Shop Companion, Spellbreaker, Spy Hunter, Ultima 4 ■A.P.T.s for: Infiltrator, Lode Runner, Montezuma's Revenge, Swordthrust series ■RDEX Softkeys: •Addition Logician •Animate •Arcade Boot Camp •Arctic Fox •Bard's Tale II •Cat'n Mouse •Counting Critters •Dam Busters † •Destroyer •Draw Plus v1.0 •Dr. Ruth's Comp. Game Of Good Sex •Echo 1.0 •E.D.D. 4 •Gamemaker •Hard Ball •Infiltrator •List Handler † •Locksmith 6.0 Fastcopy † •Magic Slate •Math Critters •Millionaire •Mind Mirror •One On One •Paintworks Plus v1.0 •Paintworks Plus v1.1 •PHM Pegasus •Portal •Quotient Quest •Reader Rabbit •Saunder's Chemistry CAI •Science Toolkit •Shanghai •Strip Poker † •Super Bunny •Super Sunday •Swordthrust series † •Term Paper Writer •Thief •Top Fuel Eliminator •Typing! † •Up-n-Down •Willy Byte •Writer's Choice - Elite v1.0 •Writing A Character Sketch •Writing A Narative

MORE back issues

48 October 1987 ∎*Features:* •Dungeon Editor & Encounter Editor for Ultima III •APT for Shadowkeep •Softkey for Shadowkeep •Softkey for Apple Business Graphics ∎*RDEX Softkeys:* •816 Paint GS •Amnesia •Arctic Fox •Award Maker Plus •Bard's Tale II •Betterworking Word Processor •Beyond Castle Wolfenstein •Black Magic •Bookends Extended •Bop & Wrestle •Chess 7.0 •Chessmaster 2000 •Deluxe Paint GS •Destroyer •Hacker II •Hacker II GS •Hardball •Infiltrator •Instant Music GS •J-Bird •Mabel's Mansion •Marble Madness •Mean 18 GS Golf •Megabots •Might & Magic •Miner 2049er II •Mouse Word •Music Construction Set GS •Music Studio GS •New Oregon Trail •Paintworks Plus 1.0 GS •Paintworks Plus 1.0 GS •Paul Whitehead Teaches Chess •PHM Pegasus •Poetry Express •Print Shop color version •Rambo: First Blood part II •Rocky Horror Show •Sargon III* •Shanghai GS •Spindizzy •TelePorter •Temple Of Apshai trilogy •Top Draw GS •Transylvania •Ultima I •World's Greatest Baseball Game

47 September 1987 ∎*Features:* •Infocom-text Reader Enhancement •Color Ultimapper mod to Ultimapper IV •Towne Mapper utility for Ultima IV •Dungeon Mapper utility for Bard's Tale •Hardware Corner: Interrupting Your Apple •Softkey for Charlie Brown's 1,2,3s ∎*RDEX Softkeys:* •Guitar Wizard •Gemstone Warrior •Notable Phantom •Micro Wine Companion •Stickybear Printer •Note Card Maker •Starcross •Wishbringer •Dinosaur Dig •Dam Busters •Pirate Adventure •Infiltrator •MECC software •Banner Catch •Turtle Tracks •PFS File •Microzine #12, #13, #14 •Marble Madness •Writer Rabbit •Arcticfox •Age Of Adventure •Might And Magic •Space Station •Alternate Reality •Mindshadow •Gemstone Warrior •Strip Poker •Lucifer's Realm •Manuscript Manager •Bank Street Writer III •Kids On Keys •The Missing Ring •Graphic Solution •Empire I, II •Champ. Golf

46 August 1987 ∎*Softkeys* •Advanced Microsystems Technology programs •Word Attack •Star Blazer •Science Toolkit •The Color Enhanced Print Shop •Video Vegas •The Handlers •K.C. Deals On Wheels •Law Of The West •Break The Bank Blackjack •Foundation Course In Spanish •OGRE •Puzzles And Posters ∎*Features* •The Shift Key/Lower Case Option For II+ •Amazing Computer Facts •Shape Magic utility ∎*Review:* Multiscribe

45 July 1987 ∎*Softkeys* •Mouse Calc •Sands of Egypt •Number Farm •Agent U.S.A. •Wavy Navy •Kindercomp •Flight Simulator Update •Raid over Moscow •Crime Stopper •Key Perfect 5. •The Final Conflict •Miss Mouse •Snoggle ∎*Features* •Write Protecting the Microsoft RAM Card •Keys to Success on the Franklin Ace •Modified F8 ROMs on the Apple /// ∎*Core* •Owner's Review of Copy Master II

44 June 1987 ∎*Softkeys* •Arcade Boot Camp •Goonies •Zorro •Coveted Mirror •Crimson Crown •Compubridge •Fleet System 3 •Microwave •Escape •Catalyst 3.0 •Number Farm •Alphabet Circus •Joe Theisman's Pro Football •Black Cauldron •Intern. Gran Prix ∎*Features* •Making DOSless Utilities •Pixit Printer Drivers ∎*Review:* Z-RAM Memory Expansion Board •Reading the Joystick

43 May 1987 ∎*Softkeys* •Graphics Expander •Information Master •Certificate Maker •Elite •Catalyst 2.0 and 3.0 •Murder On The Mississippi •Temple Of Apshai Trilogy •Troll Associates programs •Spell It •Regatta •Cdex Training programs •Think Fast ∎*Features* •How to Write-Protect your Slot Zero •Capturing Locksmith 6.0 Fast Copy •Revisiting DOS to ProDOS and Back ∎*Core* •Computer Eyes / 2: a Review ∎*APTs* •Sword of Kadash & Rescue Raiders •Ultimaker IV

42 April 1987 ∎*Softkeys* •Light Simulator •Beach-Head •Monty Plays Scrabble •Racter •Winnie the Pooh •Infocom Stuff, Kabul Spy, Prisoner II •Wizardry 1 & 2 •Lucifer's Realm •The PFS Series •Dollars and Sense •Strip Poker •Coveted Mirror •Wizard's Crown •The Swordthrust Series •Axis Assassin •Manuscript Manager •The Crown of Arthain •Address Book •Decimals 3.0 •Dragonfire ∎*Features* •Auto Duel Editor •Wizard's Crown Editor •Questron Mapper ∎*Core* •The Games of 1986 in Review ∎*Adventure Tips* •Ultima IV

41 March 1987 ∎*Softkeys* •The Periodic Table •Gemstone Warrior •Inferno •Frogger •Story Maker •Adventure Writer •Mummy's Curse •Zaxxon •The Quest •Pitfall II •H.E.R.O. ∎*Features* •A Two-Drive Patch for Winter Games •Customizing the Speed of a Duodisk •Roll the Presses Part Two: Printshop Printer Drivers •The Games of 1986

40 February 1987 ∎*Softkeys* •Adventure Writer •E-Z Learner •Mychess II •Raster Blaster •Cranston Manor •Ghostbusters •Designer's Pencil •The American Challenge •Encyclopedia Britannica Programs •Crime Wave ∎*Features* •Taking the Wiz out of Wizardry •Adding a Printer Card Driver to Newsroom ∎*Core* Games of 1986

39 January 1987 ∎*Softkeys* •MIDI/8+ •Homeword v2.1 •Borrowed Time •Amazon •Speed Reader][•Discovery! •M-ss-ng L-nks series •Donald Ducks's Playground •Mastering the SAT •Copy][Plus 4.4C •Master of the Lamps •One on One •Bridge Baron •A.E. •Great American Cross-Country Road Race •Computer Preparation for the SAT •Castle Wolfenstein •Luscher Profile •Skyfox •Silent Service •Echo Plus •Swashbuckler •Randamn ∎*Features* •Electronic Disk Drive Swapper •Abusing the Epilogues •Print Shop Companion's Driver Game ∎*Core* •Keyboard Repair •Fixing the Applesoft Sample Disk

38 December 1986 ∎*Softkeys* •Cyclod •Alternate Realty •Boulder Dash I & II •Hard Hat Mack (Revisited) •The Other Side •F-15 Strike Eagle •Championship Lode Runner •Gato V 1.3 •I, Damiano •Wilderness •Golf's Best ∎*Features* •The Enhanced/ Unenhanced //e •Looking into Flight Simulator's DOS ∎*Core* •Appavarex •Installing a RAM disk into DOS 3.3

37 November 1986 ∎*Softkeys* •Under Fire •Pegasus][•Take 1 (revisited) •Flight Simulator II v1.05 (part 2) •Magic Slate •Alter Ego •Rendezvous •Quicken •Story Tree •Assembly Language Tutor •Avalon Hill games •Dark Crystal ∎*Features* •Playing Karateka on a //c •Track Finder •Sylk to Dif ∎*Core* •Breaking In: tips for beginners •Copy][Plus 6.0: a review •The DOS Alterer

36 October 1986 ∎*Softkeys* •Flight Simulator II v 1.05 •AutoDuel •Critical Reading •Troll's Tale •Robot War •General Manager •Plasmania •Telarium Software •Kidwriter v1.0 •Color Me ∎*Features* •ScreenWriter meets Flashcard •The Bus Monitor •Mousepaint for non-Apples ∎*Core* •The Bard's Dressing Room ∎*APT* •Championship Lode Runner

35 September 1986 ∎*Softkeys* •Olympic Decathlon •Hi-res Cribbage •Revisiting F-15 Strike Eagle •Masquerade •The Hobbit •Pooyan •The Perfect Score •Alice in Wonderland •The Money Manager •Good Thinking •Rescue Raiders ∎*Feature:* Putting a New F8 on Your Language Card ∎*Core:* •Exploring ProDOS by installing a CPS Clock Driver

still more back issues

To order, Use the back order form on page 42

34 **August 1986** ■*Softkeys* •Crisis Mountain •Terripin Logo •Apple Logo II •Fishies 1.0 •SpellWorks •Gumball •Rescue at Rigel •Crazey Mazey •Conan •Perry Mason: The Case of the Mandarin Murder •Koronis Rift ■*Feature:* •More ROM Running ■*Core:* •Infocom Revealed

33 **July 1986** ■*Softkeys* •Word Juggler •Tink! Tonk! •Sundog v2.0 •G.I. Joe & Lucas Film's Eidolon •Summer Games II •Thief •Instant Pascal •World's Greatest Football Game •Graphic Adventure #1 •Sensible Grammar & Extended Bookends •Chipwits •Hardball •King's Quest II •The World's Greatest Baseball Game ■*Feature:* •How to be the Sound Master ■*Core:* •The Mapping of Ultima IV

32 **June 1986** ■*Softkeys* •Revisiting Music Construction Set •Cubit •Baudville Software •Hartley Software •Bridge •Early Games for Young Children •Tawala's Last Redoubt •Print Shop Companion •Kracking Vol II •Moebius •Mouse Budget, Mouse Word & Mouse Desk •Adventure Construction Set ■*Feature:* •Using Data Disks With Microzines ■*Core:* •Super IOB v1.5 a Reprint

31 **May 1986** ■*Softkeys* •Trivia Fever •The Original Boston Computer Diet •Lifesaver •Synergistic Software •Blazing Paddles •Zardax •Time Zone •Tycoon •Earthly Delights •Jingle Disk •Crystal Caverns •Karate Champ ■*Feature:* •A Little Help With The Bard's Tale ■*Core:* •Black Box •Unrestricted Ampersand

30 **April 1986** *Softkeys* • Millionaire • SSI's RDOS • Fantavision • Spy vs. Spy • Dragonworld • King's Quest • Mastering the SAT • Easy as ABC • Space Shuttle • The Factory • Visidex 1.1E • Sherlock Holmes • The Bards Tale • *Feature* • Increasing Your Disk Capacity • *Core* • Ultimaker IV, an Ultima IV Character Editor

29 **March 1986** *Softkeys* • Threshold • Checkers v2.1 • Microtype • Gen. & Organic Chemistry Series • Uptown Trivia • Murder by the Dozen • Windham's Classics • Batter Up • Evelyn Wood's Dynamic Reader • Jenny of the Prairie • Learn About Sounds in Reading • Winter Games • *Feature* • Customizing the Monitor by Adding 65C02 Disassembly • *Core* • The Animator

28 **February 1986** *Softkeys* • Ultima IV • Robot Odyssey • Rendezvous • Word Attack & Classmate • Three from Mindscape • Alphabet Keyboarding • Hacker • Disk Director • Lode Runner • MIDI/4 • Algebra Series • Time is Money • Pitstop II • Apventure to Atlantis • *Feature* • Capturing the Hidden Archon Editor • *Core* • Fingerprint Plus: A Review • Beneath Beyond Castle Wolfenstein (part 2)

27 **January 1986** *Softkeys* • Microzines 1-5 • Microzines 7-9 | Microzines (alternate method) • Phi Beta Filer • Sword of Kadash • Another Miner 2049er • Learning With Fuzzywomp • Bookends • Apple Logo II • Murder on the Zinderneuf • *Features* • Daleks: Exploring Artificial Intelligence • Making 32K or 16K Slave Disks • *Core* • The Games of 1985: part II

26 **Softkeys** • Cannonball Blitz • Instant Recall • Gessler Spanish Software • More Stickybears • Financial Cookbook • Super Zaxxon • Wizardry • Preschool Fun • Holy Grail • Inca • 128K Zaxxon • *Feature* • ProEdit • *Core* • Games of 1985 part I

25 *Softkeys* • DB Master 4.2 • Business Writer • Barron's Computer SAT • Take 1 • Bank Street Speller • Where In The World Is Carmen Sandiego • Bank Street Writer 128K • Word Challenge • Spy's Demise • Mind Prober • BC's Quest For Tires • Early Games • Homeword Speller • *Feature* • Adding IF THEN ELSE To Applesoft • *Core* • DOS To ProDOS And Back

24 *Softkeys* • Electronic Arts software • Grolier software • Xyphus • F-15 Strike Eagle • Injured Engine • Mr. Robot And His Robot Factory • Applecillin II • Alphabet Zoo • Fathoms 40 • Story Maker • Early Games Matchmaker • Robots Of Dawn • *Feature* • Essential Data Duplicator copy parms • *Core* •DOS-Direct Sector Access

23 *Softkeys* • Choplifter • Mufplot • Flashcalc • Karateka • Newsroom • E-Z Draw • Gato • Dino Eggs • Pinball Construction Set • TAC • The Print Shop: Graphics Library • Death In The Caribbean • *Features* • Using A.R.D. To Softkey Mars Cars • How To Be The Writemaster • *Core* • Wheel Of Money

22 *Softkeys* • Miner 2049er • Lode Runner • A2-PB1 Pinball • The Heist • Old Ironsides • Grandma's House • In Search of the Most Amazing Thing • Morloc's Tower • Marauder • Sargon III • *Features* • Customized Drive Speed Control • Super IOB version 1.5 • *Core* • The Macro System

20 *Softkeys* • Sargon III • Wizardry: Proving Grounds of the Mad Overlord and Knight of Diamonds • The Report Card V1.1 • Kidwriter • *Feature* • Apple][Boot ROM Disassembly • *Core* • The Graphic Grabber v3.0 • Copy II+ 5.0: A Review • The Know-Drive: A Hardware Evaluation • An Improved BASIC/Binary Combo

19 *Softkeys* • Rendezvous With Rama • Peachtree's Back To Basics Accounting System • HSD Statistics Series • Arithmetickle • Arithmekicks and Early Games for Children • *Features* • Double Your ROM Space • Towards a Better F8 ROM • The Nibbler: A Utility Program to Examine Raw Nibbles From Disk • *Core* • The Games of 1984: In Review-part II

16 *Softkeys* • Sensible Speller for ProDOS • Sideways • Rescue Raiders • Sheila • Basic Building Blocks • Artsci Programs • Crossfire • *Feature* • Secret Weapon: RAMcard • *Core* • The Controller Writer • A Fix For The Beyond Castle Wolfenstein Softkey • The Lone Catalog Arranger Part 1

1 *Softkeys* • Data Reporter • Multiplan • Zork • *Features* • PARMS for Copy II Plus • No More Bugs • APT's for Choplifter & Cannonball Blitz • 'Copycard' Reviews • Replay • Crackshot • Snapshot • Wildcard

If you are also interested in obtaining out-of-print back-issues

Please see the descriptions listed on page 41

40   COMPUTIST 54   April

and Out-of-print back issues

To order, Use the back issue order form on page 42

New IBM COMPUTIST?

In order to make IBM RDEX easier to read and understand, the following formats will be used.

Commands you type are bold. Long lines tend to wrap down like this

But *don't type these comments*

Remember, press the [ENTER] key after every command line. If something important in shown on the monitor after a command,

it will be shown like this

DEBUG is usually invoked in softkeys. It has several powerful single-letter commands listed below:

    Assemble   *address*
    Compare   *range/address*
    Dump   *range/address*
    Enter   *address   list*
    Fill   *range   list*
    Go *address*
    Hexarithmetic *value   value*
    Input from *port-address*
    Load file   *address drive sector sector...*
    Move memory *range address*
    Name of file *drive path filename.ext*
    Output control *port-address*
    Proceed   *address value*
    Quit debug
    Register   *register-name*
    Search   *range   list*
    Trace   *address   value*
    Unassemble address/range
    Write   *address   drive sector sector...*

Mike Basford

Some of these are from BBS...

Putting Sargon III on harddisk

Here's how XT-owners can transfer *Sargon III* to hard-disk (assume *Sargon* Disk is in A:, and default drive is C:)

C:
COPY A: SARGON.III SARGON3.COM *Original*
    SARGON3.COM is ignored
DEBUG
N SARGON3.COM *file name*
L *load new SARGON3.COM*
A 7F9F *Assemble the following:*
MOV AH,1,0
[ENTER]
A 83E1 *Assemble the following:*
MOV AH,2
[ENTER]
W *write*
Q *quit DEBUG*
COPY A:B000.BOK

B000.BOK contains the book openings. None of the other files are required and should not be copied to fixed disk. *Sargon* should function normally.

(?) Unfortunately, when *Sargon* exits, it reboots the system. Can anyone find an easy change for that one.

Softkey for...

Time Manager
IBM

■ **Requirements**

☐ *DEBUG*
☐ *Time Manager* disk (80-column version)
☐ floppy drive address and a destination drive address (may be a hard disk or floppy)

Place a disk with *DEBUG* in drive C: and *Time Manager* disk in floppy drive A:

DEBUG *don't specify a filename yet*
L 600 0 A5 40 *drive A=0, drive B=1, etc.*
F DS:100 L 500 90 *pad 100—600 with NOPs*
R CX *and at the prompt (:), type:*
600 *(size)*
N x:TM.COM *x can be any drive you want to contain the unprotected module.*
W *write the file*
Q *and quit DEBUG*

That's it. You can now run the new *TM.COM* without the floppy, but you will still need the data disk and TM will do physical drive selection on the data disk. For example, on my Davong system, floppy drive 2 is logically addressed as D: but to TM it is B:.

Softkey for...

R:base 400 v1.11
---

[1] Format a System Disk under DOS 2.0 OR 2.1 and label it according to the original *R:BASE* diskette.

[2] Copy the files from the original diskette to the corresponding DOS 2.x formatted diskette and put the original *R:BASE* diskette back in the manual.

[3] Follow this debug 'script':

RENAME RBASE.EXE RBASE.DAT
DEBUG *invoked without filename, please*
N RBASE.DAT *File name*

Assuming you have a 128K + machine...

L 1000:0 *Load*

(S) Search for these strings and (E) change them as shown:

S 2000:0 L EFFF 72 10 E8 54 01 72 0B
E 2000 22AB 90 90 E8 54 01 90 90

S 2000:0 L EFFF 74 01 F9 1F C3
E 2000:240E 74 01 F8

S 2000:0 L EFFF B4 02 CD 13
E 2000:22EB B4 04 90 F9

S 2000:0 L EFFF 8E C0 BE 78 00
E 2000:230C 90 90 90 90 90 90 90 90
**   90 90** *eleven 90s*

S 2000:0 L EFFF 52 B4 42 E8 C1 00
E 2000:231D 26 C7 06 00 09 50 04 26 C7
**   06 02 09 E8 0F 26 C7 06 04 09 00 01**
**   26 C7 06 09 02 15 C3**

W 1000:0 *write it*
Q *quit DEBUG*

RENAME RBASE.DAT RBASE.EXE

Note: Now all of the *R:BASE* copy-protection is removed, and *DISKCOPY*-able. All the checks (for specially-formatted tracks) have been completely removed.

You may load all the files on the newly-formatted and unprotected diskette directly to hard-disk or RAM-disk, in any sub-directory you set up

Softkey for...

Prokey v3.0

Hard-disk drive is assumed as B:

[1] Write-protect (tab) the *ProKey* Distribution disk and put it in B: , and a DOS disk in floppy drive A:

b: *make b: the default drive*
A:DEBUG PKLOAD.COM *invoke DEBUG*
G 3FFC

    AX=0100   BX=3FE1   CX=0000   DX=03F4
    SP=FFFE   BP=41CA   SI=0920   DI=0110
    DS=0B4F   ES=0B4F   SS=0B4F   CS=0B4F   IP=3FFC
    NV   UP   EI   PL   NZ   NA   PO   NC

R CX *check register CX*
    CX 0000 *arid at the prompt (:) type:*

40CA

Now change disks in drive b: Use a formatted blank disk.

W
    WRITING 40 CA BYTES

Q *quit DEBUG*

ProKey is now unprotected. You may now manually copy all remaining files from the ProKey distribution disk onto your fresh disk in B:. The resulting *PKLOAD.COM* will start normally, and may even be copied onto a hard disk and run from there without the bother of putting the distribution disk into one of the floppy drives!!!

LAYOUT.COM is protected in the same way as *PKLOAD.COM*. To 'adjust' *LAYOUT.COM*, use the same procedure as for *PKLOAD* with these exceptions:

Change A:DEBUG PKLOAD.COM
    to A:DEBUG LAYOUT.COM

Change G 3FFC to G 1F8F

Change 40ca (*Setting CX*) to 205d
That's it!

David G. Alexander

Introduction to

IBM Disk Format, Access and Copy Protection

The following discussion will focus on the 360K, 5¼" floppy which is the most common format.

Both sides of the disk are formatted and designated as sides Ø and 1. Each side has 40 tracks of 9 sectors each. Each sector contains 512 bytes of data.

Sector ID field

A sector has a sector identification (ID) field composed of A1 A1 A1 FE, four bytes (designating the track, side, sector number and size) and two CRC bytes. The size-ID byte for a 512-byte sector is 02.

Data field

Following the sector-ID field is a data field composed of A1 A1 A1 FB, 512 bytes of data and two CRC bytes.

The sectors are separated by 00s (sync-bytes) and 4Es (gap-bytes). There is no nibblization scheme, and ASCII text appears normally in a raw track-dump.

Each disk has a system area on side Ø, track Ø which has a one-sector boot record, a two-sector file-allocation table and at least one root directory. The files follow the root directory.

Disk Access

The IBM is an interrupt-driven machine, and disk access is performed by means of interrupt number $13. (In the Apple system we scan a disk looking for $C08C,X disk-access codes) In the IBM system we look for CD 13 codes which unassemble to INT 13.

The interrupt uses the AH-register to select the function to be used and performs a function comparable to an RWTS call in the Apple system.

Here is a chart describing how the interrupt is used.

Table 1

Registers	Interrupt $13 codes
AH =	00 Reset Disk
AH =	01 Read Status of Disk System to:
AL =	Return Code explanation
00	Successful
01	Bad Command Given to INT
02	*Address Mark Not Found
03	Write-attempt on write-protected disk
04	*Requested Sector Not Found
08	DMA Overrun
09	Attempt to Cross DMA Boundary
10	*Bad CRC on Disk Read
20	Controller Failure
40	Seek Operation Failure
80	Attachment Failure

* indicates most used in copy-protection

AH =	02 Read Sectors
Input	DL = Drive Number (0-3)
	DH = Head (side) number (0-1)
	CH = Track Number
	CL = Sector Number
	AL = Number of Sectors to Read
	ES:BX = Load Address
Output	
AH =	Return code (carry flag set)
AL =	Number of Sectors Read
AH =	3 Write (same params as above)
AH =	4 Verify (same params as above)
AH =	5 Format (same params as above; ES:BX points to format table)

Protection Schemes

IBM copy-protection is generally simpler than Apple protection, with no quarter tracks and other exotic format alterations.

The IBM equivalent to Apple's *COPYA* copy program is called *DISKCOPY*, and any non-standard sector format will stop it cold.

Anything can be altered, such as sector numbers, sector order, sector size, deliberately incorrect CRC bytes, etc.

The general copy-protection scheme is to provide at least one non-standard, preferably difficult-to-reproduce sector which will foil *DISKCOPY* and have a signature check in the program to perform an INT 13 operation on the altered sector. Whether or not the disk check passes will depend on the return code.

A very simple scheme would be to provide an altered sector which will generate a CRC error (carry flag set) when accessed. The code would look something like this.

```
INT 13      Read the sector
JC  1B00    Continue program upon carry set
            (protected sector found)
INT 19      Reboot
```

This simple check can be disabled by changing the JC 1B00 to JMP 1B00 so that the

program will continue on to the code at $1B00 regardless of the outcome of the read operation.

Self-modifying code

However, many programs use self-modifying code to hide the CD 13s in the disk-check routines, and it can take many hours of single-stepping through program code to unravel the protection. A typical example would be as follows.

```
1B00:0000 31DB    XOR BX,BX
1B00:0002 8EDB    MOV DS,BX
1B00:0004 BB0DD0  MOV BX,000D
1B00:0007 8A07    MOV AL,[BX]
1B00:0009 3412    XOR AL,12
1B00:000B 8807    MOV [BX],AL
1B00:000D         FIRST WORD
```

In this section of code, [AL] is set to DF at location 1B00:0007. When you eXclusive-OR DF and 12 you get a CD (for the INT opcode) which is placed right next to a 13 at 1B00:000D to produce a **CD 13**!

Altered sectors can be identified using *Locksmith PC*, Central Point Software's *PC Option Board*, etc. *Norton Utilities* can be used to scan a disk or file for CD 13 codes which are not encoded.

Programs including *PC WATCH*, *TRAP13* and *INT13* are resident "front end" programs which intercept ROM BIOS interrupt-13 calls to perform disk functions on specific sectors. The programs print out the requested disk drive, side, track, sector, number of sectors, and operation type contained in the registers listed earlier. After the interrupt processing is completed, the program prints the status code returned (which indicates whether an error has occurred), and the address in the calling program to which control will be returned.

Once the questionable INT 13 calls are found, the next step is to Unassemble the surrounding code using *DEBUG* and figure out how the routine works.

Another way to decode the INT 13 calls is to use *DEBUG*'s GO command. Set a breakpoint at the return address given by the front end program. When *DEBUG* stops, you will have decoded not only the INT 13 but everything else leading up to it.

Once the disk check is located and analysed, the next step is to figure out how to disable or bypass it. This will vary from disk to disk and is beyond the scope of this introduction. Happy IBM cracking!

CLASSIFIED ADS

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

ALL of our Super IOB controllers (through 1986) 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:

► TWO 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.**

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 60 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 65 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

Legends tell of the days when the ancient back issues of Hardcore COMPUTIST were readily available to anyone who wished to purchase them. Those days may be long past, but the treasures with those ancient documents has been diligently transcribed to the pages of a modern reference work:

The Book(s) of Softkeys

Volume I *Compiled from issues 1-5*

contains softkeys for: •Akalabeth •Ampermagic •Apple Galaxian •Aztec •Bag of Tricks •Budge's Trilogy •Buzzard Bait •Cannonball Blitz •Casino •Data Reporter •Deadline •Disk Organizer II •Egbert II Communic. Disk •Hard Hat Mack •Home Accountant •Homeword •Lancaster •Magic Window II •Multi-disk Catalog •Multiplan •Pest Patrol •Prisoner II •Sammy Lightfoot •Screen Writer II •Sneakers •Spy's Demise •Starcross •Suspended •Ultima II •Visifile •Visiplot •Visitrend •Witness •Wizardry •Zork I •Zork II •Zork III ☐ PLUS 'how-to' articles and listings of need-to-have programs used to make unprotected backups.

Volume II *Compiled from issues 6-10*

contains softkeys for: •Apple Cider Spider •Apple Logo •Artist Arcade Machine •Bank Street Writer •Cannonball Blitz •Canyon Climber •Caverns of Freitag •Crush, Crumble & Chomp •Data Factory V •DB Master •The Dic*tion*ary •Essential Data Duplicator I & 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 4,4c,4.1c •Spy Strikes Back •Time Zone v1.1 •Visible Computer: 6502 •Visidex •Visiterm •Zaxxon ● software for: •Hayden •Sierra Online ☐ PLUS the ultimate cracking program: Super IOB 1.5 •••and more!

Volume III *Compiled from issues 11-15*

contains softkeys for: •Alien Addition •Alien Munchies •Alligator Mix •Comp. Prep. SAT •Cut & Paste •Demolition Division. •DLM software •EA (Electronic Arts) software •Einstein Compiler 5.3 •Escape From Rungistan •Financial Cookbook •Flip Out •Hi-res Computer Golf II •Knoware •Laf Pak •Last Gladiator •Learning With Leeper •Lion's Share •Master Type 1.7 •MatheMagic •Minus Mission •Millionaire •Music Construction Set •One-on-one •Penguin software •PFS software •The Quest •Rocky's Boots •Sabotage •Seadragon •Sensible Speller 4 •Snooper Troops II •SoftPorn Adventure •Stickybear series •Suicide •TellStar •Tic Tac Show •Time Is Money •Transylvania •Type Attack •Ultima III Exodus •Zoom Graphics •Breaking Locksmith 5.0 Fast Copy ☐ PLUS feature articles on ● Csaver ● The Core Disk Searcher ● Modified ROMs.

Name_____ ID#_____

Address_____

City_____ State_____ Zip_____

Country_____ Phone_____

VISA [master card] ____-____-____ Exp._____

Signature_____ CP54

☐ Volume I - $7.95
+ $2 shipping/handling

☐ Volume II - $12.95
+ $2 shipping/handling

☐ Volume III - $17.95
+ $2 shipping/handling

☐ All 3 volumes! - $30.00
+ $2 shipping/handling

Foreign orders (except Canada and Mexico) please add $5 for shipping and handling. Washington residents add 7.8% sales tax. Most orders are shipped within 5 working days, however, please allow 4-6 weeks delivery. **US Funds drawn on US banks only.**

Send to: Book Of Softkeys PO Box 110846-T Tacoma, WA 98411   (206) 474-5