For The Serious User Of Apple][Computers

COMPUTIST

Issue No. 44 June 1987

USA $3.75
Canada/Mexico $7.00
All Others $13.25

Softkeys For:

Coveted Mirror
Escape
Microwave
Catalyst 3.0
Number Farm
Black Cauldron
Goonies

Feature:

Making DOSless Utilities

Core:

ZRAM Memory Expansion: a review

(Page 24)

COMPUTIST
PO Box 110846-T
Tacoma, WA 98411

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

Coping With COMPUTIST

Welcome to COMPUTIST, a publication devoted to the serious user of Apple][and Apple][compatible computers. Our magazine contains information you are not likely to find in any of the other major journals dedicated to the Apple market.

New readers are advised to read this page carefully to avoid frustration when attempting to follow a softkey or when 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 disk can usually be copied by the use of Apple's COPYA program (on the DOS 3.3 System Master Disk).

■ **Commands And Controls:** In any article appearing in COMPUTIST, commands which a reader is required to perform are set apart by being in boldface and indented:

PR#6

The [RETURN] key must be pressed at the end of every such command unless otherwise specified.

Control characters are specially boxed:

6[CTRL-P]

Press [6]. Next, place one finger on [CTRL] and press [P]. Remember to enter this command line by pressing [RETURN].

■ **Requirements:** COMPUTIST programs and softkeys require one of the Apple][series of computers and a disk drive with DOS 3.3. These and other special needs are listed at the beginning of the article under "Requirements".

■ **Software Recommendations:**

1) Applesoft Program Editor such as Global Program Line Editor (GPLE).

2) Sector Editor such as DiskEdit (from the Book of Softkeys vol I) or ZAP from Bag of Tricks.

3) Disk Search Utility such as The Inspector, The CIA or The CORE Disk Searcher (from the Book of Softkeys vol III).

4) Assembler such as the S-C Assembler from S-C software or Merlin/Big Mac.

5) Bit Copy Program such as Copy][Plus, Locksmith or The Essential Data Duplicator

6) 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:** This powerful deprotection utility (COMPUTIST 32) and its various controllers are used in many softkeys. This utility is now available 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][Plus - Apple //e - Apple 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][Plus - Apple 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 - Apple //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:** The Apple][Reference Manual and DOS 3.3 manual are musts for any serious Apple user. Other helpful books include: *Beneath Apple DOS*, Don Worth and Pieter Lechner, Quality Software; *Assembly Language For The Applesoft Programmer*, Roy Meyers and C.W. Finley, Addison Wesley; and *What's Where In The Apple*, William Lubert, Micro Ink.

■ **Keying In Applesoft Programs:** BASIC programs are printed in COMPUTIST 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 keyed in appear in COMPUTIST as delta characters (Δ). All other spaces are there for easier reading. NOTE: If you want your checksums (See "Computing Checksums" section) to match up, you must only key in (Δ) spaces after DATA statements.

■ **Keying In Hexdumps:** Machine language programs are printed in COMPUTIST as both source code and hexdumps. 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 at the end of each line (a "$" and four digits). 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:

E003G

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

■ **Keying In Source Code** The source code is printed to help explain a program's operation. To key it in, you will need the S-C Assembler. Without this assembler, you will have to translate pieces of the source code into something *your* assembler will understand. A table of S-C Assembler directives appears in COMPUTIST 17.

■ **Computing Checksums** Checksums are four-digit hexadecimal numbers which tell if you keyed 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 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*
BRUNCHECKSOFT

Get the checksums with: **&** [RETURN] and correct the program where the checksums differ.

■ **CHECKBIN Instructions:**

CALL -151
BLOAD *program 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 ending address of the file followed by a [CTRL-Y] [RETURN].

xxx.xxx[CTRL-Y]

Correct the lines at which the checksums differ.

You have a LEGAL RIGHT to an unlocked backup copy

Our editorial policy is that we do NOT condone software piracy, but we do believe that users are entitled to backup commercial disks they have purchased. In addition to the security of a backup disk, the removal of copy-protection gives the user the option of modifying programs to meet his or her needs.

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

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

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

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

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

United States Code title 17, §117 (17 USC 117)

Be assured of receiving the latest issue of COMPUTIST each month without the hassle of making a trek to the local computer store and not finding COMPUTIST so of course you ask the clerk if they have COMPUTIST and they tell you they don't carry COMPUTIST but maybe you could try the computer store down the block because they might have an issue of COMPUTIST so you go to the computer store down the block and ask them if they have COMPUTIST but of course they don't (they just ran out of COMPUTIST yesterday) so you ride your unicycle clear across town to see if the computer store across town has COMPUTIST but when you go inside and ask for COMPUTIST they don't have COMPUTIST either and suggest that you try the store you originally went to in the first place so you end up going home disgusted that you missed another issue of COMPUTIST.

Can't Find COMPUTIST Anywhere?

Stop searching and **subscribe now!**

annual subscription rates (12 issues):

U.S. subscription sent third class $32
U.S./Canada/Mexico sent First Class $45
U.S./Canada/Mexico First Class plus library disk $100

Other Foreign $75
Other Foreign plus library disk $140

415 792-4908

- Use the form on the right to order or renew your subscription.
- You may upgrade your current subscription to a magazine + disk combination by sending $5.50 ($6.50 foreign) per remaining issue.
- Check your mailing label to see if you need to renew your subscription.
- 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.

Yes I want to subscribe. Enclosed are U.S. Funds (drawn on a U.S. bank) for a 12 issue subscription.

☐ **New Subscriber**
☐ **Please renew my current subscription**
☐ **This is a change of address (include latest address label)**

☐ **U.S. $32**
☐ **U.S./Canada/Mexico First Class $45**
☐ **U.S./Canada/Mexico First Class plus Library Disk $100**

☐ **Other Foreign $75**
☐ **Other Foreign plus Library Disk $140**

Name _____ ID# _____

Address _____

City _____ State _____ Zip _____

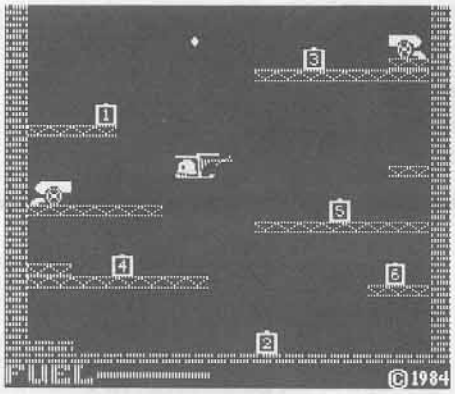
Country _____ Phone _____

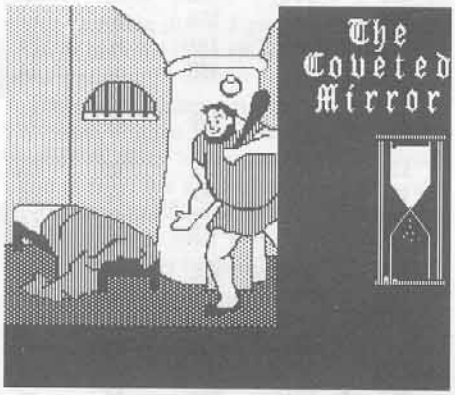
VISA MasterCard _____ - _____ - _____ - _____ Exp. _____

Signature _____ CP44

U.S. Funds drawn on U.S. bank, please. Subscription will not commence until funds are received.
Send orders to: COMPUTIST PO Box 110846-T Tacoma, WA 98411

Is it time to renew? Are you moving soon?
Take time now to save time later!





This month's cover:
Graphics from Penguin's "The Coveted Mirror."

SUBSCRIPTIONS: Rates (for 12 issues): U.S. $32, U.S. 1st Class, Canada & Mexico $45, Foreign $75. Direct inquiries to: **COMPUTIST, Subscription Department, PO Box 110846-T, Tacoma, WA 98411.**

DOMESTIC DEALER RATES: Call (206) 474-5750 for more information.

Change Of Address: Please allow 4 weeks for change of address to take effect. On postal form 3576 supply your new address and your most recent address label. Issues missed due to non-receipt of change of address may be acquired at the regular back issue rate.

COMPUTIST

Issue 44 June 1987

Publisher/Editor: Charles R. Haight **Managing Editor:** Ray Darrah
Technical Editor: Robert Knowles **Circulation:** Debbie Holloway
Advertising: (206) 474-5750 **Printing:** Ebsco Media, Birmingham AL
COMPUTIST is published monthly by SoftKey Publishing, 5233 S. Washington, Tacoma, WA 98409
Phone: (206) 474-5750

softkeys:

features:

core:

departments:

Cavalier Computer's **Microwave** *by Steve Ellis*, Sublogic's **Escape** *by Charles Taylor*, **Catalyst 3.0** *by Kevin Sartorelli*, DLM's **Number Farm and Alphabet Circus** *by The Nipper*, Avant Garde's **Joe Theisman's Pro Football** *by Tony Phalen*, Sierra On-Line's **Black Cauldron** *by The Nipper*, **International Granprix** *by Steve Ellis*

input

Please address letters to:

COMPUTIST
Editorial Department
PO Box 110846-K
Tacoma, WA 98411

Include your name, address and phone number.

Correspondence appearing in the INPUT section may be edited for clarity and space requirements. In addition, because of the great number of letters that we receive and the small size of our staff, a response to each letter is not guaranteed.

Our technical staff is available for phone calls between 1:30 pm and 4:30 pm (PST) on Tuesdays and Thursdays only.

Opinions expressed are not necessarily those of COMPUTIST or SoftKey Publishing.

Certificate Maker on a Franklin

I am a recent subscriber to your excellent magazine. I find it to be the best source of Apple information around.

I have used the softkey for Infocom programs described in COMPUTIST No. 24 and found it to work on several of my disks. However, I recently purchased "TRINITY" and when I tried to back it up I was unable to. Do any of your readers have a method which will work with this new program?

I have used Springboard's "CERTIFICATE MAKER" on my Laser 128 and found it to be a very useful program. When a friend purchased this program and tried to use it on his Franklin 1000 he was unable to use it since it is in ProDOS. When I tried to patch it as I have done with other programs on the Franklin the disk would subsequently not boot. A call to Springboard's help desk resulted in the information that "CERTIFICATE MAKER" would not work on a Franklin and that there was no way to make it do so.

However, with some experimentation I believe I have found a way. First, create a ProDOS disk with any volume name. Then copy the ProDOS system file to it. Using a program such as Copy][Plus Ver. 6.6, copy all the "CERTIFICATE MAKER" files to this disk. Then change the volume name to that which is on the "CERTIFICATE MAKER" disk. In this case it was "CM1A."

The ProDOS file must be patched to allow it to run on the Franklin. Since this is ProDOS 1.1.1 use a sector editor to search for and change $AE $B3 $FB $E0 $38 to $A2 $EA $EA $E0 $38 and $69 $0B $D0 $03 to $69 $0B $EA $EA.

When this disk is booted the logo and main title of the program come up. When the option to make a certificate is selected, and the program asks for the number of the certificate to be used, insert the original disk after typing the number. The program then runs normally.

This use of a pre-boot disk may not be the best way to make this program run on the Franklin, but due to my limited time to experiment, it seemed the easiest way. Perhaps another reader with more time and experience can describe a better solution.

Richard A. LaLonde
Port Huron, MI

APT's for Karateka

I found 2 APT's for Karateka. These patches only work on a broken copy such as the one presented in an earlier issue of COMPUTIST. Infinite Hits is just as it sounds. No matter what hits you... your flags (health) doesn't diminish. One Shot Kill is also as it implies. All you have to do is punch or kick the foe once, and he is dead! This also works with the evil Akuma and his pet bird. Only one restriction applies to One Shot Kill. Hitting the bird before you enter his chamber (before the man-eating gate) just sends him away as it did before. To kill the bird, you must kick down his door, and then punch or kick him.

Infinite Hits:

1) Scan your disk or file for the bytes:

C6 B6 D0 04 A9 00 85 5D 60

2) Change all but the 60 to EA's:

EA EA EA EA EA EA EA EA 60

One Shot Kill:

1) Scan your disk or file for the bytes: (I found 3 occurences)

C6 B7

2) Change these two bytes to EA's:

EA EA

Marc Batchelor
Lompoc, CA

Stellar Seven APT

Here is how to get unlimited Fuel and Shields in Stellar Seven. Get out a sector editor and change track $04, Sector $04: (all values given in hexadecimal).

TRACK	SECTOR	BYTE	FROM	TO
$04	$04	$7B	8D	EA
$04	$04	$7C	FC	EA
$04	$04	$7D	81	EA
$04	$04	$91	EF	EA
$04	$04	$92	8D	EA
$04	$04	$93	FB	EA
$04	$04	$94	81	EA
$04	$04	$95	A9	EA
$04	$04	$96	00	EA
$04	$04	$97	E5	EA
$04	$04	$98	ED	EA
$04	$04	$99	8D	EA
$04	$04	$9A	FC	EA
$04	$04	$9B	81	EA
$04	$04	$9E	00	92
$04	$04	$BE	8D	EA
$04	$04	$BF	FE	EA
$04	$04	$C0	81	EA
$04	$04	$DC	8D	EA
$04	$04	$DD	FE	EA
$04	$04	$DE	81	EA
$04	$04	$DF	60	EA
$04	$04	$E1	00	92

Alan Takagi
Phoenix, AZ

Time Zone Fix

I'm writing to tell you of a problem with the Time Zone softkey in COMPUTIST No. 31. Making the sector edits described in the softkey resulted in a copyable Time Zone where I couldn't pick any items up. If you can't pick the oxygen mask up in the time machine, then exchange the following sector edits for those in the earlier softkey.

TRACK	SECTOR	BYTE	FROM	TO
$03	$0F	$00	$CE	$60
$03	$0F	$01	$03	$E0

Next, I would like to make a plug for the best word processor on the market for an Apple //e or //c. Over two years ago I wrote a program to download custom character fonts to my

input

Apple DMP printer. I thought it was great and used this capability almost daily. My program has become obsolete, however, now that I can print almost any number of custom fonts, each with a multitude of different styles and sizes, with MultiScribe, by StyleWare. I couldn't be happier.

Bryce L. Fowler
Palo Alto, CA

Another Pirate Letter

Although I agree with the other pirates that people will seldom buy high-priced originals, I believe that that's not the real reason why we do what we do. Speaking for myself and friends, we consider pirating a sport. For instance, even though I've got a ton of wordprocessing disks, I go that extra mile to crack or trade for that extra utility I'll probably never use.

There is great competition among pirates. The pirate with the most games and the biggest number of allies is the one the other pirates come to first. It's a social event when pirates get together and discuss what they just traded and brag about what they just cracked. Of course, the leader of the group is in charge of what goes on.

I have several apprentices who I plan on teaching the trade. In return, they go out and look for games I want. They make the trade after it's been cleared through me. It works out well and everyone enjoys himself.

As long as there is fun in the "Hack" the pirate population will flurish. It can't be stopped. Most pirating is done on modems in the privacy of one's own home.

Remember this - beware of Dragon Law. They have been known to turn in pirates. They receive money for each pirate they hook.

And thanks COMPUTIST for making my sport easier!

LOOK FOR ME!

The COPY CAT
Newark, NY

Hint Decoder Program

If any of my fellow COMPUTIST readers have sent away for the hint book for The Crimson Crown, you've noticed that there are a ton of hints in it. The bad news is that each one is encoded. The task of decoding them could take hours (if not years) to get them all. The Hobbit is the same way except there are only a few hints that are encoded. The task is still time consuming. If you're like me, lazy, and don't want to spend the time decoding all of those hints, just type in the following program and it will take care of everything for you.

```
10 G$ = "" : HOME : INPUT "TITLE:^ " ;T$ :G =
    LEN (T$ ) :A = (80 - G ) / 2 + 1 : PR# 1 :
    HTAB A : PRINT T$ : PRINT : PR# 0
20 IF T$ = "THE^ HOBBIT" THEN B = 76 :C = 90 :D
    = 65 :E = 75 :F = - 11 :J = 15
30 IF T$ = "THE^ CRIMSON^ CROWN" THEN B = 65
    :C = 78 :D = 79 :E = 90 :F = 12 :J = - 14
40 GET A$
50 FOR G = B TO C : IF A$ = CHR$ (G ) THEN PRINT
    CHR$ (G + F ); :L = L + 1 :G$ = G$ + CHR$ (G
    + F )
60 NEXT G
70 FOR H = D TO E : IF A$ = CHR$ (H ) THEN PRINT
    CHR$ (H + J ); :L = L + 1 :G$ = G$ + CHR$ (H
    + J )
80 NEXT H
90 FOR Q = 32 TO 64 : IF A$ = CHR$ (Q ) THEN
    PRINT CHR$ (Q ); :L = L + 1 :G$ = G$ + CHR$
    (Q )
100 NEXT Q
110 IF A$ = CHR$ (13 ) THEN PRINT : PR# 1 :
    PRINT G$ : PR# 0 :G$ = ""
120 IF A$ = CHR$ (27 ) THEN G$ = "" : INVERSE :
    PRINT "CANCELLED" : NORMAL : PRINT
130 GOTO 40
```

Operating the Program

The first thing you should do is to turn on your printer, then run the program. When the program asks you for the title of the game type in The Crimson Crown or The Hobbit, this will set it up according to the encoded hints. If you make a mistake, don't panic just press escape and that line will not be sent to the printer. Sorry there is no delete function included in the program.

A few more things. For those of you who can't figure out the riddles in The Crimson Crown, here are the answers in order: FEAR, CLOUDS, and DREAMS. I've been able to get all of the treasures in The Crimson Crown except the scroll and the stone tablet (you need the scroll in order to get the stone tablet). If anybody can help me with this please write to:

Keith Parker
117 Chestnut Circle
Richmond, IN 47374

p.s. Thanks so much for publishing the ultimate Apple magazine. It has really helped me learn more about the Apple computer.

Speeding Up Microtype

Concerning your softkey for Microtype in COMPUTIST No. 29, I have put normal or fast DOS onto Microtype. By following these steps. You to may put fast or normal DOS on your disk.

1) Initialize a disk by the filename of HELLO using your DOS.

2) Copy tracks $3-$22 or decimal 3-35 from your unprotected Microtype.

3) Now here comes a step that might take a little while. Take all the dates off of the end of the filenames. You can speed this process by using a sector editor to do this on track $11 or decimal 17.

Everything will now run fine. You may even put Microtype on a hard disk.

Concerning deprotection of Mastertype:
1) Use a swap controller to copy tracks $1-$22 or decimal 1-35.

2) Then copy on track $0 from the 3.3 system master. By doing this you change the read/write routine to normal while keeping their strange command table. You also climinate the almost murder to copy track 0 of Mastertype.

Have fun with your unprotected programs.

Matthew Bancroft
The Christmas Elf
(age 11)
S. Dartmouth, MA

Fixing Modified CTRL Y

Recently I have observed a condition that although I can correct it, I can't explain the cause. Sound familiar!!??

I have been using a modified F8-ROM (COMPUTIST No. 19). All functions seem to work fine and I enjoy it.

A short time ago I had a need to use the APPLE-USR function from the Monitor [CTRL-Y], which as you know causes a jump to location $3F8 where a user routine can be located. No amount of coaxing would cause the [CTRL-Y] to initiate the routine. A direct call to this address would produce the routine action. Systematically, I back tracked through the Monitor to the CHRTBL ($FFCC) and SUBTBL ($FFE3) to where the keyboard character decode tables are. All seems correct, the Earl Taylor Article did change the call

input

address from $FEC9 to $FEC4. This was added to the routines by him with the change in the SUBTBL. I found that a swap of the addresses in the table would cause the routine to function. That is, a swap in the table with another function and both routines would work. Nothing was changed except the location in the table. Actually, in my case I swapped locations in the table with [B], but I have tried other table locations without any problems.

I have used this on two Apple][Plus systems, one with an Apple Language Card and the other with a Cocx 16K-RAM Card, with the same reaction and results. Oh! I almost forgot to tell you what it did before the change - "Nothing". It merely behaved as though you had just hit "Return". Has anyone else experienced this problem?? Any explanations or suggestions?? If this has been discussed before, I missed it.

I enjoy your excellent magazine and always look forward to the next issue. Keep up the great work. I learn something with each issue.

Ralph H. Benson
Richardson, TX

Odds and Ends

Here are a couple of odds and ends that I've collected and thought that other readers may find some interest in.

APT for Lode Runner

[@] will give you extra men
[^] will quit the current level and send you to the next level

If you use either of these hidden options, you will not be able to save your score.

APT for Conan

Unlimited men: search for the byte sequence CE 4B 03 and change it to EA EA EA.

Unlimited hatchets: search for the byte sequence CE 4E 03 and change it to EA EA EA.

APT for Lost Tomb

Unlimited men: search for the byte sequence CE F4 02 and change it to EA EA EA.

Unlimited whips: search for the byte sequence AE F1 02 F0 0A CA and change the CA to EA.

Both of these changes should be within $50 bytes of each other.

Ultima IV with Mockingboards

Ultima IV, when played with a Mockingboard installed, is truly a superb game to play. The music has to be heard to be believed. There are, however, 'secret' keys that enable the user to be able to play any of the tunes at will from the title screen after the Mockingboard has been activated. Apparently Origin System either forgot to mention it or the accidently left in a music test mode while debugging the game. Whatever the reason, here are the keys to press to get the different tunes:

```
C - Castle Theme Song
D - Dungeon Music
O - Main Surface Music
T - Theme Music
B - Lord British's tune
(1-5) - play different variations of curr-
ent theme
0 - turn off music
```

I found out about the keys while playing around in the code and thought users might be interested in knowing about these "Easter Eggs" (to coin an old phrase from the Atari 2600 games).

Computer identification

If you write programs for several different flavors of Apples, you undoubtedly need to be able to find out which one you're on. Here is a short routine to accomplish the feat:

```
10 ADDRESS = 64898
20 ID = PEEK( ADDRESS) + PEEK( ADDRESS + 1)
30 IF ID = 208 THEN PRINT "FRANKLIN ACE 2000"
40 IF ID = 253 THEN PRINT "LASER 128"
50 IF ID = 264 THEN PRINT "APPLE ][+"
60 IF ID = 296 THEN PRINT "APPLE //E"
70 IF ID = 420 THEN PRINT "APPLE //E
   (ENHANCED)"
80 IF ID = 468 THEN PRINT "APPLE //C"
```

I do not have access to an original Apple][or to earlier series Franklin computers. I encourage readers to send those ID values in so that others may benefit.

Jim S. Hart
Jacksonville, NC

Quickloader Deprotects

You can deprotect a few single-load programs using Southern California Research Group's "Quickloader".

After your program is running, push "B" and "RESET" (very few pointers are disturbed) then SAVE or BSAVE etc.

One program this works with is "Instant Recall" (Sams Software)

BSAVE INSTANT RECALL, A$803,L$4500

Put a Regular DOS on Tracks 0-2 and:

10 PRINT CHR$(4)"BRUN INSTANT RECALL"

Add Pronto DOS or Diversi DOS or like and the program loads faster than original. Works where COMPUTIST softkey for this program did not on my copy. (//e and //c version).

Thomas J. Scott
Mattapan, MA

Apple Writer with a Corvus

This letter is being typed on a //e with Apple Writer //e word processing software. The disk which booted the computer is a "softkeyed" copy of the original master disk which was purchased along with the computer some two or three years ago. I'm grateful to COMPUTIST for providing the instructions which helped me to make this backup.

There is a feature in this Apple Writer which apparently is not covered in the published documentation. When the disk starts booting type a "C" and near the end of the boot you will get a prompt "ENABLE CORVUS IN SLOT 6 ? Y/N." In order to make this work properly you must first boot the Corvus then do a CTRL-RESET and type PR#4 (4=appropriate slot) to boot the Apple Writer, if you neglect the CTRL-RESET it won't work.

Respond with a "Y" and if you have a Corvus in slot 6 the Apple will reach out and stroke and tickle it and cause you to be able to load and/or save files to the Corvus.

One would think that this same technique would work for a Sider. But try as we might, I and others have been unable to do this successfully. Being grossly ignorant of these things myself, I solicited the aid of a man who is, I think, quite expert at things pertaining to Apples, Siders, and Corvuses (Corvi?). After he made some unsuccessful tries he suggested that I get my Apple dealer to pose the question to Apple on their "hot-line."

The dealer did this and the reply he got from Apple was that the solution to my problem would be for me to buy a PRODOS version of Apple Writer which would work on the Sider.

input

Seems that this is the super-easy way out for Apple - they don't do anything and I spend $200 for a new piece of software that I don't really need.

James T. Baker
Huntsville, TN

Create with Garfield and PrintMaster

First of all, I would like to thank you for a speedy delivery of my sample edition of COMPUTIST. I found it very interesting and thought that I might be able to contribute something to your softkeys.

After looking through the "Most Wanted" list, I found that there are two programs that I've cracked and am now going to pass onto you.

Create with Garfield:

```
Boot up COPYA and change - B934:18 60
                           B990:18 60
```

Use a sector editor and for each of these, change the bytes to 18, F0, 82, 49:

```
Track $1D, Sector $02:
Track $1F, Sector $00:
Track $1F, Sector $06:
Track $22, Sector $04:
```

i.e. One change would be; Track $1F, Sector $00-18: A2 00 60. This changes the read during moves on menus (from AADE to DEAA).

PrintMaster:

I don't know how to physically change the disk for this, but all you have to do is to copy the disk with COPYA or Copy][Plus sector copy. Then use Copy][Plus bit copier and use the Sync on tracks 1, 2, & 3. This should now work. Word of warning: only use Copy][Plus to sync the tracks. I tried with other bit copiers and they didn't work.

One last note before I go. A friend lent me one of his COMPUTISTs, and it had parameters for Law of the West, Hardball, and various other ones. All that was mentioned was that I crack these using Locksmith 5.0. Is there any reason for using the parameters if Locksmith copies it normally?

Robert Brown
Sydney, Australia

APT for Raid Over Moscow

I enjoy your magazine very much but it's very hard to find someone that sells it up here.

I have an APT for Raid Over Moscow from Access Software. Press the "^" key during the title page and you will start playing at the last screen inside the Reactor Room without going through the other 7 or 8 screens.

The two games I just can't back-up are Fight-Night from Accolade and Impossible Mission from Epyx. Thanks for a super magazine.

Wakit So
British Columbia, Canada

APT's for Oldies but Goodies

After many hours of programming and playing games I have found many simple and user-modifiable APT's for some of the older Apple][games. Most of these require that you have a broken copy of the game.

Snake Byte:

BLOAD SNAKE BYTE
CALL-151
76AE:No. of Snakes
250G

Gold Rush:

BLOAD GOLD RUSH
CALL-151
BE3:No. of Men

Threshold:

Unlimited Ships:

45B0:EA EA EA
7ECD:EA EA EA

Prevent the laser from overheating:

7666:4C 7D 76

Unlimited Fuel:

7323:EA EA EA
7839:EA EA EA

Night Crawler:

BLOAD NIGHT CRAWLER
CALL-151
340A:No. of Ships
3300G

I would like to thank Nibbles 'N' Bytes and Lancer of Brand-X for the needed help they have given me in disassembling the code that made these APT's possible. Keep up the good work!

Roman Drozd
Perry Hall, MD

Locksmith Praise

There is a new piece of software I feel your readers should know about. For anyone who thinks the Locksmith series is obsolete, and hasn't seen the Locksmith 6.0, I suggest you buy it. It has many useful utilities, far more than the 5.0, and a readable, informative manual. The most interesting feature is the Automatic Boot Tracer, which sumates the operation of the 6502. It is possible to "boot" a disk under the control of this utility, and trace the boots to find JMP's to nibble count routines, or find out where a certain routine occurs, stop the ABT, examine code if you wish, and continue. The bit-copier is also a lot better than the 5.0 bit copier. Last but not least, the file disk is not protected.

Ray Brooke
New Brunswick, Canada

A New Print Shop

The latest version of the Print Shop (Broderbund Software) can't be deprotected by the softkey published in COMPUTIST No. 17.

This article was a great help as it gave me a place to start. Tracing the whole disk failed to find any JuMP (or JSR) to address $BCE0. I booted the original disk and jumped into the monitor using a Wildcard 2. Tracing the HELLO routine (which loads at $800) does indeed find an indirect jump to the infamous $BCE0 - a neat example of self-modifying code. The indirect jump on the disk is to $17E0 which gets changed to $BCE0 during execution. The fix is to NOP the three bytes of the indirect jump so the next instruction (an RTS) is executed.

The procedure for making a bootable copy can be summarized as follows:

1) Make a copy of the original diskette. I used Copy][Plus with the Print Shop Alt. 4 option. This copy won't boot.

2) Using a sector editor program (I used ZAP from Bag of Tricks 2) find the indirect jump string $6CDB0E. I found it in track $19, sector 8. Continuing with ZAP I changed these three bytes to $EA (NOP) and wrote the sector back.

3) Using INIT (Bag of Tricks 2) I initialized (erased) track $22. This is optional but it does make it easer to "copy the copy."

Ron Tipton
Raymore, MO

input

Operation Frog

I noticed in COMPUTIST No. 38, you had listed Operation Frog by Scholastic, on the most wanted list.

I have been successful in deprotecting this program.

Requirements:
FID
An initialized disk with no hello file
Operation Frog

1) Initialize a disk and delete the hello.

2) Boot Operation Frog. After the boot is complete reset into the monitor. I do this with a Wildcard, but pressing CONTROL-RESET a couple of times and then typing CALL-151 will also work.

3) Move a part of the protected RWTS to a safe area

4800<B800.BFFFM

4) Remove Operation Frog and replace it with the slave disk created earlier. Boot it by typing 6 [P].

5) Insert your Apple system master and BLOAD FID.

6) Enter the monitor.

CALL -151

7) Move the protected DOS back into an area where it can be used.

B800<4800.4FFFM

8) Return to Applesoft

[C]

9) Activate FID

CALL 2051

10) Put Operation Frog into drive 1 and your slave disk into drive 2.

11) Choose copy files from the FID menu. Use the wildcard "=" when asked for file name. Transfer all files from Operation Frog to the slave disk.

When this is finished your new copy will be fully functional and COPYAable. The back of Frog is unprotected and may be copied by any copy program.

That is all there is to it. I hope I have been of help.

R. A. Clark
Jeffersonville, IN

Jane

On my copy of Jane the disk can be copied with any copier if the following modifications are made:

TRACK	SECTOR	BYTE	TO
$7	$C	58	00
$7	$C	98	00
$9	$4	80	00
$9	$4	38	00
$9	$F	18	00
$9	$C	46	00

Thank you for a very nice magazine that is most useful to all serious Computists in the world.

Judy Sui Chu Chanduloy
North Point, Hong Kong

Bank Street Reply

In reply to Mr. Bancroft's problem with the Bank Street Writer softkey (COMPUTIST No. 25), he is quite correct that the softkey, as written, will not permit document saves to a data disk. Omitting the recommended sector edit to track $04, sector $07, will permit document saves but will no longer allow changes to the utility program.

I determined that track $04, sector $07 loads into memory at $B700. The subroutine has two entry points. One at $B709 and one at $B70C. If entered at $B70C, DOS is modified to read the nonstandard prologs and epilogs on the protected BSW disk. If entered at $B709, DOS is returned to normal.

The real problem actually resides on track $04, sector $06, which loads into $3600. When BSW attempts to do a document SAVE, the code at $363B first calls the subroutine through $B70C. That is, DOS is modified to read the protected disk. If a data disk is in the drive, BSW expects its first read attempt to fail, and expects the carry flag to be set. BSW then calls the subroutine through $B709 and expects the second attempt to be successful. In the event that the first attempt is successful, BSW believes that the program disk is in the drive, and does not permit the SAVE.

The routine must be altered to not set the carry flag on the first read attempt, and to not make the second attempt. The set carry flag instruction is at $B64D cleverly disguised in a BIT $38 instruction. The instruction can be disabled in several ways. I chose to change byte $42 on track $04, sector $06 from a $0A to a $09. This accomplishes both objectives. Also, on track $04, sector $07 instead of changing byte $37 to a $60 as recommended, I changed byte $39 to a $60 instead. This preserves both entry points to the subroutine.

BSW will now work properly.

Don Birdsall
Fulton, NY

JigSaw & Mychess II

On my copy of JigSaw the disk can be copied with any copier if the following modifications are made:

TRACK	SECTOR	BYTE	TO
$0	$9	A8	00
$0	$0	B0	00

On my copy of Mychess II the disk can be copied with any copier if the following modifications are made:

TRACK	SECTOR	BYTE	TO
$0	$4	EA	00
$0	$5	D5	00

Thank you for a very nice magazine that is most useful to all serious Computists in the world.

Judy Chanduloy
North Point, Hong Kong

Keyboard Repair Castle Wolfenstein

Philip Goetz's article on Keyboard repair was great. I would like to add some comments with regards to it. My][Plus was purchased second hand and the original owner was taken to abusing the keyboard in the area of the RESET key, to the detriment of it and six other local buttons.

I found a shop that repaired Apple]['s for a school system and purchased second hand keys for $1.50 each. These keys can be refurbished easily enough by carefully disassembling and retensioning the stainless spring contact within. Take a cue-tip, moistened with alcohol, and wipe over the surfaces of both the moving and stationary contacts. Re-assemble and install.

input

A word about soldering equipment. To de-solder, use an iron with sufficient mass but low temperature. I use an Ungar 50W #4039 plugged into a Variac set to 90-95VAC. Use damp sponges and wipe the tip of the iron before every soldering or desoldering operation. (Note: Weller temperature controlled irons are excellent, but expensive). For soldering, I use an Ungar 25W iron with a "nib" tip, #6952 or #6962.

When de-soldering use a solder sucker, such as "Soldapult III" model PT209, to remove the bulk of the solder. Clean up with "Soder-Wick" #3, cutting off the used braid frequently with diagonal cutters. Finally, to break loose any remaining solder at a solder through hole, take a pair of tweezers and wiggle the component lead back and forth. The component should practically fall out by itself when the circuit board is turned upside-down.

A different subject. My philosophy is to have programs, games, or otherwise, running out of a Hard disk or RAM-disk. To that end I have deprotected a couple of disk-intensive programs and made them usable with a RAM-disk. This does require an old F8 ROM or similar RESET capability.

Castle Wolfenstein

1) INITialize a slave disk.

2) Boot the protected game disk and press CTRL-RESET as soon as the title page appears. Once in the monitor, move RWTS to A$1900. Type

```
1900<B800.BFFFM
```

3) Now boot the slave disk. When the drive stops, insert your Super IOB v 1.5 and save the RWTS to it.

```
BSAVE RWTS.WOLF,A$1900,L$800
```

4) Now write a Controller using Nick Galbraith's program "Controller Writer". Specify the start track as $03, the end track as $22. Also the start sector as $00 and end sector as $0C. Specify RWTS.WOLF when asked by the program and 0 for sector edits. Use an original name like "CASTLE STEIN", when asked.

5) To copy the Wolfenstein game disk insert your Super IOB/Controller/RWTS disk and type

```
LOAD SUPER IOB V1.5
EXEC CASTLE STEIN.CON
RUN
```

6) Now copy the game using your pre-INITialized slave disk and by following the screen directions. When completed, copy the disk to your RAMdisk or Hard Disk. To play the game, run the 'HELLO program.

Two other programs were deprotected similarly, except the orginals are 16 sector disks. Modify the Controller accordingly. (Galactic Empire and Castles of Darkness)

I would like some advice deprotecting an old 13-sector copy of Zork I and a 16-sector copy of AppleWriter II. I would like to be able to run these programs from Hard Disk as well.

COMPUTIST is the best Hacker's magazine I've come across, keep up the good work. Hackers unite!

Norman F. Hogarth
Lomita, CA

The Pirate Parade

I would like to respond to the series of letters pro and con on software piracy, especially about pirates. I am a pirate. I go under the code name "The Enigma". My friends know I have a lot of software (right now, if I go with the list price of my stuff, it comes to over $50,000). My reasons for piracy are a bit different from some, but in other areas I am in agreement.

One: I can't usually afford to go and buy software. Face it, it is very expensive. If it wasn't for my pirating or getting duplicates from friends, my computer is nothing more than a very expensive dust collector (and I have a lot of dust where I live). I need not go into the price variances from one region to another (ex. Far East vs. here). That has been covered by many people quite well in the past few letters.

Two: On some pieces of software, the protection scheme is better than the program itself. It is a very good way to learn some tricks in assembly language. I look at the protection scheme as a Jigsaw/Crossword puzzle. Face it, some games themselves are so bad you are almost embarrassed to let your friends know you bought it, and if you did, you find out you can't return it to the store because of some wierd return policy (which is in some cases actually valid. More on this later).

I act as an information service for my friends. I also do private tutoring in Appleworks (This letter is being typed on a 'pirated' version of Appleworks by the way). Face it. A lot of salesmen do not know what the hell they are talking about, or know how a program will fit a customers' needs. I work in a computer store, by the way, and I hear these complaints left and right on some local Apple dealers! Example: A friend of mine bought Appleworks for their business and was informed that Appleworks cannot do mailing labels (HA!). So they spent another $70 (?) on Habamerge because it(!) can do mailing labels. I could have saved them a lot of grief if they spoke to me first. My advice is free and I usually steer people in the right direction. My friends come to me and ask my advice on a particular program, and if I have a cracked version, I give them my unbiased advice. Based upon their needs, I recommend the program that they can handle. I believe I have saved hardware and software companies 3 times the amount that I currently have because I help them understand their equipment.

Three: I do not believe in stealing programs. My views on copyright laws are this: Pirates cannot make money off of someone elses work. If person A has Appleworks and friend B says he wants a copy, it is up to person A whether he/she will let their friend have a copy. If he/she charges for copying it he deserves to have his butt kicked into jail and have the keys thrown away! Lets say you have the hottest album at your house, your friend finds out and asks if you can make a tape for him. You say, "Sure, no problem, just give me a blank tape, and I'll run it off." Technically you just broke copyright laws, but that is not enforced. That is the view I take. Another thing that perturbes me, are the deadbeats who buy a program, take it home, copy it, and then try to return it for a different one in its place. That is why some places have such strict return policies. If you can't get the program legitimately (copying from a cooperative friend, or buying it flat out), don't get it at all.

To wrap it up, all pirates are not the same. Some of your better cracks are probably from pirates who are actually using their real names (WOW! What a concept!). There are a lot of pirates out there who give pirates like me a bad reputation, but some of them are evolved hackers who break into MCI (phreaking), call long distance BBS systems that support downloads, getting pre-released software and doing all sorts of nasty stuff. Just look in your papers on those guys who were caught and read on all the stuff they have. There is stuff in public domain software that can give the capabilites to a novice hacker that makes "War Games" look like a rank amateur. This (software piracy) is really just the tip of the iceberg in some cases. I apologize for the length of this letter, but this has been bothering me for a long time. I hope you publish this letter, either in part or as a whole.

The Enigma

readers' softkey & copy exchange

Steve Ellis' softkey for...

Microwave

Requirements:
Apple][series with at least 48K
sector editor
disk copier which can ignore errors
blank disk
Cavalier's Microwave disk

Microwave from Cavalier Computer is one of those increasingly rare releases - a one byte crack. The disk can be copied using any copy program which will ignore errors. It is encoded normally, with track $11 containing the data for a nibble count. Changing one byte disables this nibble count.

1) Copy the entire disk, ignoring any errors on track $11 (or simply do not copy track $11).

2) Search the copied disk for the sequence: $F0 03 4C 32 8A. I found this at track $01, sector $0C, byte $4F.

3) With a sector editor, change the $4C in the sequence to a $2C.

You now have a fully working copy. And here is a little cheat for the game. Track $01, sector $03, byte $46 contains a $03. Changing this number to a $00 gives you only one monster to worry about. This byte is found in memory at $8146.

Charles Taylor's softkey for...

Escape

SubLogic
713 Edgebrook Drive
Champaign, IL 61820

Requirements:
A blank disk
Super IOB
DOS 3.3 System Master

Escape is a hi-res strategy game that places you in an electronic prison. You must try to escape the prison by causing the guards to run into each other or into the electric posts in the prison. Sometimes you can just outrun them. Other devices to assist you are an eight shot weapon, and the ability to transport within the cell. Transporting is a mixed blessing, to be used only when faced with certain death, because frequently you transport into the guards or posts.

Escape is copy-protected by altered data marks (D5 AA DA for the prologue instead of D5 AA AD, and ED AA for the epilogue instead of DE AA), and by two odd DOS commands. In Escape's DOS "B" will BLOAD a binary program, and "A" will run an Applesoft program.

Procedure

1) Boot the system master and init a blank disk side

INIT A

2) Merge the controller included below into Super IOB and copy the Escape program to your freshly INITed disk. This controller will read data prologues of D5 AA DA and epilogues of ED AA, copying tracks $3-$22.

3) You can now CATALOG the copy, but running the copy will produce a syntax error at line 200 of the file "A" if you first remove the error code at the beginning of the file. (The first four line numbers.) Line 200, which appears to read PRINT "BD", is actually PRINT "[CTRL-D]BD", and is supposed to BLOAD file "D". Load and unlock files A and B, edit the following lines listed below, and save the files back to the copy. You now have a deprotected Escape.

File "A":

line	from	to
200	PRINT "BD"	PRINT CHR$(4) "BLOAD D"
220	PRINT "BE"	PRINT CHR$(4) "BLOAD E"
410	PRINT "AB"	PRINT CHR$(4) "RUN B"
450	PRINT "AC"	PRINT CHR$(4) "RUN C"
7100	PRINT "BF"	PRINT CHR$(4) "BLOAD F"

File "B":

line	from	to
760	PRINT "AC"	PRINT CHR$(4) "RUN C"

controller

```
1000 REM ESCAPE
1010 TK = 3 :LT = 35 :ST = 15 :LS = 15 :CD = WR
     :FAST = 1
1020 RESTORE : GOSUB 210 : 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 "COPYDONE" : END
5000 DATA 213 ,170 ,218
5010 DATA 222 ,170 ,237 ,170
```

controller checksums

```
1000 - $356B    1040 - $B560
1010 - $2445    1050 - $D1B3
1020 - $DDA5    5000 - $8249
1030 - $E83C    5010 - $3290
```

Kevin Sartorelli's softkey for...

Catalyst 3.0

Requirements:
A copier that can ignore read errors
ProDOS User's Disk
A DOS 3.3 slave disk
Catalyst 3.0 original disk
A blank disk

Catalyst 3.0 is a ProDOS based disk that has a file that is unreadable by normal ProDOS. This file contains the Catalyst loader and some of the system setup parameters. In order to deprotect Catalyst this file is read by boot code tracing and the information saved as a SYS type file.

1) Copy the Catalyst disk using Locksmith Fast Copy or any copier that ignores errors.

2) Boot ProDOS and exit to BASIC.

3) Insert your original Catalyst disk in the drive and load the .SYSTEM file into memory.

BLOAD CATSTART.SYSTEM,A$2000,TSYS

4) Enter the Apple monitor and patch the .SYSTEM file to enter the monitor after the load is completed.

CALL-151
2175:4C 59 FF

5) Start the loader to read the protected tracks into memory.

2000G

6) When the loader stops, disable the program's check for the protected tracks.

40F5:EA EA EA

7) Boot a DOS 3.3 slave disk and save the protected tracks as a binary file.

BSAVE BOOT4,A$4000,L$2100

8) Use ProDOS' CONVERT or Copy II Plus 6.x to move the new file to ProDOS.

9) Boot ProDOS and exit to BASIC.

readers' softkey & copy exchange

10) Load the .SYSTEM file into memory again.

```
BLOAD CATSTART.SYSTEM,A$2000,TSYS
```

11) Enter the monitor and add the following hexdump to the .SYSTEM file.

```
CALL -151

28EF:A9 18 8D 00 20 A9 90 8D
28F7:01 20 A9 32 8D 02 20 A0
28FF:00 B9 00 4A 99 00 60 C8
2907:D0 F7 CE 02 29 CE 05 29
290F:AD 05 29 C9 3F D0 EA 4C
2917:00 20
```

12) Patch the .SYSTEM file to jump to our new code, and to call the code from the protected track.

```
2000:4C EF 28
20A9:4C 00 40
```

13) Load the data captured from the protected track into memory.

```
BLOAD BOOT4,A$2A00
```

14) Save the resulting new .SYSTEM file to the copied Catalyst disk.

```
BSAVE CATSTART.SYSTEM,A$2000,L$2B00,TSYS
```

15) Boot your deprotected copy of Catalyst 3.0.

source code for patch

```
(placed at end of CATSTART.SYSTEM)

                        .OR  $28EF
restore start of program
  28EF: A9 18           LDA  #$18
  28F1: 8D 00 20        STA  $2000
  28F4: A9 90           LDA  #$90
  28F6: 8D 01 20        STA  $2001
  28F9: A9 32           LDA  #$32
  28FB: 8D 02 20        STA  $2002

loop to move protected track data
  28FE: A0 00           LDY  #$00
  2900: B9 00 4A  LOOP  LDA  $4A00,Y
  2903: 99 00 60        STA  $6000,Y
  2906: C8              INY
  2907: D0 F7           BNE  LOOP
modify loop for another page
  2909: CE 02 29        DEC  LOOP+2
  290C: CE 05 29        DEC  LOOP+5
  290F: AD 05 29        LDA  LOOP+5
  2912: C9 3F           CMP  #$3F
  2914: D0 EA           BNE  LOOP

go start program when done
  2916: 4C 00 20        JMP  $2000
```

The Nipper's softkey for...

Number Farm and Alphabet Circus

DLM
1 DLM Park
Allen, TX 75002

Requirements:
48K Apple II
Alphabet Circus or Number Farm
COPYA, Sector Editor
A blank disk

DLM has begun to distribute a series of early childhood programs written by Neosoft. Since they are intended for use with small children it is very important to be able to back them up. Unfortunately the programs in this series, and their various versions, use a variety protection systems.

The Protection

Disks in this series using the protection present on these two programs have a standard prologues and epilogues (D5 AA 96, DE AA, D5 AA AD, DE AA). This means they can be copied with COPYA. The usual form of protection on such disks is a Nibble Count, and these two are no exception.

To explore the nibble count, first make a copy of your original disk using COPYA. Now boot up the copy. It will boot for a bit and then reboot again and again. The problem is a nibble count is done on Track 00. By stopping the boot just before it reboots and checking the program counter you will find the program was executing code on the $90 page. To be exact, the code responsible for the nibble count resides from $9058 to $90B3. A search of the disk for the first five bytes of the count code (BD 89 C0 A9 56) shows you that this code is on the disk at Track $06, Sector $0D starting at byte $5C. All that you need to do is place a simple RTS (60) at the start of this routine to negate the nibble count.

By the way, the catalog on the disk starts on track $03 sector $0F and the nibble count is contained in a binary file called HELLO3. To make your backup catalogable change byte $01 on track $11 sector $00 from $11 to $03.

Step by Step

Run COPYA from your DOS 3.3 System Master disk and copy you original disk. Now boot up a sector editor such as DiskEdit (Book of Softkeys Vol I) and make the following changes:

Track	Sector	Byte	From	To
06	0D	5C	BD	60

That's it. You now have a COPYAable version to let the children use. Enjoy!

Tony Phalen's softkey for...

Joe Theisman's Pro Football

Avant Garde Creations
P.O. Box 30160
Eugene, OR 97403
$39.95

Requirements:
48K Apple II
Super IOB
Two blank disks (or one notched one)
Joe Theisman's Pro Football disk

Joe Theisman's Pro Football is a football simulation which teaches one the art of football, from both the offensive and defensive perspectives. It covers everything from equipment (such as how to take care of it, how to use it correctly, and why it's used) to the players (such as what each player's main "job" is). It's a very educational program on the "how-to's" of football, and even comes equipped with its own version of "Pro Football". Being a program that uses a lot of disk access, it would only be right to make one or two backup copies of it, and then store the original away for safe keeping.

The Protection

I first wateched the boot process and listened for any abnormal noises. It appeared to have a modified DOS (since no prompt appeared). Also, the title page came on very quickly, so I figured some files were loaded right away. I then tried to copy it, first using COPYA, but it gave me an "unreadable" error. So out came Disk Muncher, and while watching each track's status I noticed that track 0 read O.K., but track 1 didn't. Track 2 read O.K., but track 3 didn't! Hmmm... interesting. So I let Disk Muncher copy the rest of the disk, and I saw that every other track, starting with track 1, gave me a read error. Well, there's the protection! I then booted CIA's The Linguist and checked all the markers. The address headers had been changed to D5 AA AD, from the normal D5 AA 96. So, making these changes, I tried to read in the next track,but to no avail. Looking at a nibble dump of that track, I saw that the data field epilogs had been changed to FF FF, from the normal DE AA. I then changed the address headers back, and read in the next track, and it worked!

readers' softkey & copy exchange

Knowing these changes, I set out to write a controller that would ignore the modified markers and write the correct markers back to the copy disk.

Step by Step

1) Install the controller into Super IOB.

2) Run Super IOB.

3) Copy front AND back.

4) Boot and enjoy!

The controller also does a sector edit on track 0, sector 3. This may not be needed with some versions, but with mine it only booted 3 out of 5 times without it. Joe Theisman's Pro Football occupies almost all of the $22 tracks, and there is no nibble count, so there is no need to worry about that! After playing it for some time, you'll wonder if there is a way to make it harder!

controller

```
1000 REM JOE THEISMAN'S PRO FOOTBALL
     CONTROLLER
1010 TK = 0 :ST = 0 :LT = 35 :CD = WR
1020 T1 = TK : GOSUB 490
1030 POKE 47405 ,24 : POKE 47406 ,96 : POKE
     47497 ,24 : POKE 47498 ,96
1040 POKE 47445 ,213 : IF TK / 2 <> INT (TK / 2
     ) THEN POKE 47445 ,212
1050 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST
     < DOS THEN 1050
1060 IF BF THEN 1090
1070 ST = 0 :TK = TK + 1 :ZZ = 212 : IF TK / 2 =
     INT (TK / 2 ) THEN ZZ = 213
1080 POKE 4744 ,ZZ : IF TK < LT THEN 1040
1090 GOSUB 490 :TK = T1 :ST = 0 : GOSUB 230
1095 RESTORE : GOSUB 310
1100 POKE 47405 ,208 : POKE 47406 ,19 : POKE
     47497 ,208 : POKE 47498 ,183
1110 GOSUB 430 : GOSUB 100 :ST = ST + 1 : IF ST
     < DOS THEN 1110
1120 ST = 0 :TK = TK + 1 : IF BF = 0 AND TH < LT
     THEN 1110
1130 IF TK < LT THEN 1020
1140 HOME : PRINT : PRINT "COPYDONE" : END
5000 DATA 1^ CHANGES ,0 ,3 ,66 ,24
```

controller checksums

```
1000 - $356B      1090 - $B7C0
1010 - $3266      1095 - $1652
1020 - $C11A      1100 - $6AB4
1030 - $6F2E      1110 - $54B5
1040 - $FED0      1120 - $65C7
1050 - $F9D7      1130 - $568B
1060 - $72A3      1140 - $851A
1070 - $21AA      5000 - $5048
1080 - $2460
```

The Nipper's softkey for...

Black Cauldron

Sierra On-Line

Requirements:
48K Apple][
The Black Cauldron
COPYA
Sector Editor

The Black Cauldron is an animated adventure game in the style of King's Quest, from the people at the Disney studio and Sierra On-Line. It has state of the art graphics and movement, and is a real delight to play with commands in pull-up windows. Before I let the children at any new game, I set out to make a backup.

The Protection

The five sides that contain The Black Cauldron seem to contain no apparent protection and all of the sides can be copied with COPYA. When you try and boot it up, it runs fine until the first screen. After you center the joystick and press the button the "please wait" appears and the disk drive turns off and the system freezes. During this last disk access there is a check of track $00 which of course fails on our copy. After stopping the program, at the point it accesses track $00, I checked the counter I found that the code in the $FF00 page was being executed. This is a nasty location because it is difficult to view the bank switched RAM. If you have some way of way of saving this memory, such as the Senior PROM or Wildcard II, you will find that the code is loaded from Track $11 Sector $0F. This is the nibble count code. There are two ways of defeating this code.

The first is to NOP (EA) the jump subroutine to the nibble count (20 00 FF) that is on Track $03 Sector $0B. Instead I choose to put a return subroutine (60) at the start of the nibble count. Unfortunately there is a checksum done on the nibble count code so we have to put the BD we replaced where the next 60 is so the checksum will be correct.

Step by Step

1) Run COPYA and copy all five sides of The Black Cauldron.

2) Use your sector editor on the boot side only of the copied disk and perform the following modifications:

Track	Sector	Byte	From	To
11	0F	10	BD	60
11	0F	4B	60	BD

That's it. You now have an unprotected backup.

Steve Ellis' softkey for...

International Granprix

Requirements:
Apple][series with at least 48K
blank disk
Riverbank's International Granprix disk

International Granprix uses a protection scheme involving quarter tracks and a quick loader. This makes converting the entire disk to normal pretty tough, but since it is a single load game it can be placed into a normal DOS 3.3 binary file.

Following are the steps needed to make Granprix into a file:

1) Format a slave disk and Delete the Hello program.

INIT HELLO
DELETE HELLO

2) Boot the Granprix disk and when the program starts, press CTRL-RESET. This will put you in the monitor.

3) Move the needed code so that it will not be erased by a boot.

1000<6000.A0B0M
6000<800.900M

4) Boot the blank slave disk and move the code back.

6[ctrl]P
CALL -151
800<6000.6100M

5) Type in this short program to move the code to its final location.

```
50A3:A2 00 BD 00 10
50A8:9D 00 60 E8 D0 F7 EE A7
50B0:50 EE AA 50 AD AA 50 C9
50B8:A1 D0 EA 4C 00 60

7FD:4C A3 50
```

6) BSAVE the program.

BSAVE GRANPRIX,A$7FD,L$48C5

You now have an 74 sector cracked version of International Granprix.

Making DOSless Utilities

Adam Levin

Requirements:
A blank disk
DOS 3.3
An assembler (optional)

Often in these pages I read about the handy utilities some deprotectionists have at their disposal. The ones which interest me the most are the utilities which exist on an EPROM (Eraseable Programmable Read Only Memory), since they can be used instantly and independently of DOS. There are, however, reasons why some people (myself included) don't have one. My primary reason is that I refuse to commit myself to a final version of any of the utilities I use; I'm constantly re-writing them, adding new ones and dropping unused ones. Whatever the reasons, I must admit that there are times when I'm trying to deprotect commercial software, and I wish I could do a search, or dump my RAM card's contents without disturbing what I'm working on by booting DOS. I have found a method which allows me to do all of these things and more. To explain how my system works, I'll have to discuss a bit about how DOS boots a standard 3.3 diskette.

There is a machine language program stored in the ROM of the disk drive's controller card. When a disk is to be booted, this program first calibrates the drive's read/write head by moving it as far out (to the edge of the diskette, track $00) as possible. The controller then reads in the 256 bytes on track $00, sector $00 and writes them into memory from $0800 to $08FF. Since the controller card software cannot move the read/write head other than for the above-mentioned calibration, it is confined to the information on track $00. The byte which gets placed at $0800 (it came from track $00, sector $00, byte $00, and I call it the 'number-of sectors' byte) tells how many sectors the controller should read in.

On a normal DOS diskette, the number-of-sectors byte is a $01, so after reading sector $00 to page $08 ($0800 - $08FF), the controller program is done; it has read in one sector. If the number-of-sectors byte was anything from $02 to $10 ($10 being the maximum number of sectors on a track), the controller program would still have to read some more sectors. Additional sectors are read into memory starting at $0900 and continuing up to $17FF ($0F pages). It should be noted that successive pages are not read from the diskette sequentially; the sectors are interleaved, to increase the speed of the boot.

In any case, when all required sectors have been read in, the controller program JMPs to $0801 to begin executing the code which was just read in; usually the second stage of the bootstrapping process, called 'boot 1'.

I say 'usually' because I propose a change of events here. If we create a machine language program which originates at $0801, and write it onto the diskette starting at track $00, sector $00 (not forgetting to precede it with an appropriate number-of-sectors byte), this program will load and run at boot- up! No DOS was needed, and only the memory where the program resides, and anything it alters, gets changed. Additionally, the text page ($0400 - $07FF) may or may not get clobbered, depending on how RESET is being handled at that moment. I prefer to send all output from my DOSless utilities to my 80 column card (Videx-style, having its own text memory), since this minimizes the chance that the text page will get changed. I also try to keep my routines under $FF bytes in length. Going over that by even a few bytes requires that another sector be read in, overwriting an entire page in memory just for those bytes. The maximum length of the entire routine must be under 4,096 bytes, since this is the maximum number of bytes which will fit on a single track.

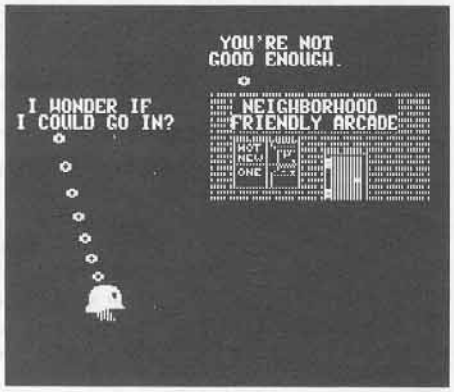
I've written a shell program (Listing 1) which you should type in with your assembler and save. Then, whenever you come up with a program which you could use as a DOSless utility, plop it into the shell and assemble it. If you are keying in the hex code directly, you'll have to precede it with a number-of-sectors byte (included in the hexdump accompanying this article). You can calculate it using LP - FP + 1, where LP is the high byte of the last page of the program, and FP is the high byte of the first page of the program. So, for example, if your program extends from $0801 to $0A36; $0A - $08 + $01 = $03, which would be the number-of-sectors value. If keying in by hand, or if your assembler doesn't do it for you, save the program with some memorable name. Next, run the Applesoft program, "BOOTMAKER" (Listing 3) which will put your routine onto a diskette. Normally, tracks $01 and $02 of an initialized diskette are taken up by DOS. Since the DOSless boot program can't use them, "BOOTMAKER" will add them to the VTOC, giving you 8,192 extra bytes of storage space.

I've included a sample DOSless utility to get you started, "COPYDOWN" (Listing 2). When run, it copies all of a 16K RAM card into main memory, ready to be examined.

To use a DOSless utility, simply boot the diskette. This can be accomplished in a number of ways, depending upon what software is running. Some protected programs re-boot if you press RESET. Unfortunately, many of them clear memory before doing so; maybe you'll get lucky. If not, you'll need a way to drop into the monitor, where you can use C600G to boot.

BootMaker

```
10 REM BOOTMAKER
20 REM BY ADAM LEVIN
30 REM COPYRIGHT 1987 COMPUTIST
40 REM TURNS A MACHINE LANGUAGE PROGRAM INTO
   A SELF-RUNNING DISKETTE.
50 REM
60 D$ = CHR$ (4 )
70 ONERR GOTO 360 :ERR = 0
80 REM CLEAR PROGRAM BUFFER (HGR ) AND LOAD
   PROGRAM.
90 HGR : TEXT : HOME : PRINT "^ ^ ^ ^ BOOT^
   MAKER^ FOR^ DOSLESS^ UTILITIES"
100 PRINT : PRINT "ENTER^ FILENAME^ OF^
   DOSLESS^ PROGRAM" : INPUT ":" ;F$ : IF F$
   = "" OR F$ = "?" OR F$ = "CATALOG" THEN
   PRINT D$ "CATALOG" : GOTO 100
110 PRINT D$ "BLOAD^ " F$ ",A8192" : REM BLOAD
   TO $2000.
120 REM INSURE THAT PROGRAM IS UNDER (4 ,096
   BYTES ) IN LENGTH.
130 NOS = PEEK (8192 )
140 PRINT : IF NOS > 16 THEN PRINT
   "NUMBER-OF-SECTORS^ MUST^ BE^ LESS^
   THAN^ 17!" :ERR = 1 : GOTO 360
150 REM READ IN BOOTMAKER'S SUBROUTINES AND
   DATA.
160 FOR LOC = 768 TO 819 : READ BYTE : POKE LOC
   ,BYTE : NEXT
170 REM INTERLEAVE PAGES OF PROGRAM BEFORE
   WRITING THEM TO DISKETTE. THEN , WHEN
   LOADED FROM DISKETTE AT BOOT , PROGRAM
   WILL BE CONTIGUOUS.
180 FOR LOC = 0 TO NOS - 1 : REM USE MONITOR'S
   -MOVE- ROUTINE TO SHIFT EACH PAGE. COPY
   PAGES FROM $2000-$2FFF TO $3000-$3FFF.
190 POKE 60 ,0 : POKE 61 ,LOC + 32 : REM 'FROM'
   ADDRESS
200 POKE 62 ,255 : POKE 63 ,LOC + 32 : REM
   'END' ADDRESS
210 POKE 66 ,0 : POKE 67 , PEEK (LOC + 804 ) +
   48 : REM 'TO' ADDRESS
220 CALL 799 : NEXT : REM MUST USE MACHINE
   LANGUAGE TO CALL -MOVE-.
230 REM GET VOLUME NUMBER FOR DISKETTE. THIS
   IS IMPORTANT SINCE FILES CAN STILL BE
   SAVE ON TRACKS 1 TO 35
240 PRINT : PRINT "ENTER^ VOLUME^ NUMBER^ OR^
   <RETURN>^ FOR^ #254" ; : INPUT ":" ;V$ :
   IF V$ = "" THEN V$ = "254"
250 V = VAL (V$ ) : IF V > 254 THEN 240
260 POKE 42347 ,96 : REM DISABLE SAVING OF A
   -HELLO- PROGRAM
```




```
270 POKE 44723 ,4 : REM ADD TRACKS 1 & 2 TO
   VTOC.
280 POKE 46922 ,76 : POKE 46923 ,0 : POKE
   46924 ,3 : REM PATCH -INIT- TO JMP TO
   BOOTMAKER'S ROUTINE.
290 PRINT : PRINT "INSERT^ A^ BLANK^
   DISKETTE,^ PRESS^ A^ KEY." : WAIT -
   16384 ,128 : POKE - 16368 ,0
300 REM INIT DISKETTE , WRITING PAGES AT
   $3000 - $3FFF TO TRACK 0 INSTEAD OF DOS.
310 PRINT : PRINT D$ "INIT" F$ ",V" ;V
320 REM RESTORE PATCHES TO THEIR ORIGINAL
   VALUES.
330 POKE 42347 ,76 : POKE 44723 ,12 : POKE
   46922 ,173 : POKE 46923 ,231 : POKE 46924
   ,183
340 PRINT : IF NOT ERR THEN PRINT "ALL^ DONE."
   : END
350 PRINT "PROGRAM^ ABORTED." : END
360 ERR = 1 : GOTO 330
370 REM DATA FOR MACHINE LANGUAGE ROUTINES.
380 DATA 169 ,16 ,141 ,224 ,183 ,141 ,225 ,183
   ,169 ,64 ,141 ,231 ,183 ,169 ,0 ,141 ,236
   ,183 ,169 ,15 ,141, 237 ,183 ,169 ,63
   ,141 ,241 ,183 ,76 ,147 ,183 ,160 ,0 ,76
   ,44 ,254
390 REM DATA FOR SECTOR DE-INTERLEAVE TABLE.
400 DATA 0 ,7 ,14 ,6 ,13 ,5 ,12 ,4 ,11 ,3 ,10
   ,2 ,9 ,1 ,8 ,15
```

CopyDown Hexdump

```
0800: 01 BD 88 C0 AD 88 C0 A9   $1A58
0808: D0 85 03 A9 10 85 01 A9   $DB3E
0810: E0 85 04 20 3E 08 AD 80   $370B
0818: C0 A9 D0 85 03 A9 20 85   $6216
0820: 01 A9 00 85 04 20 3E 08   $0389
0828: AD 82 C0 20 58 FC A2 00   $3C4D
0830: BD 56 08 10 06 20 ED FD   $F82C
0838: E8 D0 F5 4C 69 FF A0 00   $F9FF
0840: 84 02 84 00 B1 02 91 00   $FB8B
0848: C8 D0 F9 E6 01 E6 03 A5   $059A

0850: 03 C5 04 D0 EF 60 A0 A0   $04BA
0858: C4 CF D3 CC C5 D3 D3 A0   $204F
0860: B1 B6 CB A0 C3 CF D0 D9   $5E22
0868: C4 CF D7 CE 8D A0 C2 C1   $A8D8
0870: CE CB A0 B1 A0 A0 A0 C2   $99CE
0878: C1 CE CB A0 B2 A0 A0 A0   $43EF
0880: A0 CD C1 C9 CE 8D A4 C4   $4D30
0888: B0 AD A4 C4 C6 A0 A0 A4   $C9F0
0890: C4 B0 AD A4 C4 C6 A0 A0   $C9E4
0898: A4 C5 B0 AD A4 C6 C6 8D   $5BED
08A0: A0 A0 A0 A1 A0 A0 A0 A0   $3E75
08A8: A0 A0 A0 A0 A1 A0 A0 A0   $56E0
08B0: A0 A0 A0 A0 A0 A1 8D A0   $7F4F
08B8: A0 A0 D6 A0 A0 A0 A0 A0   $0469
08C0: A0 A0 A0 D6 A0 A0 A0 A0   $525E
08C8: A0 A0 A0 A0 D6 8D A4 B1   $BB26
08D0: B0 AD A4 B1 C6 A0 A0 A4   $FE7C
08D8: B2 B0 AD A4 B2 C6 A0 A0   $6270
08E0: A4 B3 B0 AD A4 B4 C6 8D   $C82F
08E8: 00                        $BF48
```

checksums

```
10  - $BADD     210 - $4C42
20  - $9B13     220 - $00E8
30  - $4D3B     230 - $62B6
40  - $AD92     240 - $CCDF
50  - $C899     250 - $120C
60  - $BB61     260 - $6A21
70  - $BAF7     270 - $FE33
80  - $09ED     280 - $4A40
90  - $D919     290 - $0FD6
100 - $017A     300 - $F609
110 - $A111     310 - $D8BE
120 - $0F12     320 - $36BC
130 - $7183     330 - $8CBC
140 - $AF11     340 - $7B1F
150 - $C78F     350 - $90F9
160 - $75B5     360 - $31AE
170 - $D313     370 - $E6AC
180 - $C10E     380 - $28A8
190 - $642B     390 - $E04F
200 - $44E3     400 - $F294
```


```
* DOSLESS UTILITIES - PROGRAM SHELL *
* BY ADAM LEVIN *

* ALLOWS EASY CREATION OF SELF-BOOTING PROGRAMS *
* WITHOUT DOS. *

.OR $0800 MUST ORIGINATE HERE!
.TF FILENAME

0800: 01 .DA /LAST-*+$0100 NUMBER OF SECTORS BYTE
0801: BD 88 C0 START LDA $C088,X TURN OFF DISK DRIVE

* ---------- *

0804- LAST .EQ * LABEL TO MARK END
```

```
*                    DOSLESS COPYDOWN                    *
*                      - BY ADAM LEVIN                   *
*                  COPYRIGHT 1987 COMPUTIST              *

*  COPIES THE CONTENTS OF A 16K RAM CARD TO MAIN MEMORY.  *

            .OR $0800    MUST ORIGINATE HERE!
*   THE NEXT LINE SAVES THE OBJECT CODE AS 'COPYDOWN'.    *
            .TF COPYDOWN
0800: 01                .DA /LAST-*+$0100 NO. OF SECTORS
0801: BD 88 C0          LDA $C088,X  TURNS OFF DISK DRIVE
*-------------------------------------------------------*
*          INSERT YOUR PROGRAM AFTER THIS LINE          *
*-------------------------------------------------------*

0000- RAMSLT    .EQ 0         16K RAM CARD SLOT
0000- MAIN      .EQ $00,$01   MOTHERBOARD RAM POINTER
0002- CARD      .EQ $02,$03   RAM-CARD RAM POINTER
0004- END       .EQ $04       HIGH BYTE OF LAST PAGE +1
C088- B1RAM     .EQ $10*RAMSLT+$C088 BANK 1/RAM READ/WRITE PROT
C080- B2RAM     .EQ $10*RAMSLT+$C080 BANK 2/RAM READ/WRITE PROT
C082- B2ROM     .EQ $10*RAMSLT+$C082 BANK 2/ROM READ/WRITE PROT
*-------------------------------------------------------*
*          COPY BANK 1 $D000-$DFFF TO $1000.            *
      BANK1.TO.DFFF
0804: AD 88 C0          LDA B1RAM    SET RAM CARD TO READ BANK 1
0807: A9 D0             LDA #$D0     STARTING ADDRESS
                                     (HIGH BYTE) IS $D0
0809: 85 03             STA CARD+1   SAVE IT IN PTR TO RAM CARD
080B: A9 10             LDA #$10     STARTING ADDRESS (HIGH)
                                     OF MAIN IS $10
080D: 85 01             STA MAIN+1   SAVE IT IN PTR TO MAIN MEMORY
080F: A9 E0             LDA #$E0     ENDING ADDR OF BANK 1 (+1)
0811: 85 04             STA END      SAVE IT IN 'END' LOCATION
0813: 20 3E 08          JSR CPYDWN   CALL THE CPYDWN ROUTINE
*-------------------------------------------------------*
*          COPY BANK 2 $D000-$DFFF TO $2000.            *
      BANK2.TO.FFFF
0816: AD 80 C0          LDA B2RAM    SET CARD TO READ BANK 2
0819: A9 D0             LDA #$D0     SAME FORMAT AS ABOVE
081B: 85 03             STA CARD+1
081D: A9 20             LDA #$20
081F: 85 01             STA MAIN+1
0821: A9 00             LDA #$00
0823: 85 04             STA END      ENDING ADDRESS PLUS 1
0825: 20 3E 08          JSR CPYDWN
0828: AD 82 C0          LDA B2ROM    SET RAM CARD TO READ ROM
082B: 20 58 FC          JSR $FC58    HOME CURSOR & CLR SCREEN
082E: A2 00             LDX #$00     PRINT DIAGRAM SHOWING THE
0830: BD 56 08  .1      LDA MSG,X    MEMORY MOVES
0833: 10 06             BPL .2
0835: 20 ED FD          JSR $FDED
0838: E8                INX
0839: D0 F5             BNE .1
083B: 4C 69 FF  .2      JMP $FF69    JUMP TO THE MONITOR
*-------------------------------------------------------*
083E: A0 00     CPYDWN  LDY #$00     COPIES MEM FROM CARD TO MAIN
0840: 84 02             STY CARD     SET LOW BYTE OF RAM CARD
                                     POINTER TO 0
0842: 84 00             STY MAIN     DITTO FOR MAIN MEM PTR
0844: B1 02     0       LDA (CARD),Y TAKE A BYTE FROM RAM CARD,
0846: 91 00             STA (MAIN),Y AND STORE IT IN MAIN MEMORY
0848: C8                INY          POINT TO NEXT BYTE
0849: D0 F9             BNE .0       LOOP IF STILL IN SAME PAGE,
084B: E6 01             INC MAIN+1   INC HIGH BYTES IF NOT
084D: E6 03             INC CARD+1
084F: A5 03             LDA CARD+1   COMPARE CURRENT PAGE WITH
                                     ENDING PAGE
0851: C5 04             CMP END
0853: D0 EF             BNE .0       LOOP IF NOT THERE YET,
0855: 60                RTS          RETURN IF DONE
0856: A0 A0 C4
0859: CF D3 CC
085C: C5 D3 D3
085F: A0 B1 B6
0862: CB A0 C3
0865: CF D0 D9
0868: C4 CF D7
086B: CE        MSG     .AS -/  DOSLESS 16K COPYDOWN/
086C: 8D                .HS 8D
086D: A0 C2 C1
0870: CE CB A0
0873: B1 A0 A0
0876: A0 C2 C1
0879: CE CB A0
087C: B2 A0 A0
087F: A0 A0 CD
0882: C1 C9 CE          .AS -/ BANK 1   BANK 2    MAIN/
0885: 8D                .HS 8D
0886: A4 C4 B0
0889: AD A4 C4
088C: C6 A0 A0
088F: A4 C4 B0
0892: AD A4 C4
0895: C6 A0 A0
0898: A4 C5 B0
089B: AD A4 C6
089E: C6                .AS -/$D0-$DF  $D0-$DF  $E0-$FF/
089F: 8D                .HS 8D
08A0: A0 A0 A0
08A3: A1 A0 A0
08A6: A0 A0 A0
08A9: A0 A0 A0
08AC: A1 A0 A0
08AF: A0 A0 A0
08B2: A0 A0 A0
08B5: A1                .AS -/  !       !       !/
08B6: 8D                .HS 8D
08B7: A0 A0 A0
08BA: D6 A0 A0
08BD: A0 A0 A0
08C0: A0 A0 A0
08C3: D6 A0 A0
08C6: A0 A0 A0
08C9: A0 A0 A0
08CC: D6                .AS -/  V       V       V/
08CD: 8D                .HS 8D
08CE: A4 B1 B0
08D1: AD A4 B1
08D4: C6 A0 A0
08D7: A4 B2 B0
08DA: AD A4 B2
08DD: C6 A0 A0
08E0: A4 B3 B0
08E3: AD A4 B4
08E6: C6                .AS -/$10-$1F  $20-$2F  $30-$4F/
08E7: 8D                .HS 8D
08E8: 00                .HS 00
*-------------------------------------------------------*
*       END MARKER USED TO DETERMINE NUMBER OF SECTORS  *
*-------------------------------------------------------*
08E9- LAST      .EQ *
```

an owner's review of...

Applied Engineering's

by Jerry D. Greer

Applied Engineering
P.O. Box 798
Carrollton, TX 75066
(214) 241-6060
prices: 256K $329; 512K $389;
768K $449; 1 meg $599

What is a Z-RAM?

Applied Engineering's Z-RAM board is a combination memory expansion and Z-80 microprocessor add-on for the Apple //c computer. It is available with 256K, 512K, 768K, or 1M of additional memory and fits inside the //c case. The //c computer is still the //c; the board does not change that. It does give you more opportunities to use it! The card is apparently 100% compatible with all software on the market that will run in the 65C02 microprocessor. All peripherals such as the mouse, external disks, modem, and printers are compatible with the card. The gang at AE has put most of their effort into making the Z-RAM a real aid to users of Apple's best selling and easy to use Appleworks.

The Z-RAM is also a very high speed solid state disk drive. It gives you the ability to load programs completely into memory. The result is a fantastic increase in speed. This RAM disk is compatible with ProDOS, DOS 3.3, Pascal, and CP/M.

The ability to run CP/M operating system software is one of the super features of the Z-RAM. With the onboard Z-80 microprocessor, you can run CP/M programs like Wordstar, dBase II, Turbo Pascal, Microsoft Basic, and more than 3000 other commercially available programs. There is a whole world of CP/M users groups out there and they have many public domain programs that are yours for the cost of a disk.

Two kinds of Z-RAM boards for the //c

When Applied Engineering first released the board for the //c, it was limited to 512K of memory. With the 128K onboard the motherboard, that gave you 640K of memory. That board is now loosely referred to as the Z-RAM I by the Applied Engineering group. Today, you can buy the Z-RAM II board that has a maximum of 1 meg (again in addition to the 128 already onboard). The Z-RAM II can be purchased with as little as 256K of memory and it can be upgraded in 256K increments to the 1 meg maximum. The Z-RAM II comes with a completely revised manual that should eliminate many of my criticisms of their very poor "first edition."

The only difference between the Z-RAM I and the Z-RAM II is in the total memory on the board. In other ways, they are based on nearly identical technologies. Keep in mind that Applied Engineering has discontinued their production of the Z-RAM I. They have assured me that they will continue to support purchasers who own it.

I would guess that you will start to see some very attractive discounts on the Z-RAM I as supplies are sold out. You may see many used ones available as owners choose to move up to the new 1 meg board. The Z-RAM I is a fine board and will be worth any discounted price. Applied Engineering says that they do not have any upgrade policy for owners of the now discontinued I's. However, they did state to me that registered owners will be given a 20 percent discount if they purchase the new Z-RAM II. Check it out; if you can do this, the 1 meg board will cost you $480. plus shipping. If you buy a used one, make sure that you get the software and the manual that go with it.

Why choose the Z-RAM?

Who knows? Maybe it was Steve Wozniak's smiling face in all the advertisements that helped me settle on the Applied Engineering board. Really, would the Woz put anything in his Apple that was not good for it? What really counts though is the way the product works when you get it all plugged in. That smiling face won't save you from a disaster if the product does not live up to its claims. Whatever my reasons, an Applied Engineering 512K Z-RAM ended up in my Apple //c and since then, I haven't been able to do without it.

The expanded memory and CP/M really sold me on the Z-RAM. While I looked around for the right board, the price continued to drop a little each month and finally, I made my move. Will it drop more? Very likely, yes. But you must determine when it is right for you and then make your move.

What's in the box?

When you buy a Z-RAM board, you get the printed circuit board with chips, the Z-RAM User Manual and the software. One of the disks is the CP/M software and the second is the Super Appleworks Desktop Expander. Hidden "inside" the Expander disk are the RAMdrive programs for DOS 3.3 and ProDOS. The Z-RAM II has a five year warranty.

How much memory do you actually get?

Easy to answer! Just add the basic //c's 128K of memory to that memory on the board. That gives you the total. However, there is a difference between the total memory and the usable memory. That difference is caused by differences in the Expander, ProDrive or RAMdrive software that is supplied by the manufacturer. Software revisions have required small amounts of additional memory. The available memory may go down but the ease of use of the software will probably go up. For example, with Appleworks, some early advertisements stated that you would have 425K of usable memory with the 512K board. Later, this dropped to 413K. Now it is at 394K. The benefit offsetting this reduction of memory is that the programmers have improved the software for Appleworks. You can now load Appleworks entirely into memory and work about two or three times as fast.

Z-RAM Memory Expansion

Installation

I got a kick out of the advertisements that told me that I could install the board in ten minutes using only a screwdriver, and that it took "slightly longer without". This is probably true after you have done it 10 times! My advice is to allow about an hour if you are very cautious about opening the case. The popping and cracking of plastic made me wonder what I had got myself into, but encouraged by the manual, I went on.

Once the case was off and the disk drive was laying safely aside, things went better. Part of the procedure is to pull out the 65C02 and the MMU from the //c motherboard and replace them into two new sockets in the Z-RAM board. Prying out those chips gave me another excuse to pause. I wandered around my Hackin' Shack a few more minutes before I found the courage to proceed. Again, though, it proved to be easy and finally, the chips took up their new places on the Z-RAM board.

The next step requires the most care. The pins on the Z-RAM board are lined up with the now empty CPU and MMU sockets on the //c motherboard. The Z-RAM assembly is gently pushed down. Because pin alignment is critical, it pays to take it easy here. None of my pins were bent in the process and the new board easily took its place in the case.

The Apple //c is assembled with a supporting truss assembly under the keyboard. Clearance is very close and the truss had to come out before I could fit the Z-RAM board and the keyboard back into the case. The truss is held in place with two tiny expansion pins and easily comes off. If you use your keyboard with a gentle touch, this should never cause a problem.

The Z-RAM board comes with a jumper wire that must be attached to a chip on the motherboard, right under the built-in drive. The attaching hook is a spring loaded device that fits a little too tight between the drive and the motherboard, so I used a razor blade to trim the tabs on this hook. That made it fit much better. Trimming should eliminate all possibility of a broken trace on the motherboard. Otherwise, careless placement of the hook device between the drive and the motherboard could cause serious problems.

Problems and Distractions

Applied Engineering has a very fine product and it is too bad that their first manual left so much to be desired. Their first edition looked cheap and was very incomplete. While it was irritating, it was not a serious problem for users. It should not discourage anyone from accepting a Z-RAM I board. Applied Engineering has issued a completely revised manual for their new Z-RAM II, which will go far in improving their image!

The CP/M that comes in the package is actually CP/AM, a version modified by Applied Engineering for their use. Anyone who has used CP/M very much will see that there are differences between it and the original CP/M. There is, or was, an obscure bug in one file transfer routine that I discovered. The AE representative with whom I spoke was genuinely interested in the problem and took notes for their programming staff. My experience with their service staff makes me believe that the problem will be corrected in the next update.

And speaking of updates! I don't like to have to purchase software updates! The cost of a disk and mailing (maybe $2) doesn't get to me but paying $12 for a corrected copy of something that I have purchased just doesn't seem fair. You also need to keep in mind that Applied Engineering has no formal method of informing individual customers about updates in their software. Dealers are advised and you can get current information from your dealer. Direct purchasers can call the company periodically to see what the status is.

What good is it?

The Z-RAM board is basically designed to improve your ability to use Appleworks. With 512K on the Z-RAM board, I have a desktop that will hold 394K of files. Because Appleworks is now completely loaded into memory, it will run much faster than one accessed from the disk drive. With the Z-RAM, you can work faster with only the built-in //c disk drive than others can using two disk drives.

With this much memory on the desktop, it is easy to make files that exceed the capacity of a single floppy disk (about 143K.) The software that comes with the board will segment large files for Appleworks and will automatically save them off to multiple disks. Your maximum file size is limited only by the size of your desktop memory.

When using Appleworks, the software will create a print spooler in /RAM. That is a real time saver and allows you to continue work while your files slowly run out on your printer. Versions 5.2 or later of the Expander software permits you to purge the spooler if you want to abort a print run. Earlier versions of the Expander software would not let you abort printing a document without bringing down your entire system and starting it all over again.

My move into CP/M has been gradual. The potential is exciting and I am starting to see more that can be done in that area. The CP/M system is fast and being different from our regular Apple systems, is fun and challenging to use. The software provided is enough to get you interested.

In my opinion, Ramdrive (for DOS 3.3) and Prodrive (for ProDOS) are the "hidden jewels" in the deal. This software permits you to set up very fast electronic "drives" in RAM and use them for increased speed and ease of operation with practically any program.

I'm glad I did it

My experience with the 512K Z-RAM has been excellent. The board has worked perfectly and the software has delivered what it is supposed to. The response to me by Applied Engineering has been extremely prompt and very courteous. The board layout and construction appear to be very well engineered. If you choose the Applied Engineering Z-RAM board for your //c, you won't go wrong. They have got me hooked. What is next you ask? Why, a 1 meg board of course!

a routine investigation of...

The Joystick

by R. Wideman

The joystick is a popular input device used in many programs. Whether it be a game or an application program, the joystick gives a user flexibility in control of images and icons, or selection of items from a menu. The joystick frees the user from the oftentimes cumbersome aspects of the keyboard-keys to remember, typing errors to overcome, and a dependence on the keyboard's rather permanent location. As programmers, you may wish to consider adding to your programs the option of using a joystick for input. There are a number of ways to access the benefits of a joystick, and a few will be discussed here that are implemented with assembly language.

The first place most beginning programmers use their joystick resources is with BASIC. This is accomplished through the use of the PDL(x) command, where x stands for either paddle 0 or paddle 1. This is quite an easy command to include in a BASIC program and there is not much to be said of it. The routine that actually accomplishes this task for BASIC is in Apple ROM at $FB1E. A look at this routine will reveal some basics of joystick programming.

```
FB1E:AD 70 C0 LDA $C070   trigger joystick
FB21:A0 00    LDY #$00    Reset counter
FB23:EA       NOP         Delay
FB24:EA       NOP         Delay
FB25:BD 64 C0 LDA $C064,X Read pdl 0 or 1
FB28:10 04    BPL $FB2E   If +, all done
FB2A:C8       INY         Update counter
FB2B:D0 F8    BNE $FB25   Keep reading
FB2D:88       DEY         Over 255, fix
FB2E:60       RTS         Return to caller
```

This routine is implemented by loading the X register with 00 for paddle zero or 01 for paddle one, and then executing a JSR $FB1E. The value of the paddle will be returned in the Y register. The routine works by first setting the joystick trigger to prepare the joystick hardware, and then clearing the Y register counter for a new reading. After a small delay, which allows the hardware to catch up, the selected paddle is read until the values go positive (<128). The number of times a negative value is read (>=128) is counted in the Y register. This number can range from 0 to 255. Once a positive value is read or the Y register counts beyond 255, the routine terminates and returns to the calling program. For some programs, this routine will be sufficient for intended purposes. By using it, you could save 17 bytes of memory. But there are drawbacks to using this routine.

You can only read one paddle at a time. To read both paddles would require re-loading the X register and calling the routine again. And neither of the paddle buttons are read. To do that requires polling $C061 and $C062 for negative values, for buttons zero and one respectively. And there is yet a more subtle problem with the routine. Its execution time

Joystick Reader Source Code

```
*--------------------------------------------------------------------*
*                        Joystick Routine                            *
*                        By R. Wideman                               *
*--------------------------------------------------------------------*

              .OR $300
              .TF OBJ.JOYSTICK READER

05-   PAD0    .EQ $05          Paddle zero
06-   PAD1    .EQ $06          Paddle one
07-   BUT0    .EQ $07          Button zero
08-   BUT1    .EQ $08          Button one

0300: A9 00     PREAD  LDA #$00        Clear out previous values
0302: 85 05            STA PAD0
0304: 85 06            STA PAD1
0306: 85 07            STA BUT0
0308: 85 08            STA BUT1
030A: A2 7F            LDX #$7F        Highest maximum value = 127
030C: AD 70 C0         LDA $C070       Reset trigger
030F: AD 64 C0  LOOP   LDA $C064       Read paddle zero
0312: 29 80            AND #$80        High-bit set?
0314: 0A               ASL             Put high-bit into carry
0315: 2A               ROL             Make 00 or 01
0316: 65 05            ADC PAD0        Update paddle value
0318: 85 05            STA PAD0        Store new value
031A: AD 65 C0         LDA $C065       Read paddle one
031D: 29 80            AND #$80        High-bit set?
031F: 0A               ASL             High-bit into carry
0320: 2A               ROL             Make 00 or 01
0321: 65 06            ADC PAD1        Update value
0323: 85 06            STA PAD1        Store new value
0325: CA               DEX             Have we timed out yet?
0326: D0 E7            BNE LOOP        No, keep reading
0328: AD 61 C0         LDA $C061       Timed out, read button zero
032B: 10 02            BPL NXBT        Positive value=not pushed
032D: E6 07            INC BUT0        It was pushed, set flag
032F: AD 62 C0  NXBT   LDA $C062       Button one pushed?
0332: 10 02            BPL PDNE        Positive value=not pushed
0334: E6 08            INC BUT1        It was pushed, set flag
0336: 60        PDNE   RTS             All done, return to caller
```

varies with the value of the paddle read. The larger the value read, the longer the time needed for the routine to execute. It is actually sometimes possible to tell if a program, such as a game, is using the ROM paddle routine because joystick controlled images will move faster or slower depending upon the joystick values. This can be a very undesirable side effect in crucial situations!

To eliminate these drawbacks, a joystick routine should meet the following criteria:

1) Execute in a constant amount of time, independent of the readings.

2) Read both paddle zero and paddle one at the same time.

3) Read both buttons, although this could be optional.

The joystick routine presented along side (as source code) and at the end (as a hexdump) of this article will satisfy these requirements. The routine is shown as starting at $300 but is completely relocatable and therefore will work anywhere in memory that you care to put it.

After calling this routine, the joystick values will be in the zero page locations indicated at the beginning of the listing. Zero page addresses were used because they speed up the execution. It should also be noted that any time a branch (BNE, BPL, etc.) has to cross a page boundary in memory, execution time will be increased. A routine, such as this one for the joystick, should be placed entirely within a single memory page to avoid such delays.

For each call, the routine will take exactly the same amount of time to execute, unless a button is pressed. This consistency is provided by the main loop which is controlled by the X register. X is loaded with 127 and decremented to 0 each time to ensure the constant execution time.

However, since both paddles are being read, their maximum value is approximately cut in half. This maximum value is reflected in the value that X is loaded with. Any higher value for X would be superfluous since no more worthwhile values would be read from the paddles. This could be a drawback if your program requires more specific readings. Most programs, though, use the joystick values to indicate a change in direction rather than specific positions. Again, a counter is used to indicate the final paddle values. It counts the number of negative values read and will range from 0 to approximately 127.

Joystick Reader Hexdump

```
0300: A9 00 85 05 85 06 85 07   $B0CD
0308: 85 08 A2 7F AD 70 C0 AD   $330A
0310: 64 C0 29 80 0A 2A 65 05   $B339
0318: 85 05 AD 65 C0 29 80 0A   $5AD0
0320: 2A 65 06 85 06 CA D0 E7   $34AE
0328: AD 61 C0 10 02 E6 07 AD   $4E03
0330: 62 C0 10 02 E6 08 60      $7ACE
```

softkey for...

Arcade Boot Camp

by Jim S. Hart

Penguin Software
830 4th Ave.
P.O. Box 311
Geneva, IL 60134
(312) 232-1984
$19.95

Requirements:
Arcade Boot Camp original disk
Super IOB
A blank disk
A fast DOS (optional)

Browsing around in one of the local computer stores recently, I came upon a program by the name of "Arcade Boot Camp". Noting that the publisher was Penguin software (which meant high quality programs at an affordable price) I decided to buy it and try it out. It turns out that it is really not a single game but several mini-arcade style games that are meant to increase your arcading abilities. Among the different scenarios are car driving, target shooting, maze navigating, and chopper flying. Everything about this disk is to my liking except that old bugaboo - copy protection.

My EDD III could copy it, but that also copied the protection. It was time for a round of deprotecting this disk. Upon booting, the good ol' familiar Applesoft prompt came up and then the game loaded in. If you see the prompt when booting it usually means that a somewhat normal DOS is in operation and you may be able to use a regular DOS on the deprotected disk. The Swap Controller has a good chance of working too (with the hope that there weren't any nasty nibble counts hiding on the disk).

Sometimes, however, there is more work to be done such as taking out a nibble count. Or maybe you have to use their DOS because the programs on disk use non standard routines built into it. This is just food for thought.

For those who are more interested in the protection scheme, the disk has data headers of DE AA AB and address and data epilogues of DA AA. On top of this, the address headers alternate between D5 AA 96 on even tracks (0, 2, 4...) and D4 AA 96 on odd tracks. This is a typical trick on Penguin programs.

It turns out that Arcade Boot Camp is easily deprotected using the controller below or the Swap controller. Follow the steps below to get your original into normal format:

1) Initialize the blank disk and delete the hello program. If you have a fast DOS, use it to initialize the blank.

```
INIT HELLO
DELETE HELLO
```

2) Install the controller below into Super IOB. The controller will skip the DOS tracks, leaving the normal DOS on your INITialized disk alone.

3) RUN Super IOB and copy the original to the blank. When the copying process is done, the copy will be deprotected and you'll be able to put away your original in a safe place.

controller

```
1000 REM  ARCADE BOOT CAMP
1010 TK = 3 :LT = 4 :ST = 15 :LS = 15 :CD = WR
     :FAST = 1
1020 GOSUB 490 :T1 = TK :LT = TK + 1 : RESTORE
     : GOSUB 170 : GOSUB 2000
1025 GOSUB 610 : IF PEEK (BUF ) < MB AND LT < >
     35 THEN LT = LT + 1 :TK = TK + 1 : GOSUB
     2000 : GOTO 1025
1030 GOSUB 230 :TK = T1 :LT = 35 : GOSUB 490 :
     GOSUB 610 : IF PEEK (TRK ) = LT THEN 1050
1040 TK = PEEK (TRK ) :ST = PEEK (SCT ) : GOTO
     1020
1050 HOME : PRINT "COPYDONE" : END
2000 POKE 47445 ,212 + (TK / 2 = INT (TK / 2 ) )
     : RETURN
5000 DATA 218 ,170 ,218 ,170
```

controller checksums

```
1000  - $356B     1040  - $E594
1010  - $E357     1050  - $85E3
1020  - $6306     2000  - $3863
1025  - $51FF     5000  - $AE83
1030  - $2C8C
```

Printer Pixit

by Kevin Sartorelli

Requirements:
Pixit disk
One blank disk

Below is a method of deprotecting Pixit, followed by a method of modifying the interface drivers to work with any "funny" interface card you may have.

Deprotecting Pixit

The Pixit disk is protected by having its shape tables and other data on protected tracks 1 to $B. Once this data is in memory it is a simple matter to save it out to a normal disk as binary files.

Initialize a blank disk with the following greeting program:

```
10 ONERR GOTO 40
20 TEXT : HOME : POKE 230,64 : HCOLOR= 0 :
    HPLOT 1,1 : CALL 62450 : HGR2 : PRINT
    CHR$ (4) "MAXFILES1" : PRINT CHR$ (4)
    "BLOADTITLE.PIC"
30 PRINT CHR$ (4) "BLOADTABLE^ 8A00" : PRINT
    CHR$ (4) "BLOADTABLE^ 3C00" : PRINT
    CHR$ (4) "RUNSTART"
40 TEXT : HOME : POKE 216,0 : NEW
```

INIT PIXIT

Now, boot up the original Pixit disk and press [C] when the title picture has appeared. The computer will continue to work for a while and then stop with a beep. The wanted tables are now in memory.

Type **TEXT** so you can see what you are typing.

To save the tables insert your initialized disk (don't boot it!) and type:

BSAVE TITLE.PIC,A$4000,L$1FF8
BSAVE TABLE 3C00,A$3C00,L$400
BSAVE TABLE 8A00,A$8A00,L$E00
BSAVE DEMO TABLE,A$6000,L$2A00

Now copy all the files from the original disk to your copy. This can be done with FID, Copy][Plus or any file copy program.

Lastly we need to modify the demo program to load in the demo table when the demo is run (if we load the demo table each time the disk is booted it takes longer - the file is 44 sectors - and the demo is not run very often).

Unlock the demo file and load it into memory with:

UNLOCK DEMO1
LOAD DEMO1

Type in the following line:

PRINT CHR$ (4)"BLOAD DEMO TABLE"

Save the modified file and lock it:

SAVE DEMO1
LOCK DEMO1

Your deprotected copy of Pixit is now ready to use.

Modifying the interface driver

When Pixit is booting, pressing "P" will bring up a printer configuration menu. However, Pixit has a very limited choice of printers (9 of them) and interfaces (14 possible cards). I found that the card I have does not work with any of the available choices, so I modified Pixit as follows.

The files to be modified to change the interface driver are PC (which is the menu) and PCM (which contains the available drivers).

To modify the file PC do the following from BASIC:

UNLOCK PC
LOAD PC
LIST 11000-11010

You should see the following lines:

```
11000 DATA INTERFACE ,5 ,1 ,0 ,0 ,APPLE^
    PARALLEL ,8 ,1 ,0 ,0 ,APPLE^ SUPER^
    SERIAL ,9 ,1 ,2 ,0 ,DUMPLING^ GX ,10 ,1
    ,12 ,0 ,DUMPLING^ 64 ,11 ,1 ,14 ,0
    ,EPSON^ APL ,12 ,1 ,0 ,0
11010 DATA GRAPPLER ,13 ,1 ,6 ,0 ,K-T^
    PARALLEL ,14 ,1 ,0 ,0 ,MICROBUFFER ,8
    ,21 ,4 ,0 ,PKASO ,9 ,21 ,8 ,0 ,PRINTMAX
    ,10 ,21 ,0 ,0 ,SSM^ PARALLEL ,11 ,21 ,10
    ,0 ,TYMAC ,12 ,21 ,0 ,0 ,VERSA^ PARALLEL
    ,13 ,21 ,0 ,0 ,WIZARD^ IPI ,14 ,21 ,0 ,0
```

Drivers

The numbers after the card name are:

1) VTAB to print name

2) HTAB to print name.

3) Interface number

4) Unused (used for the printer selection).

We are going to replace the Dumpling 64 card position with our own name. To insert the name of your card just change the name in line 11000. For example, if your card is called "FREDS SPECIAL 4" line 11000 would look like:

```
11000 DATA INTERFACE ,5 ,1 ,0 ,0 ,APPLE^
    PARALLEL ,8 ,1 ,0 ,0 ,APPLE^ SUPER^
    SERIAL ,9 ,1 ,2 ,0 ,DUMPLING^ GX ,10 ,1
    ,12 ,0 ,FREDS^ SPECIAL^ 4 ,11 ,1 ,14 ,0
    ,EPSON^ APL ,12 ,1 ,0 ,0
```

Save the modified file and lock it:

SAVE PC
LOCK PC

Now we get to the nitty gritty; the modification of the PCM file. Before you can modify this file, you MUST know how to output a byte directly to your printer via machine language. Below is an example of a routine that will work on a printer card that has the printer status found in bit 7 of $C0n0 (where n is the slot number):

```
3000- PHA          Save the output char.
3001- LDA $C080,X  Get printer status
3004- BMI $3001    If printer not ready
3006- PLA          Restore output char.
3007- STA $C081,X  Send it to printer
300A- STA $C080,X
300D- STA $C081,X
3010- RTS          Byte printed
```

The above routine assumes that upon entry, the x register contains the slot number * 16 of the printer card. However, Pixit does not furnish its printer drivers with this information. It is therefore required to modify the routine to use absolute addresses. Below is the routine modified for the same card in slot 1 (and only 1).

```
3000- PHA          Save the output char.
3001- LDA $C090    Get printer status
3004- BMI $3001    If printer not ready
3006- PLA          Restore output char.
3007- STA $C091    Send it to printer
300A- STA $C090
300D- STA $C091
3010- RTS          Byte printed.
```

Now that we have created a routine to output a byte to our printer, we need to make it a permanant part of our deprotected Pixit disk. The file that we must include our driver in is called PCM. This file loads in at $8000. Load in in now.

BLOAD PCM

The first part of this file is involved with getting the new configuration values from the BASIC program PC and moving the appropriate dump type down in memory to $285, moving the appropriate printer codes to $2B6 and moving the appropriate interface driver to $267.

The current interface driver is held in a file called MC5. It would be possible to place the new interface driver directly into this file starting at $267 but it would mean that when Pixit was configured again, our driver would be lost.

The Dumpling 64 interface driver (in the PCM file) is in memory at $8256. To see what we will be replacing, type:

CALL-151
8256L

You should now see a screenful of code starting with:

```
8256- 48        PHA
8257- AD 91 C0  LDA  $C091
825A- 29 02     AND  #$02
825C- D0 F9     BNE  $8257
```

Now, let's type in our new driver. The actual code you type may be different if you came up with a different driver.

8256:48 AD 90 C0 30 FB 68 8D 91
825F:C0 8D 90 C0 8D 91 C0 60

To see if it has been entered correctly type **8256L** and you should see:

```
8256- 48        PHA
8257- AD 90 C0  LDA  $C090
825A- 30 FB     BMI  $8257
825C- 68        PLA
825D- 8D 91 C0  STA  $C091
8260- 8D 90 C0  STA  $C090
8263- 8D 91 C0  STA  $C091
8266- 60        RTS
```

Since we are replacing the last driver in the file, you can make your driver just about as long or short as you like. When saving the file, use a length parameter that is address of the last byte (of your driver) plus one, minus $8000. In this case, we would use $8266+1-$8000, or $267 bytes.

We now have to save the modified file back to disk (remembering to unlock it first of course).

UNLOCK PCM
BSAVE PCM,A$8000,L$267

Remember to use the correct length depending upon how long your driver is. Pixit has now been modified to use your interface driver instead of the Dumpling 64 driver.

To check that the new driver works, boot the modified Pixit disk and type "P" while the disk is booting. You should be greeted with the available printers. Pick your printer, and then pick the new interface card from the menu.

When the program has configured the disk you will be left in BASIC. Type RUN PIXIT to enter Pixit. Enter the picture editor and see if it will print the picture in memory. If it does, all is well.

If not, something has gone wrong. Check the original code at $8256 to ensure that your Pixit is the same version as mine, you may need to change the location you patch. Check that you typed in the new interface driver correctly, and check that your new driver is the correct one.

Using the above, you should be able to get Pixit working fine through any interface card you may have. Happy printing!

Goonies

by Clay Harrell

Datasoft
19808 Nordoff
Chatsworth, CA 91311

Requirements:
COPYA from the DOS 3.3 System Master
A sector editor
A blank disk
Original Goonies or Zorro disk

Optional:
The Senior PROM v2.0
Super IOB 1.5

Both Goonies and Zorro from Datasoft are good graphic games that use very similar protection, but each with their own twist.

The first thing I always do when deprotecting a program is to get the program into a normal DOS 3.3 format. This way sector edits and easy copies can be made (even if the program doesn't run). After the disk is in a normal format, then you can remove the protection.

To do this, I try the easiest steps first: Using the Senior PROM's copy routines, I defeat the DOS error checking and try copying the disk. If you don't have a Senior PROM, boot your DOS 3.3 System Master, and at the BASIC prompt type:

```
CALL -151
B942:18
RUN COPYA
```

Try and copy the original disk. This small modification to DOS ignores minor disk alterations performed by many software publishers. Using the Senior PROM, I found that track $00 would copy fine, but the other tracks would not copy in this manner.

The next step was to use the Senior PROM's nibble read to examine the unreadable tracks. This will tell us how the other tracks are formatted. Nibble reading track $01 revealed the following information:

```
Data prologue bytes:          $A9 BA F7
Address prologue bytes:       $CA EE DD
Data and Address epilogues:   $DE AA
```

Zorro and Goonies had the usual 342 bytes of data between Data prologue and epilogue bytes. This denotes the disk is in 6+2 format, or a normal 16 sectors per track.

Only tracks $00-13 contained data on Zorro. For Goonies, only tracks $00-20 contained data. The rest of each disk is blank.

This information is invaluable when trying to convert the disk to normal DOS format. The prologue bytes are road markers to DOS: the Address prologue bytes tell DOS the next eight bytes denote what track and sector is being read. The Data prologue bytes tell DOS that the next 342 bytes is the actual data. The epilogue bytes are just insurance bytes, telling DOS "it ends here."

The Address and Data prologue bytes on both Goonies and Zorro were considerably different from those on a normal DOS disk. This makes it difficult for a copy program to find what track it is trying to read (Address prologue bytes), and where the data starts for the sector (Data prologue bytes).

Using the Senior PROM's "Alter prologue Byte" option, I was able to easily copy the original Zorro and Goonies disks to blank disks in normal DOS 3.3 format. Track $00 was copied first, as it was unprotected. Then tracks $01-13 (on Zorro) and tracks $01-20 (on Goonies) were read with $A9 BA F7 and $CA EE DD Data and Address prologue bytes, respectively, and written in normal DOS format.

You can also convert the disk with COPYA. Here is the procedure:

1) Boot your DOS 3.3 System Master disk, and initialize a blank disk by inserting a blank disk in the drive and typing:

```
INIT HELLO
```

2) Then run COPYA from your DOS 3.3 System Master by typing:

```
RUN COPYA
```

3) When COPYA is done loading, and is at the slot prompt, type [C] to stop the program.

4) Now we need to modify COPYA so it will only copy tracks $01-13 (for Zorro), or tracks $01-20 (for Goonies). Type:

```
CALL-151
302:14 (Zorro only)
35F:14 (Zorro only)
302:21 (Goonies only)
35F:21 (Goonies only)
2B0:A9 00 8D D1 02 8D D2 02 60
2DC:20 B0 02 A9 FF
2E6:F8
3D0G
```

(delete lines 246 through 250, and 70)

```
DEL 246,250
70
```

Tell COPYA to write with the normal prologues and epilogues:

```
258 POKE 47335,213: POKE 47345,170:
    POKE 47356,173: POKE 47445,213:
    POKE 47455,170: POKE 47466,150
```

Tell COPYA to read using the altered prologues and epilogues:

```
197 POKE 47335,169: POKE 47345,186:
    POKE 47356,247: POKE 47445,202:
    POKE 47455,238: POKE 47466,221

RUN
```

5) Copy the original Zorro and/or Goonies disk to a blank disk.

6) Run your favorite sector editor and copy track $00 from the original Zorro and/or Goonies disk to track $00 of the disk you just copied in the above step.

Both disks are now in normal DOS format. If you boot this disk, of course it won't run since we haven't removed the protection yet.

The first step in removing the protection is to tell the protected DOS that it should use normal DOS prologue bytes instead of the protected ones. Using the Senior PROM, I interrupted the Zorro program while it was trying to read the disk. Using the Disassemble at Program Counter feature of the Senior PROM (it disassembles at the location the program was interrupted), I found the routines to change were at $D118-D139 and $D180-D1A7 in Bank 2 of the Bank Switched Memory (upper 16K). Bytes $D123, $D12D, $D138 needed to be changed from $A9 BA F7 (the old Data prologue bytes) to $D5 AA AD (normal DOS 3.3's Data prologue bytes). Likewise, bytes $D191, $D19B, $D1A6 needed to be changed from $CA EE DD (the old Address prologue bytes) to $D5 AA 96 (normal DOS 3.3's Address prologue bytes).

Where Zorro used the upper 16K to run their program loader, Goonies uses page $03 and text page 1 (memory from $400-7FF). Both memory

& Zorro

areas are tricky to work with, but text page 1 is even more so. The reason being if you reset out of Goonies, text page 1 is automatically destroyed, since anything printed on the screen will wipe this memory.

If you have a Senior PROM, there is no problem dealing with text page 1 in Goonies; just press the NMI button and use the "G" command to set the location to examine, and then press "D" to disassemble at this location.

This will give a disassembly of volatile memory $400-7FF if desired. Using this technique, it was easy to discover the start of the Goonies protection code:

```
0318-  A0 20        LDY #$20
031A-  88           DEY
031B-  F0 61        BEQ $037E
031D-  BD 8C C0     LDA $C08C,X
0320-  10 FB        BPL $031D
0322-  49 A9        EOR #$A9
  (1st byte of the Data prologue)
0324-  D0 F4        BNE $031A
0326-  EA           NOP
0327-  BD 8C C0     LDA $C08C,X
032A-  10 FB        BNE $0327
032C-  C9 BA        CMP #$BA
  (2nd byte of the Data prologue)
032E-  D0 F2        BNE $0322
0330-  A0 56        LDY #$56
0332-  BD 8C C0     LDA #$C08C,X
0335-  10 FB        BPL $0332
0337-  C9 F7        CMP #$F7
  (3rd byte of the Data prologue)
0339-  D0 E7        BNE $0322
```

It was pretty obvious that the code for both Goonies (in text page 1) and Zorro (in BSM) had to be on track $00 since it was the only track the program could load from our copied disk (remember it was never protected). Using the Senior PROM sector editor, I searched track $00 for the code, and it could not be found! There had to be some encoding routine making a simple search and edit impossible.

Doing some minor boot-code tracing led to some interesting information. The first instruction on track $00, sector $00 was a JMP $8A2. Everything past this code looked like garbage. The code at $8A2 blanks out hi-res page 1, and then jumps indirectly through location $99 (to be sneaky, no doubt) to $8D8. The routine at $8D8 is interesting: this code is a routine used to unencode the rest of page $08. After unencoding page $08, it jumps back down to $814 and loads in the program loader across text page 1 and page 3.

It turns out the routine at $8D8 is the same type of routine that is used to unencode the Goonies disk. Using this code, I wrote the following routine to unencode the bytes on the disk to what they were in memory:

```
900: A9 FF      LDA #$FF
902: 85 06      STA $06
904: E6 06      INC $06
906: A5 06      LDA $06
908: 48         PHA
909: 4A         LSR
90A: 68         PLA
90B: 6A         ROR
90C: C5 07      CMP $07
90E: D0 F4      BNE $0904
910: A5 06      LDA $06
912: 20 DA FD   JSR $FDDA
915: 60         RTS
```

Using this routine, I could load the byte I wanted in location $07, type '900G", and the byte that it corresponded to on the disk would be printed. From this I was able to determine the following sector edits to change both the Zorro and the Goonies operating system to read a normal DOS 3.3 disk:

```
Track $00, sector $02
      byte $23 from $53 to $AB
      byte $2D from $75 to $55
      byte $38 from $EF to $5B
      byte $91 from $95 to $AB
      byte $9B from $DD to $55
      byte $A6 from $BB to $2D
```

Now the copy of Goonies would boot and run! But the Zorro has some additional protection. A few moments after booting the Zorro disk, there was another disk check routine verifying the original disk. Using the Senior PROM to interrupt the program and Disassemble at the Program Counter, the routine was found at $D4D5 in Bank 2 of the Banked Switch Memory. This routine was very easy to defeat by putting a "Return from Subroutine" at the very beginning of the routine. Of course the Return from Subroutine instruction had to be encoded using the previous routine before written to the disk. Here is the sector edit necessary to defeat the routine:

```
Track $00, sector $05
      Byte $D5 from $7B to $C0
```

The programs may be softkeyed using the procedures described above, or with the Super IOB controller given here. The controller will ask you which program you are deprotecting and make the additional sector edits for Zorro.

controller

```
1000 REM ZORRO AND GOONIES
1010 TK = 0 :LT = 1 :ST = 15 :LS = 15 :CD = WR
     :FAST = 1
1015 HOME : VTAB 9 : INPUT "G)OONIES^ OR^
     Z)ORRO=>" ;B$
1017 GOSUB 490 : GOSUB 610 :LT = 20 + 13 * (B$
     = "G" ) :TK = 1 : GOTO 1025
1020 GOSUB 490
1025 RESTORE : GOSUB 190 : GOSUB 210 : GOSUB
     610 : IF TK = 1 THEN TK = 0 : GOSUB 2000
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 "THAT'S^ ALL" : END
2000 FOR A = 1 TO 6 : READ B ,C : POKE 13312 + B
     ,C : NEXT
2010 IF B$ = "Z" THEN POKE 12757 ,192
2020 RETURN
5000 DATA 202 ,238 ,221
5010 DATA 169 ,186 ,247
5020 DATA 35 ,171 ,45 ,85 ,56 ,91
5030 DATA 145 ,171 ,155 ,85 ,166 ,45
```

controller checksums

```
1000  - $356B     1050  - $3ED6
1010  - $EA41     2000  - $E445
1015  - $0C91     2010  - $81F5
1017  - $9C28     2020  - $058A
1020  - $1302     5000  - $A289
1025  - $3E57     5010  - $D6B4
1030  - $0BCE     5020  - $6D8A
1040  - $B79B     5030  - $973D
```

Coveted Mirror

by Marc Batchelor

Requirements:
Apple][Plus or better
Super IOB

I was reading some of the back issues of COMPUTIST that I have when I came upon the softkey for Transylvania and The Quest in COMPUTIST No. 13. In reading through the article a bit, I learned that the controller listed at the end of the article was supposed to work on all Penguin releases. "Wow!" I thought, "Now I can finally backup my Coveted Mirror disk." I then eagerly typed in the controller and installed it into Super IOB v1.5. Much to my dismay, the controller did not work. It kept getting the dread "Drive Error." Just for laughs, I tried the controller in the oldest version of Super IOB I could find. Believe it or not, it worked!

I then scrutinized the controller to figure out why it wouldn't work with the new Super IOB but it would work with the old one. After a while, the answer became clear. It was because the "Normalizer" routine of the new versions of Super IOB (v1.2 and greater) fix the ending markers. This was missing from the old Super IOB since it didn't have a "Altered Ending Markers" routine. In line 1110 of the controller presented with that article, the reference to the "Normalizer" routine (line 230) counts on the fact that the ending markers won't be fixed.

I then created a modified Penguin controller that will work with Super IOB v1.5. The controller is designed to read disks with ending markers (both address and data) of $DA AA, normal address prologs on the even tracks and address prologs of $D4 AA 96 on the odd tracks. The controller also takes advantage of the fast reading and writing of Super IOB v1.5.

After copying the disk with this new controller, I thought I was finished. I was wrong. As it turns out, the hello program of the copy disk must be changed. So, I derived an alternate hello program and tried again. Everything went well until the program prompted me to flip over the disk and press a key to start the adventure. When I did so, I promptly received a message telling me that there was something wrong. So, I booted my DOS 3.3 system master and attempted to catalog the back side of the Coveted Mirror disk. Well, there were no files, which lead me to believe that there was not supposed to be DOS on this disk. I then remembered that the controller started at track 2. So, I changed it to start at track 0 and copied the backside again. Finally, it worked like a charm.

Step by Step

1) Format both sides of a blank disk with the following:

```
INIT SETUPA
```

2) Install the controller at the end of this article into Super IOB and use it to copy the front side of your Coveted Mirror disk. Do not use Super IOB's format option. If Super IOB comes up with a "Drive Error," try putting a "POKE 47426,24" at the end of line 1010.

3) Change the "TK = 2" in line 1010 of the controller to a "TK = 0" and use the controller to copy the back side of your Coveted Mirror disk.

4) Clear memory, type in this short BASIC program and save it on the FRONT side of the copied Coveted Mirror disk.

```
FP
10 PRINT CHR$(4) "MAXFILES1" :
   PRINT CHR$(4) "BRUN SETUP"
SAVE SETUPA
```

5) Now you can sit back and relax knowing that your original disk can be locked up in a safe somewhere while you use your copy.

p.s.

In examining the unprotected copy of my Coveted Mirror with Copy][Plus' "View Files" option, I found a list of the verbs used in Coveted Mirror in the file "CM.MESSAGES." This may help you in solving this game.

controller

```
1000 REM PENGUIN SOFTWARE
1010 TK = 2 :LT = 35 :ST = 15 :LS = 15 :CD = WR
     :FAST = 1
1015 T1 = TK : GOSUB 490
1020 LT = TK + 1 : GOSUB 1060 : GOSUB 610
1025 TK = TK + 1 : IF PEEK (BUF ) < MB THEN 1020
1027 GOSUB 230 :LT = 35 :TK = T1
1030 GOSUB 490 : GOSUB 610 : IF PEEK (TRK ) =
     LT THEN 1050
1040 TK = PEEK (TRK ) :ST = PEEK (SCT ) : GOTO
     1015
1050 HOME : PRINT "COPYDONE" : END
1060 RESTORE : GOSUB 170
1070 POKE 47445 ,212 + (TK / 2 = INT (TK / 2 ) )
     : RETURN
5000 DATA 218 ,170 ,218 ,170
```

controller checksums

```
1000  - $356B     1030  - $94B3
1010  - $2444     1040  - $14D3
1015  - $861D     1050  - $2012
1020  - $E161     1060  - $D772
1025  - $DFBD     1070  - $BA6F
1027  - $9946     5000  - $3C8B
```

softkey for...

Crimson Crown

by Tony Phalen

Penguin Software
830 Fourth Ave.
PO Box 311
Geneva, IL 60134
$29.95

Requirements:
64K Apple
COPYA
A sector editor
Crimson Crown original

Crimson Crown is a fantasy adventure based on Transylvania. It uses the same pictures (unfortunately) as Transylvania, with a few new ones here and there. It also has a new feature called Comprehend, which gives the program the ability to understand full and multiple sentence commands. With this is a vocabulary of over 1000 words! With a combination like this, it may be more fun testing the Comprehend feature than playing the game!

The Protection

Crimson Crown is ProDOS based, so upon bootup the normal ProDOS copyright notice is displayed. (The copyright notice was edited by Penguin to display their own copyright notices.) On normal ProDOS disks, a loader searches the directory for the first file ending with ".SYSTEM". This is usually the file called "BASIC.SYSTEM", which loads the ProDOS BASIC Interpreter. Penguin Bypasses this and instead loads the program directly through ProDOS. It loads the title page, displays it, and then jumps to the main part.

Trying to reset out of it didn't work, so I booted Bag Of Tricks and used Trax to check the marks. They were all normal! I then booted COPYA and ran it (with no modifications) and it copied with no problems. How easy! But I should have known. Just after bootup the program does a nibble count, and then hangs. Well, it shouldn't be to hard to find it since it has normal marks, right? Wrong! Penguin EORs the nibble count routine bytes with $FF, and even though we can find it in memory, it will be very hard to find it on the disk since the code will look different.

Well, after about 2 hours of trying to find it (and using the the back issues of COMPUTIST that I have), I finally found the count, which was loaded in at $3F00. Yeah, but now what? I may have found it in memory, but on the disk...? I wrote down the first few bytes of memory starting at $3F00, which were $A2 03 B5 00 48, then booted my sector editor and searched for

them. To my surprise, I found them on track $2, sector $9! So I scanned through memory and looked for my nibble count. No luck.

After a few more hours of work, I finally deciphered the code, and put a jump to the location the the program would jump to had the count been correct. Scratch one nibble count! I booted the disk and it worked like a charm, right up to the point where the game put me in some sort of cage with a steel grate over my head. No matter how hard I tried, I couldn't get out of it! Crud, another nibble count! Again, another few hours of continual searching went by. Just after the first move, in which you are at atree and can only go east, it does another count, and decides whether to put you in the pit, or in a subterranean cave (the correct place). Again, the count is hidden and very hard to find, but hard work pays off. The nibble count was found at $5980.

After looking at the routine, it seemed really easy to load a $56 into the accumulator and return from the subroutine to fool the program. Using the same technique I used to find the first routine, I found it at track 5, sector $A. I replaced the first three bytes ($4A 6D A7 in their encoded form) with the bytes $46 B9 8F (EORed form of $A9 56 60, or LDA #$56, RTS). Now that I was done, I thought I would boot my new version and have some 'real' fun. But Noooooo... Penguin has to do a checksum, also! It adds up the values and compares them to a certain value. If the code is changed, then you get to kick back in the pit. What we need to do here is balance out the checksum. How do you do this, you ask? Very carefully!

The original three bytes plus the fourth (we'll use it for adjustment) add up to $A8 and the carry set ($4A + 6D + A7 + 4A = A8, carry=1). For everything to match, our code plus the next byte must also add up to $A8, carry set. Ours equals $8E, carry set ($46 + B9 + 8F = 8E, carry=1). Subtract our total ($8E) from the original total ($A8) and you get $1A, the new fourth byte. Easy, right? We now neeed to modify our code (which was first changed to $46 B9 8F) with $46 B9 8F 1A on track 5 sector $A and we should be done. No more pit! copy complete!

Cookbook Method

1) Boot your DOS 3.3 System Master and run COPYA, copying both sides of the Crimson Crown disk.

2) Break out the sector editor and make the following changes (all on side 1):

```
Track $2, Sector $9, Byte $19
  From: $A0 89 A9
  To:   $4C A3 3F

Track $5, Sector $A, Byte $80
  From: $4A 6D A7 4A
  To:   $46 B9 8F 1A
```

3) Write the sector back out to the disk and your copy of Crimson Crown is complete!

softkey for...

Compu-

by The Nipper

Artworx
150 North Main Street
Fairport, NY 14450

Requirements:
48K Apple][
Compubridge
COPYA
A sector editor
A blank disk

A while ago I picked up a copy of Compubridge from a local software company that was going out of business. When I got it home and booted it up on my Apple //e it ran fine until I tried to read one of the files. Although all the letters were there it was almost unreadable because of the strange characters present. Instead of a space between words there was a "/" character and the numbers 0 to 9 had been replaced by the lower case letters p to y. My immediate thought was that the binary files used to store the text had been designed to be used with an Apple][or][Plus with only uppercase character sets.

To confirm this I took it to a friend's][Plus and sure enough it worked just flawlessly. So I set out to patch it so it would work with my //e. The first step was to deprotect it.

The Protection

The protection on Compubridge is really minimal. It consists of altered address epilogues from the normal DE AA to DF AA. First patch DOS to ignore read errors and use COPYA to copy it.

```
RUN COPYA
<CTRL-C>
70
CALL -151
B942:18
3D0G
RUN
```

Now make a few sector edits to make Compubridge DOS accept the standard epilogues.

Track	Sector	Byte	From	To
$0	$03	$91	$DF	$DE

You now have an open copy of Compubridge, but it still will not work on a //e or //c.

The Patch

The problem with the binary text files is that they were produced assuming only a uppercase character set. This means that it uses the ASCII code $F1 for a "1" instead of the standard code $B1. unfortunately an Apple //e or //c sees this code as a "q". What we need to do is to check each file character before it is written to the screen and if it is larger than $DA (a capital Z) we need to turn the 6 bit off. For example, to have a "1" appear on the screen as it should:

character	hex	binary
q	F1	11110001
1	B1	10110001

bridge

As can be seen the only difference between the binary number for a "q" and a "1" is the six bit is turned off for the "1". We can accomplish this by EORing the the $F1 with $40 (binary 01000000). However we must leave the characters with vales of $DA or below alone. After a bit of searching I found the that the binary program SW (Screen Write?) is responsible for printing the binary text files to the screen. SW resides in about two and a half pages of memory starting at $9300 and the access to the built-in print to screen routine (JSR FDED) that is responsible for printing the binary files is located at $93E1. Since the SW file has unused memory at the end of it, we have a convenient place to put our patch at $94AC. The first thing we need to do is to redirect SW to our patch instead of printing to the screen. This is done by making the following change:

```
93E1: 20 AC 94   (JSR 94AC)
```

This calls our simple patch which is as follows:

```
94AC:  STA $9AFF     - save character
94AF:  SEC           - set carry
94B0:  SBC #$DB      - subtract DB
94B2:  BMI $94BB     - branch if negative
94B4:  LDA $9AFF     - get character back
94B7:  EOR #$40      - turn off 6 bit
94B9:  BCS $94BE     - (always taken)
94BB:  LDA $94FF     - get character
94BE:  CLC           - clear carry
94BF:  JSR $FDED     - print character
94C2:  RTS           - return to program
```

The patch checks to see if a character is bigger than $DB. If it is, the character is EORes it with $40, to turn the six bit off, and then the character is printed. If the character is smaller than $DB it is just printed.

Step By Step

1) Deprotect it with COPYA as follows:

RUN COPYA
[C]
70
CALL-151
B942:18
3D0G
RUN

2) Now make a few sector edits to make Compubridge DOS accept the standard epilogues.

Track	Sector	Byte	From	To
$0	$03	$91	$DF	$DE

3) Now load the SW program.

BLOAD SW

4) Enter the monitor and change the screen print JSR to point to the patch.

CALL -151
93E1:20 AC 94

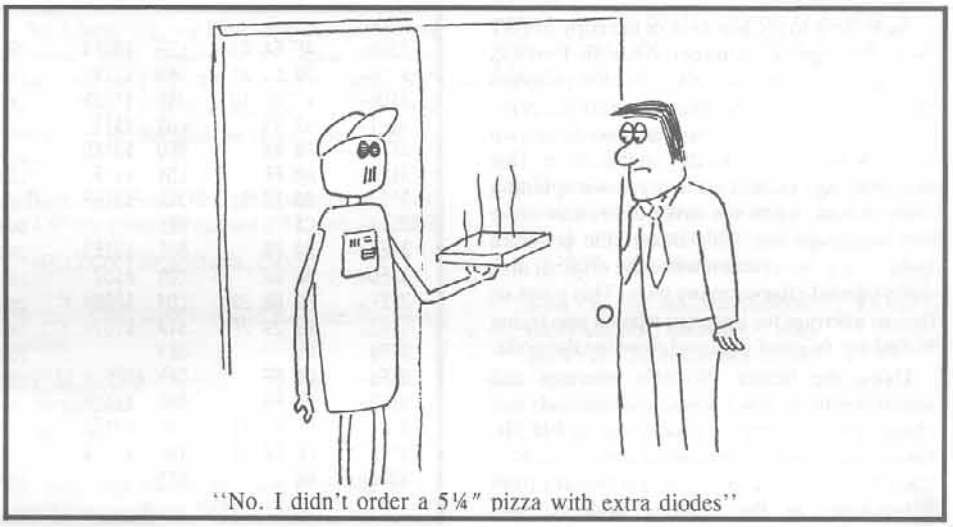
5) Type in the following hex dump to install the patch.

94AC:8D FF 94 38 E9 DB 30 07
94B4:AD FF 94 49 40 B0 03 AD
94BC:FF 94 18 20 ED FD 60

6) Now resave the SW file with the patch.

BSAVE SW,A$9300,L$2FF

That's it. Enjoy!



"No. I didn't order a 5¼" pizza with extra diodes"

Fleet System 3

by Clay Harrell

Requirements:
Apple //e or //c
COPYA
ProDOS System Master
Sector editor
Three blank disks

Fleet System Three (FST) is an excellent ProDOS-based integrated word processing / thesaurus / spell checker. The thesaurus and spell checker disks backup normally with COPYA, but the boot disk is copy protected.

The Easy Part
Finding the Protection Code In Memory

The copy protection is based around track three, which is not in normal DOS format. This was easy to diagnose: I just tried copying the disk with the Senior PROM's copy function. Errors occurred when copying track three. So I copied tracks $00-02, and then selected the copy option again with a start track of four. The rest of the tracks ($04-22) copied normally.

Using the Senior PROM's sector editor to nibble read track three, the hi-res screen of the track showed the track was empty except for data and address headers. Since there was no data on the track, it had to be used just as a protection check to verify the presence of the original FST disk.

Next thing to try was to boot the copy of FST (with the empty track three). After the ProDOS message appeared on the screen, the program hung as it tried to read track three. If the drive door was opened, it would turn on drive two and look for the original disk in that drive. This is interesting: I could leave drive one spinning for a minute, open the drive door, and drive two would go on. This means the program hadn't gone out to lunch when the original disk wasn't found after so many tries. This gives us time to interrupt the program when it was trying to find the original disk and examine the code.

Using the Senior PROM's interrupt and restart functions, it was easy to see the code that checked for the original disk started at $1E5B. I then interrupted the disk while it was checking track three, put a RTS instruction (Return from Subroutine) at the running address, and restarted the program. The program continued to load as though the original disk was found.

Since I had found the protection code, I used the Senior PROM's sector editor to search the disk for this code. When found, we could put a "60" (which is a Return from Subroutine instruction) at the start of the code to defeat it. Unfortunately, nothing was found! This means the protection code was somehow encoded on the disk, so it could not be easily found and defeated.

The Hard Part
Finding the Protection Code On the Disk

The code at $1E5B could be encoded a thousand ways on the FST disk. Trying to find it on the disk could prove to be very challenging (FST hopes it would take more time and effort to find than anyone would be willing to invest).

The first step I took was to examine how the FST disk boots. Since the disk uses ProDOS, it follows certain boot rules. These are outlined below.

After track 0, sector 0 is read into $800-8FF by the disk controller card, this code reads track 0, sector 1 into $900-9FF (using the disk controller card ROM code as a subroutine). Then the combined code at $800-9FF searches the ProDOS directory on track 0 for a file named "PRODOS". This file is read from disk into $2000 and executed. The "PRODOS" code is relocated to upper RAM and searches the directory for the first file name ending with ".SYSTEM". The system file is read into $2000 and executed.

The first thing not to do is to suspect the file "PRODOS". Software publishers will generally stay away from modifying this file as one of ProDOS's advertised features is that you can update any of your current ProDOS disks to the newest version by just copying the new file "PRODOS" to your current disk. The file to suspect is the first ".SYSTEM" file in the ProDOS directory, as it gets loaded immediately after the operating system is loaded. So, I booted my ProDOS system disk and did a "CAT" of the FST disk. The first ".SYSTEM" file was named FLEET.SYSTEM.

To examine the FLEET.SYSTEM program I booted my ProDOS system disk and bloaded the file with "BLOAD FLEET.SYSTEM, A$2000, TSYS." The "TSYS" is necessary since the file is not a BIN type file (so to speak). I then entered the Monitor and began examining the code of this program. The first instruction at location $2000 was a Jump to $31D8. This code is listed in the box below.

This is a nifty little subroutine that moves $2000-31D7 down to $1D29-2F00. Notice the first instruction changes the reset vector so a reset will reboot the disk drive. Also, after using the subroutine at $31EF to move $11 whole pages of memory ($2000-30F0 to $1D29-2E28), the code "falls through" at location $31ED and moves the last $D8 bytes to complete the move. When done, location $31DE jumps to location $1D2F.

The code at location $1D2F starts out with the instruction LSR $1D32. Following this instruction appears to be garbage; but not really. The instruction at $1D2F gets executed which changes the code at $1D32 to a ROR $1D38.

```
31D8-   4E F4 03    LSR  $03F4      ;Reboot on reset.
31DB-   20 E1 31    JSR  $31E1      ;jump to subroutine to move memory.
31DE-   4C 2F 1D    JMP  $1D2F      ;exit this routine.
31E1-   A2 11       LDX  #$11       ;index counter for # of pages to move.
31E3-   F0 08       BEQ  $31ED      ;is the counter 0?  if so goto $31ED.
31E5-   A0 FF       LDY  #$FF       ;byte counter for move.
31E7-   20 EF 31    JSR  $31EF      ;goto subroutine to move memory.
31EA-   CA          DEX             ;decrement index counter for # of pages.
31EB-   D0 F8       BNE  $31E5      ;done moving 11 pages?  if not goto $31E5
31ED-   A0 D7       LDY  #$D7       ;done 11 pages, move $D8 more bytes.
31EF-   B9 00 20    LDA  $2000,Y    ;subroutine. load a byte indexed by Y-reg.
31F2-   99 29 2F    STA  $1D29,Y    ;store a byte indexed by Y-reg.
31F5-   88          DEY             ;decrement Y-reg.
31F6-   C0 FF       CPY #$FF        ;compare Y-reg (have we moved 256 bytes?).
31F8-   D0 F5       BNE  $31EF      ;if not, go back to $31EF to move more.
31FA-   EE F1 31    INC  $31F1      ;increment loading page number.
31FD-   EE F4 31    INC  $31F4      ;increment storing page number.
3200-   60          RTS             ;return to calling location.
```

Then this instruction get executed which changes the code at $1D38... As you can see the code is modifying itself! The trick is to execute this code one instruction at a time at another location so we can examine the real code.

To start off, I typed "5000:4E 32 1D 60", followed by "5000G". This executed the LSR $1D32 instruction I had typed in at location $5000. This uncovered the ROR $1D38 instruction. So I typed "5000:6E 38 1D 60", followed by "5000G". This executed the ROR $1D38 instruction I had entered. Finally I typed "5000:6E 3A 1D 60", followed by "5000G". Now we could examine the code at $1D2F as it would be executed. The result follows:

```
1D2F-   4E 32 1D    LSR  $1D32
1D32-   6E 38 1D    ROR  $1D38
1D35-   6E 3A 1D    ROR  $1D3A
1D38-   20 A8 1F    JSR  $1FA8
1D3B-   A0 57       LDY  #$57
1D3D-   98          TYA
1D3E-   99 A8 1F    STA  $1FA8,Y
1E41-   88          DEY
1D42-   10 F9       BPL  $1D3D
```

As you can see, at location $1D38 a subroutine starting at $1FA8 is executed, and then at location $1D3B the subroutine at $1FA8 is wiped out! Obviously this has to be an important subroutine if it is destroyed the moment after it is executed.

It turns out that the subroutine at $1FA8 is the code that deciphers the whole file: hence this is why we could not find the code on the disk. I'm not going to list the subroutine at $1FA8 because it is very obscure and hard to follow, but it is a very good encoding scheme. So good in fact, that I could not find an easy why to change one byte in the code (to defeat the disk check routine at $1E5B), and have the rest of the routine decode properly. Instead of spending vast amounts of time for something I'm not sure would ever work, I took another routine. Why not have the program un-encode itself, modify it (defeat the track three disk check), defeat the encoding routine, and then save it. That's the easy way, and the best way as the code will be saved on the disk in an un-encoded format so next time you need to change it, there's no problems.

Step by Step

1) First, we need to copy the boot FST disk to a blank disk, ignoring track three. To do this, use Locksmith Fast Copy (which will copy the FST disk without bombing when track three is encountered) to copy the disk. If you don't have this program, boot your DOS 3.3 System Master and type:

```
RUN COPYA
[C]
CALL-151
3A1:18
3DOG
70
RUN
```

Copy the original FST disk normally. Note: Ignore the grinding noises on track three.

2) Boot your ProDOS system master and bload the FLEET.SYSTEM file from the FST copy at $2000.

```
BLOAD FLEET.SYSTEM,A$2000,TSYS
```

3) Enter the Monitor and execute the move routine.

```
CALL-151
31DE:60
31DBG
```

4) Enter the following code at $5000 to un-encode the code at $1D2F.

```
5000:4E 32 1D 6E 38 1D 6E 3A
5008:1D 60
5000G
1FA8G
```

5) Now we can defeat the track three disk check routine.

```
1E5B:60
```

6) Reset the move routine at $31D8 and after executing the move routine, have the program jump to location $1D3B in stead of $1D2F. This will defeat the un-encoding routine at $1D2F-1D37 and the routine at $1FA8.

```
31EF:B9 00 20 99 29 1D
31DE:4C 3B 1D
```

7) Now move the whole program up to $5000, and then back to $2000. Note we must still use the move routine at $31D8 because all .SYSTEM files are loaded at $2000. We have to first move the program up to $5000, and then back to $2000. Otherwise if we tried to move the program from $1D29-2F00 to $2000-31D7 it would overwrite itself.

```
5000<1D29.2F01M
2000<5000.61D7M
```

8) Finally, bsave the modified FLEET.SYSTEM to the copied FST disk.

```
BSAVE FLEET.SYSTEM, A$2000,L$1201,TSYS
```

Are We Done Yet?

As it turns out, we have only completed one of three steps necessary to defeat the copy protection on FST. There are two other SYS files (OP and FS) which must have the same procedure performed on them. Here are the steps:

9) Boot your ProDOS system master and bload the OP file from the copied FST disk at $2000.

```
BLOAD OP,A$2000,TSYS
```

10) Enter the Monitor and execute the move routine.

```
CALL-151
3CDE:60
3CDBG
```

11) Enter the following code at $1000 to un-encode the code at $1D2F.

```
1000:4E 32 1D 6E 38 1D 6E 3A
1008:1D 60
1000G
1FA8G
```

12) Now we can defeat the track three disk check routine.

```
1E5B:60
```

13) Reset the move routine at $3CD8 and after executing the move routine, have the program jump to location $1D3B in stead of $1D2F. This will defeat the un-encoding routine at $1D2F-1D37 and the routine at $1FA8.

```
3CEF:B9 00 20 99 29 1D
3CDE:4C 3B 1D
```

14) Now move the whole program up to $5000, and then back to $2000.

```
5000<1D29.3A01M
2000<5000.6CD7M
```

15) Finally, save the modified OP to the copied FST disk.

```
BSAVE OP, A$2000, L$1D01, TSYS
```

16) Boot your ProDOS system master and bload the FS file from the copied FST disk at $2000.

```
BLOAD FS,A$2000, TSYS
```

17) Enter the Monitor and execute the move routine.

```
CALL-151
8ADE:60
8ADBG
```

18) Enter the following code at $1000 to un-encode the code at $1D2F.

```
1000:4E 32 1D 6E 38 1D 6E 3A
1008:1D 60
1000G
1FA8G
```

19) Now we can defeat the track three disk check routine.

```
1E5B:60
```

20) Reset the move routine at $8AD8 and after executing the move routine, have the program jump to location $1D3B in stead of $1D2F. This will defeat the un-encoding routine at $1D2F-1D37 and the routine at $1FA8.

```
8AEF:B9 00 20 99 29 1D
8ADE:4C 3B 1D
```

21) Since this program is larger than the other two, we have to save the program to the copied FST disk at its present location, and then reload it at $2000, and then resave it.

```
BSAVE FS, A$1D29,L$6AD7, TSYS
BLOAD FS, A$2000, TSYS
BSAVE FS, A$2000,L$6B01, TSYS
```

22) Copy the Thesaurus and spell checker disks to blank disks using any normal copy program.

And Fleet System Three is now deprotected!

The Book Of Softkeys

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 since past, but the information contained in these ancient documents has been diligently transcribed to the pages of a modern reference work: The Book Of Softkeys.

From deep within the COMPUTIST archives comes a collection of softkeys originally contained in issues 1 through 15. These volumes also contain some of the more useful programs and tutorials presented in those early issues. The books make an economical alternative to those rare (and unavailable) back issues of Hardcore COMPUTIST.

Volume I: Issues 1-5 ($7.95)

contains softkeys for: Akalabeth | Ampermagic | Apple Galaxian | Aztec | Bag of Tricks | Bill Budge's Trilogy | Buzzard Bait | Cannonball Blitz | Casino | Data Reporter | Deadline | Disk Organizer II | Egbert II Communications 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 program listings of need-to-have programs used to make unprotected backups.

Volume II: Issues 6-10 ($12.95)

contains softkeys for: Apple Cider Spider | Apple Logo | Arcade Machine | The Artist | Bank Street Writer | Cannonball Blitz | Canyon Climber | Caverns of Freitag | Crush, Crumble & Chomp | Data Factory 5.0 | 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 4.0, 4.0c, 4.1c | the Spy Strikes Back | Time Zone v1.1 | Visible Computer: 6502 | Visidex | Visiterm | Zaxxon | Hayden Software | Sierra Online Software | **PLUS** the complete listing of the ultimate cracking program... Super IOB 1.5 | **and more!**

Volume III: Issues 11-15 ($17.95)

contains softkeys for: Alien Addition | Alien Munchies | Alligator Mix | Computer Preparation SAT | Cut And Paste | Demolition Division | DLM (Developement Learning Materials) software | EA (Electronic Arts) software | Einstein Compiler version 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 v1.7 | MatheMagic | Minus Mission | Millionaire | Music Construction Set | One On One | PFS software | PS (Penguin) Software | The Quest | Rocky's Boots | Sabotage | Seadragon | Sensible Speller IV | 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 _____ Exp. _____

Signature _____ CP44

- ☐ Volume I $7.95 + $2 shipping & handling
- ☐ Volume II $12.95 + $2 shipping & handling
- ☐ Volume III +17.95 + $2 shipping & handling
- ☐ All three volumes! $38.85 + $4 shipping & handling

Foreign orders (except Canada & Mexico), please add $5.00 per book shipping & handling. U.S. funds drawn on U.S. banks only. Most orders shipped within 5 working days, however please allow 4-6 weeks delivery for some orders. Washington State orders add 7.8% sales tax.
Send your orders to:

COMPUTIST
PO Box 110846-T
Tacoma, WA 98411

the COMPUTIST shopper...

Books

Ashton-Tate
- Intro to DBASE II $20.00

Educational Systems
- Microref for Wordstar $12.00
- Microref for Supercalc $12.00
- Microref for Visicalc $12.00

Microsoft
- The Printed Word $16.50
- Appleworks $15.75

Quality Software
- **Beneath Apple ProDos $16.00**

Software

Academy Software, Inc
- Typing Tutor + Word Invaders (//e) $16.00

Accolade
- Dambusters $24.50
- Hardball $24.50
- Fight Night $24.50
- Law of the West $24.50

Addison-Wesley
- Smart Eyes $48.00
- The Hobbit $28.00
- The Fellowship of the Ring $32.00

Activision
- Greeting Card Maker $32.00
- Term Paper Writer $48.00
- Writer's Choice $39.75
- Filer's Choice $39.75
- Planner's Choice $39.75
- Hacker $32.00
- Ghostbusters $32.00
- Tass Times in Tonetown $32.00
- Space Shuttle: Journey into Space $32.00
- The Rocky Horror Show $27.75
- Shanghai $32.00
- Little People Discovery Kit $32.00
- Gamemaker $39.75
- Labyrinth $32.00
- Borrowed Time $32.00
- Portal $36.00
- Alter Ego (male) $39.75
- Alter Ego (female) $39.75
- Mindshadow $32.00
- Murder on the Mississippi $32.00
- Hacker II $32.00

Advanced Ideas
- The Game Show $32.00
- Tic Tac Show $32.00
- Wizard of Words $32.00

American Education Computer
- Spanish $16.00
- Science, Grades 3-4 ..... $16.00
- Spelling $39.75
- Phonics $39.75
- Learn to Read $39.75
- Reading Comprehension $39.75
- U.S. History $16.00
- U.S. Geography $16.00

Artsci
- **Magic Window //e $106.00**
- **Magic Window II $106.00**
- Magic Office System $176.00
- Magic Window II extra (128K) $141.50

Ashton-Tate
- DBase II (requires Z80) $415.00

Avalon Hill
- Statis Pro Baseball $28.00
- Mission on Thunderhead $20.00
- Dreadnoughts $24.00
- Under Fire $47.50

Bantam Electronics Publishing
- Card & Party Shop $32.00
- Comic Strip Maker $32.00

Baudville
- Award Maker Plus $28.50
- Take 1 Delux $42.00
- Blazing Paddles $24.25
- Prince $35.25
- Ted Bear's Rainy Day Games $21.50
- Video Vegas $21.50

Beagle Bros
- Apple Mechanic $21.00
- Beagle Basic Utility $25.75
- Double Take $25.75
- Utility City $21.00
- DOS Boss Utility $17.25
- Silicon Salad $17.50
- DiskQuick $21.00
- Triple Dump $28.00
- Fat Cat $24.75
- D Code $28.50
- Power Print $28.50
- Extra K $28.50
- Pro-Byter $24.75
- Big U $24.75
- Macroworks $24.75
- Super Macroworks $35.25
- **Global Program Line Editor $35.25**
- Beagle Graphics $42.75
- Frame-UP $21.00
- Shape Mechanic $28.50
- Font Mechanic $21.50
- Minipix Disk 1 ..... $21.50
- Minipix Disk 2 ..... $21.50
- Minipix Disk 3 ..... $21.50
- Alpha Plot $25.50
- Beagle Screens $25.50
- Beagle Compiler $53.25
- Beagle Bag $21.00

BPI Systems
- General Accounting $279.00
- Payroll $279.00
- Accounts Receivable $279.00
- Accounts Payable $279.00
- General Accounting (ProDOS) $300.00
- Payroll (ProDOS) $300.00
- Accounts Receivable (ProDOS) $300.00
- Accounts Payable (ProDOS) $300.00
- Inventory Control (ProDOS) $300.00

Quality Software
- **Bag of Tricks II $39.75**

Broderbund Software
- Science Tool Kit Master Module $55.50
- Science Tool Kit Module 1: Speed and Motion $32.00
- Science Tool Kit Module 2: Earthquake Lab $32.00
- Where in the USA is Carmen San Diego? $36.00
- Where in the WORLD is Carmen San Diego? $32.00
- Type $36.00
- On Balance $79.50
- Bank Street Writer 64K $55.50
- Bank Street Mailer 64K $55.50
- Bank Street Mailer 128K $55.50
- Bank Street Filer 64K $55.50
- Bank Street Filer 128K $55.50
- Bank Street Speller $55.50
- Dazzle Draw $47.50
- The Print Shop $39.75
- Print Shop Companion $32.00
- Print Shop Graphics Library Disk 1 ..... $20.00
- Print Shop Graphics Library Disk 2 ..... $20.00
- Print Shop Graphics Library Disk 3 ..... $20.00
- Animate $55.50
- The Toy Shop $47.50
- Airheart $28.00
- Lode Runner $28.00
- Championship Lode Runner $28.00
- Breakers $36.00
- Karateka $28.00
- Captain Goodnight and the Islands of Fear $28.00

DLM Inc.
- Verb Viper $27.00
- Spelling Wiz $27.00
- Number Farm $24.00
- Create With Garfield $24.00
- Create w/Garfield - Delux $32.00
- Teddy Bear-rels of Fun $32.00
- Fish Scales $24.00
- Meteor Multiplication $27.00
- Alligator Mix $27.00
- Decimal Discovery $36.75
- Writing Adventure $37.00

EPYX
- Temple of Apshai Trilogy $32.00
- Championship Wrestling $32.00
- Destroyer $32.00
- The Movie Monster Game $32.00
- World Games $32.00
- Summer Games $32.00
- Summer Games II $32.00
- Winter Games $32.00
- World's Greatest Baseball Game $32.00
- World's Greatest Football Game $32.00

Infocom
- Zork I $32.00
- Zork II $36.00
- Zork III $36.00
- Deadline $39.75
- Wishbringer $32.00
- Suspended $39.75
- Planetfall $32.00
- Seastalker $32.00
- Cutthroats $32.00
- Suspect $36.00
- A Mind Forever Voyaging $36.00
- Enchanter $32.00
- Sorcerer $36.00
- Spellbreaker $39.75
- Trinity $32.00
- Fooblitzky $32.00
- Ballyhoo $32.00
- Hitchhiker's Guide $32.00
- Moonmist $32.00
- Starcross $39.75

Micropro International
- Wordstar Professional $349.50
- Wordstar $247.00

Microprose Software
- Decision in the Desert $32.00
- Crusade in Europe $32.00
- Conflict in Vietnam $32.00
- Gunship $32.00
- F-15 Strike Eagle $28.00
- Solo Flight $28.00
- Silent Service $28.00

Hardware

AAA International
- 5 ¼ inch Wooden Diskcase (Holds 70 5 ¼ inch disks) $20.25
- 3 ½ inch Wooden Diskcase (Holds 30 3 ½ inch disks) $18.00

Abati
- Letter-Quality Parallel Printer (LQ-20P) 132 COL, 18 CPS, with tractor $298.50

Acrylic Arts, Inc.
- Universal printer stand $17.00

Amdek
- Amdisk I - 3" disk drive $274.50
- Video 300A Amber monitor $183.00
- Tilt/Swivel for color 300-730 ..... $30.00

AST Research-Apple Division
- Multi I/O (with clock, calendar, 2 serial ports) $157.50
- Multi I/O (with clock & calendar) $130.50

C H Products
- Mach III w/Fire Button (//e) $44.00
- Mach II Joystick (//e) $36.50
- Mach III w/Fire Button (][,][+) $44.00
- Mach II Joystick (][,][+) $36.50

Compu Cable
- Apple][6 ft parallel cable $30.00

Hayes Microcomputer Products
- Smartmodem 300 w/Smartcom I (//c) $222.00
- Micromodem //e w/Smartcom I $186.00

Kraft Systems
- Apple Joystick (3 buttons) $42.75

Panasonic
- Dot Matrix parallel printer (KX-P1092) 80 COL, 180 CPS $454.50
- Dot Matrix parallel printer (KX-P1081) 80 COL, 120 CPS $298.50
- Dot Matrix parallel printer (KX-P1091) 80 COL, 160 CPS $389.25
- Dot Matrix parallel printer (KX-P1592) 132 COL, 180/38 CPS $627.00
- Dot Matrix parallel/serial printer (KX-P1595) 132 COL, 240/51 CPS, 15K buffer $816.00

Video-7
- Color Enhancer //e w/Dazzle Draw $162.00
- Color Enhancer //c w/Dazzle Draw $117.00

How To Order

- Circle your selection. If total order is less than $200, please add $2.00/item shipping & handling. (Most orders shipped UPS so please use street address.)
- Foreign orders please inquire as to appropriate shipping fees.
- Washington State residents please add 7.8% sales tax.
- Domestic orders over $200 FREE shipping.
- Offer good while supplies last. All products are for the Apple][unless otherwise specified

SoftKey Publishing PO Box 110816-T Tacoma, WA 98411

Name _____ ID# _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone _____

VISA MasterCard _____-_____-_____-_____ Exp._____

Signature _____ CP44

Are you missing a piece of the picture ???!!!

Issue	Mag $4.75	Disk $9.95	Both $12.95
43	☐	☐	☐
42	☐	☐	☐
41	☐	☐	☐
40	☐	☐	☐
39	☐	☐	☐
38	☐	☐	☐
37	☐	☐	☐
36	☐	☐	☐
35	☐	☐	☐
34	☐	☐	☐
33	☐	☐	☐
32	☐	☐	☐
31	☐	☐	☐
30	☐	☐	☐
29	☐	☐	☐
28	☐	☐	☐
27	☐	☐	☐
26	☐	☐	☐
25	☐	☐	☐
24	☐	☐	☐
23	•	☐	•
22	☐	☐	☐
21	•	☐	•
20	☐	☐	☐
19	☐	☐	☐
18	•	☐	•
17	•	☐	•
16	☐	☐	☐
Book of Softkeys Vol 3			
15	•	☐	•
14	•	☐	•
13	•	☐	•
12	•	☐	•
11	•	☐	•
Book of Softkeys Vol 2			
10	•	☐	•
9	•	☐	•
8	•	☐	•
☆ 7	☐	☐	☐
6	•	☐	•
Book of Softkeys Vol 1			
4	•	☐	•
3	•	☐	•
Core 2	☐	☐	☐
2	•	☐	•
1	☐	☐	☐
Core 1	☐	☐	☐
Core 3	☐	☐	☐
Computing 3	☐	•	•
Best of Hardcore Computing	•	☐	•
Core Special $10.00 (All three CORE magazines)		☐	

Some disks apply to more than one issue and are shown as taller boxes. Special "Both" disk & magazine combination orders apply to one issue and its corresponding disk.

Book Of Softkey titles are inserted for reader information only. Each book contains all of the softkeys appearing in the issues below the title line. Please consult current Book of Softkey ad for current prices and ordering information.

☆ We have a limited supply of these issues.

• Back issue is no longer available

COMPUTIST back issues and library disks are frequently referenced in current issues.

Some back issues are no longer available, but library disks can still be purchased for all back issues.

What is a library disk?

A library disk is a diskette that contains programs that would normally have to be entered by the user. Documentation for each library disk can be found in the corresponding issue.

Rates For Foreign Orders

- Canada and Mexico rates are identical to U.S. First Class unless otherwise specified.
- Other Foreign Back Issue Rates: $12.25 each. (includes shipping)
- Other Foreign Library Disk rates: $11.94 each. (includes shipping). Special "Both" disk and magazine combinations shown do NOT apply to Foreign orders.

Send me the back issues and/or library disks indicated:

Name _____ ID# _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone _____

VISA MasterCard _____-_____-_____-_____ Exp._____

Signature _____ CP44

Send check or money order to: COMPUTIST PO Box 110846-T Tacoma, WA 98411.
Most orders are shipped within 5 working days, however please allow 4-6 weeks delivery for some orders. Most orders shipped UPS so please use street address. Offer good while supply lasts. In Washington state, add 7.8% sales tax. **U.S. funds drawn on U.S. bank only.**

Complete Your Collection!

CORE 3 Games: Constructing Your Own Joystick | Compiling Games | *GAME REVIEWS:* Over 30 of the latest and best | Pick Of The Pack: All-time TOP 20 games | Destructive Forces | EAMON | Graphics Magician and GraFORTH | Dragon Dungeon | .....

CORE 2 Utilites: Dynamic Menu | High Res: Scroll Demo | GOTO Label: Replace | Line Find | Quick Copy: Copy | .....

CORE 1 Graphics: Memory Map | Text Graphics: Marquee | Boxes | Jagged Scroller | Low Res: Color Character Chart | High Res: Screen Cruncher | The UFO Factory | Color | Vector Graphics:Shimmering Shapes | A Shape Table Mini-Editor | Block Graphics: Arcade Quality Graphics for BASIC Programmers | Animation | .....

Hardcore Computing 3 HyperDOS Creator | Menu Hello | Zyphyr Wars | Vector Graphics | Review of Bit Copiers | Boot Code Tracing | Softkey IOB | Interview with 'Mike' Markkula | .....

Back Issues

43 *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 *Softkeys* | Light Simulator | *Readers' Softkeys* | 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 *Softkeys* | The Periodic Table | Gemstone Warrior | Inferno | Frogger | *Readers' Softkeys* | 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 *Softkeys* | Adventure Writer | Mychess II | Raster Blaster | *Readers' Softkeys* | Cranston Manor | Ghostbusters | Designer's Pencil | E-Z Learner | The American Challenge | Crime Wave | Encyclopedia Britannica Programs | *Features* | Taking the Wiz out of Wizardry | Adding a Printer Card Driver to Newsroom | *Core* | The Games of 1986 | .....

39 *Softkeys* | MIDI/8 Plus | Homeword v2.1 | Borrowed Time | Amazon | Speed Reader][| *Readers' Softkeys* | 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 | *Hints* | Carmen Sandiego | .....

38 *Softkeys* | Cyclod | Alternate Realty | Boulder Dash I & II | Hard Hat Mack (Revisited) | The Other Side | *Readers' Softkeys* | 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 *Softkeys* | Under Fire | Pegasus][| Take 1 (revisited) | Flight Simulator II v1.05 (part 2) | *Readers' Softkeys* | 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 *Softkeys* | Flight Simulator II v 1.05 | AutoDuel | *Readers' Softkeys* | 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 | *Advanced Playing Techniques* | Championship Lode Runner | .....

35 *Softkeys* | Hi-res Cribbage | Olympic Decathlon | Revisiting F-15 Strike Eagle | Masquerade | The Hobbit | *Readers' Softkeys* | 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 installng a CPS Clock Driver | .....

34 *Softkeys* | Crisis Mountain | Terripin Logo | Apple Logo II | Fishies 1.0 | SpellWorks | Gumball | *Readers' Softkeys* | Rescue at Rigel | Crazey Mazey | Conan | Perry Mason: The Case of the Mandarin Murder | Koronis Rift | *Feature* | More ROM Running | *Core* | Infocom Revealed | .....

33 *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 | *Readers' Softkeys* | 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 *Softkeys* | Revisiting Music Construction Set | Cubit | Baudville Software | Hartley Software | Bridge | Early Games for Young Children | Tawala's Last Redoubt | *Readers' Softkeys* | 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 *Softkeys* | Trivia Fever | The Original Boston Computer Diet | Lifesaver | Synergistic Software | Blazing Paddles | Zardax | *Readers' Softkeys* | 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 *Softkeys* | Millionaire | SSI's RDOS | Fantavision | Spy vs. Spy | Dragonworld | *Readers' Softkeys* | 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 *Softkeys* | Threshold | Checkers v2.1 | Microtype | Gen. & Organic Chemistry Series | Uptown Trivia | Murder by the Dozen | *Readers' Softkeys* | 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 *Softkeys* | Ultima IV | Robot Odyssey | Rendezvous | Word Attack & Classmate | Three from Mindscape | Alphabetic Keyboarding | Hacker | Disk Director | Lode Runner | MIDI/4 | *Readers' Softkeys* | 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 *Softkeys* | Microzines 1-5 | Microzines 7-9 | Microzines (alternate method) | Phi Beta Filer | Sword of Kadash | *Readers' Softkeys* | 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 | *Readers' Softkeys* | Financial Cookbook | Super Zaxxon | Wizardry | Preschool Fun | Holy Grail | Inca | 128K Zaxxon | *Feature* | ProEdit | *Core* | Games of 1985 part 1 | .....

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 | *Readers' Softkeys* | 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 | *Readers' Softkeys* | 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* | Direct Sector Access From DOS | ..

22 *Softkeys* | Miner 2049er | Lode Runner | A2-PB1 Pinball | *Readers' Softkeys* | 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 | *Reader' Softkeys* | 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 *Readers' 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 *Softkey* | Sensible Speller for ProDOS | Sideways | *Readers' Softkeys* | 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 |

Looking for the

Best Deal in Town?

How about ALL of our Super IOB controllers, (through 1986) in ONE package!

This package contains:

► **TWO DISKS** (supplied in DOS 3.3). Each containing at least 60 Super IOB Controllers including the standard, swap, newswap and fast controllers. In addition, each disk has the Csaver program from COMPUTIST No. 13; version 1.5 of Super IOB; 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.**

Disk 1

Volume 1 of the Super IOB collection covers all the controllers appearing in COMPUTIST No. 9 through No. 26. In addition, the newswap and fast controllers from COMPUTIST No. 32 are included. 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 III, Word Challenge, Xyphus, Zaxxon

Disk 2

Volume 2 of the Super IOB collection covers all the controllers appearing in 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

To Order: Send $9.95 for each volume or $19.95 for a complete package that includes: both disks, a reprint of "Disk Inspection and the use of Super IOB" and COMPUTIST No. 32. U.S. funds drawn on U.S. banks. Foreign orders (other than Canada or Mexico) add 20% shipping. Washington state residents add 7.8% sales tax. Mail orders to: Super IOB Collection PO Box 110846-T Tacoma, WA 98411

*Requires at least 64K of memory.

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