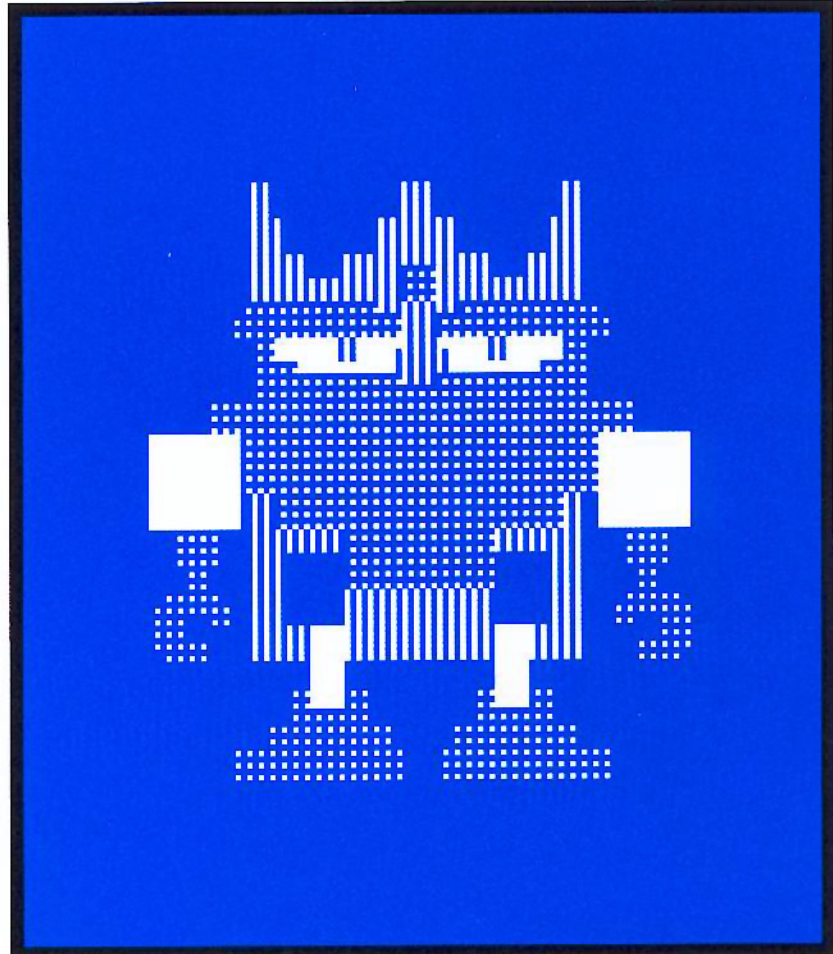**Softkeys For:**

Word Juggler
Sundog v2.0
Summer Games II
Thief
Instant Pascal
Epyx Software
Tink! Tonk!
Chipwits

Core:

The Mapping of
Ultima IV

Feature:

How to be the
Sound Master

(Page 12)

Many of the articles published in COMPUTIST detail the removal of copy protection schemes from commercial disks or contain information on copy protection and backup methods in general. We also print bit copy parameters, tips for adventure games, advanced playing techniques (APT's) for arcade game fanatics and any other information which may be of use to the serious Apple user.

COMPUTIST also contains a special CORE section which focuses on information not directly related to copy protection. Topics may include, but are not limited to: tutorials, hardware/software product reviews and application and utility programs.

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 from normal text by being indented and bold. An example is:

PR#6

Follow this with the RETURN key. The RETURN key must be pressed at the end of every such command unless otherwise specified.

Control characters are indicated by being boxed. An example is:

6 ☐ P

To complete this command, you must first type the number 6 and then place one finger on the CTRL key and one finger on the P key.

Requirements: Most of the programs and softkeys which appear in COMPUTIST require one of the Apple][series of computers and at least one disk drive with DOS 3.3. Occasionally, some programs and procedures have special requirements. The prerequisites for deprotection techniques or programs will always be listed at the beginning of the article under the "Requirements" heading.

Software Recommendations: The following programs (or similar ones) are strongly recommended for readers who wish to obtain the most benefit from our articles:

1) **Applesoft Program Editor** such as Global Program Line Editor (GPLE).
2) **Sector Editor** such as DiskEdit, ZAP from Bag of Tricks or Tricky Dick from The CIA.
3) **Disk Search Utility** such as The Inspector, The Tracer from The CIA or The CORE Disk Searcher.
4) **Assembler** such as the S-C Assembler or Merlin/Big Mac.
5) **Bit Copy Program** such as Copy][Plus, Locksmith or The Essential Data Duplicator.
6) **Text Editor** capable of producing normal sequential text files such as Applewriter][, Magic Window][or Screenwriter][.

You will also find COPYA, FID and MUFFIN from the DOS 3.3 System Master Disk useful.

Super IOB: This program has most recently appeared in COMPUTIST No. 22. Several softkey procedures will make use of a Super IOB controller, a small program that must be keyed into the middle of Super IOB. The controller changes Super IOB so that it can copy different disks. To get the latest version of this program, you may order COMPUTIST No. 22 as a back issue or order Program Library Disk No. 22.

RESET Into The Monitor: Some softkey procedures require that the user be able to enter the Apple's system monitor during the execution of a copy protected program. 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 of COMPUTIST No. 6 or the "Dual ROM's" article in COMPUTIST No. 19.

Apple //e - Apple //c: Install a modified CD ROM on the computer's motherboard. Clay Harrell's company (Cutting Edge Ent.: Box 43234 Ren Cen Station-HC, Detroit, MI 48243) sells a hardware device that will give you this ability. Making this modification to an Apple //e will void its warranty but the increased ability to remove copy protection may justify it.

Recommended Literature: The Apple]I 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, $19.95; *Assembly Language For The Applesoft Programmer*, Roy Meyers and C.W. Finley, Addison Wesley, $16.95; and *What's Where In The Apple*, William Lubert, Micro Ink, $24.95.

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. To understand this format, you must first understand the formatted LIST feature of Applesoft.

An illustration- If you strike these keys:

10 HOME:REMCLEAR   SCREEN

a program will be stored in the computer's memory. Strangely, this program will *not* have a LIST that is exactly as you typed it. Instead, the LIST will look like this:

10 HOME : REM CLEAR SCREEN

Programs don't usually LIST the same as they were keyed in 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 command words. The space inserted after these command words can be misleading. For example, if you want a program to have a list like this:

10 DATA 67,45,54,52

you would have to omit the space directly after the DATA command word. If you were to key in the space directly after the DATA command word, the LIST of the program would look like this:

10 DATA 67,45,54,52

This LIST is different from the LIST you wanted. The number of spaces you key after DATA and REM command words is very important.

All of this brings us to the COMPUTIST LISTing format. In a BASIC LISTing, there are two types of spaces: spaces that don't matter whether they are keyed or not and spaces that must be keyed. Spaces that must be keyed in are printed as delta characters (▵). All other spaces in a COMPUTIST BASIC listing are put there for easier reading and it doesn't matter whether you type them or not.

There is one exception: If you want your checksums (See "Computing Checksums" section) to match up, you *must not* key in any spaces after a DATA command word unless they are marked by delta characters.

Keying In Hexdumps: Machine language programs are printed in COMPUTIST as both source code and hexdumps. Only one of these formats need be keyed in to get a machine language program. Hexdumps are the shortest and easiest format to type in.

To key in hexdumps, you must first enter the monitor:

CALL -151

Now key in the hexdump exactly as it appears in the magazine ignoring the four-digit checksum at the end of each line (a "S" and four digits). If you hear a beep,

you will know that you have typed something incorrectly and must retype that line.

When finished, return to BASIC with a:

E003G

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

Keying In Source Code The source code portion of a machine language program is provided only to better explain the program's operation. If you wish to key it in, you will need an assembler. The S-C Assembler is used to generate all source code printed in COMPUTIST. 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 just for this purpose was printed in COMPUTIST No. 17. To translate source code, you will need to understand the directives of your assembler and convert the directives used in the source code listing to similar directives used by your assembler.

Computing Checksums Checksums are four digit hexadecimal numbers which verify whether or not you keyed a program exactly as it was printed 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 programs appeared in COMPUTIST No. 1 and The Best of Hardcore Computing. An update to CHECKSOFT appeared in COMPUTIST No. 18. If the checksums these programs create on your computer match the checksums accompanying the program in the magazine, then you keyed in the program correctly. If not, the program is incorrect at the line where the first checksum differs.

1) To compute CHECKSOFT checksums:

LOAD filename
BRUNCHECKSOFT

Get the checksums with

&

And correct the program where the checksums differ.

2) To compute CHECKBIN checksums:

CALL -151
BLOAD filename

Install CHECKBIN at an out of the way place

BRUN CHECKBIN,A$6000

Get the checksums by typing the starting address, a period and ending address of the file followed by a **☐Y**.

xxx.xxx ☐Y

And correct the lines at which the checksums differ.

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.

Our editorial policy is that we do NOT condone software piracy, but we do believe that honest 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 application programs to meet his or her needs.

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.

FREE COMPUTIST

We were afraid it would come to this.

COMPUTIST is feeling the bite of piracy. It seems that a good portion of our readers don't actually buy COMPUTIST for themselves. Instead, they get an issue after someone else is done reading it. Five (or more) readers per issue is not unheard of. Some get one issue and photocopy it for a whole user's group. We've even received reports of our articles found word for word on BBS systems. The result of all this is that our subscribership is dwindling while the number of readers is not. That's not very healthy for a magazine.

Most magazines support themselves with the money they get from advertising. We try not to. You may have noticed that COMPUTIST's subscription price is slightly higher than other 32-page magazines. Perhaps the foremost reason for this "high" subscription price is that we are dedicated to our subscribers. COMPUTIST is a magazine with a more specific focus than other magazines, that of educating the reader about the inner workings of the Apple // and, of course, software deprotection. We feel that you would rather see information packed issues rather than pages of ads.

Last summer, at the risk of losing some subscribers (which we did), we tried to make up for the increased costs of production by raising our rates. One year later, we are still looking for answers.

It's nice to see that so many people love their COMPUTIST. We would love to see more of that. Which brings up the point of all this...

If you are reading someone else's issue, PLEASE get your own subscription. You will have an issue of your very own every month and we will be better able to serve you by having the money to do so. Tell us whose issue you're reading (he must be a current subscriber) and he will get two free issues.

If you are letting others read your issue of COMPUTIST, urge them to get their own subscriptions. In fact, just to get you to do this, we are offering a bounty. That's right, a bounty. For each new subscriber you get for us, you will recieve two free issues of COMPUTIST added to your own subscription. Just make sure they mention you as the recipient of the free issues. Remember, this is for generating new subscribers only. This won't work for renewals.

If we can get enough subscribers, it will cost us less to print COMPUTIST and we will be able to reduce the subscription rates a little. To illustrate exactly what we mean, at 8,000 subscribers we are barely, if at all, breaking even, while at 16,000 subscribers we will be able to reduce the rates to $15 per 6 issues. That's why we need more subscribers. We'll be better able to serve you, and it will cost less to do so. *ed.*

CHIPWITS™

Michael Johnston & Doug Sharp

Thanks.

COPYRIGHT © 1985   DISCOURSE, Inc.

Do you need anything else?

Information

OUT 0 INNING 1
PITCH HI
INSIDE
OUTSIDE
LOW
DeSoto   RF .317
CHAMPS   0
ALL-STARS 0

This month's cover:
Graphics from Mindscape's "Tink! Tonk!"

COMPUTIST

If you've ever wanted to have better control over the noise your Apple makes, then this is the article for you. With this article you can add a volume control to your speaker, add an external speaker jack and add a line-level output that you can hook to your stereo without worrying about blowing its inputs. *by William Wingfield Jr.*

With this article and the programs accompanying it, you can map edit the entire mainland of Ultima IV. *by Jim S. Hart and Ray Darrah*

Adventure International's **Graphic Adventure #1** *by Edward Hauff*, Sensible Software's **Sensible Grammar & Extended Bookends** *by Charles Taylor*, Brainpower Inc.'s **Chipwits** *by David Kirsch*, Accolade's **Hardball** *by Jeff Lucia*, Sierra On-Line's **King's Quest II** *by Larry Rando*, Epyx's **The World's Greatest Baseball Game** *by Doni G. Grande*

input

Double ROMs Revisited

Following your articles on 32K ROM in COMPUTIST No. 19, and the letter from Wesley Felty in COMPUTIST No. 24, I would like to make a couple of comments.

First, the mods are far simpler to make on the 16K language card. There are two jumper pads marked "2716", one at the top which has to be joined, and one at the bottom which has to be cut with a razor knife. That's all you have to do to run EPROMS instead of the F8 ROM that the card comes with. Clone cards will usually be okay as they are. One more modification is to cut the trace on the underside that runs from pin #21 to isolate that pin. Solder wires from the back of the board, pin #21, to the center of a SPDT switch, and from pins #12 and #24 to the outside two contacts of the switch. The switch can be mounted on the back edge on the same side as the chips, just above the ROM socket. This is then accessible through the slot in the back of the Apple.

This card can be used in other computers that you own if you wish, you don't have to dismantle your Apple, or risk damage to the motherboard. The 32K EPROM works just like it's supposd to, since the F8 on the 16K has preference over the one on the motherboard (in fact you can remove the one on the motherboard if you want).

I have to say this is the best softkey device I have ever used, and recommend anyone who was afraid of modifying the motherboard, to do this alteration instead. The results are unreal! One nice "bug" I've found is with my lower case chip, I already have the flashing underline prompt which of course is still there when in the normal F8, but when I go to the other F8, the prompt changes to a flashing chessboard, so its easy to see which F8 you're using.

I love hardware mods like this. Keep them coming! Does anyone have any ideas on using 64K EPROMS, 2764 to hold bigger stuff such as DOS or Copy][Plus Utilities?

I also have a couple deprotection hints.

Lifesaver deprotects easily with Copymaster II copycard, but the load is very slow. I tried to boot trace it, but I'm not good enough at assembler yet. This new F8 should help though.

Microzine 3.1.5A deprotects easily with the following. Init a new disk with regular or fast DOS by typing INIT HELLO and then DELETE HELLO. BLOAD FID from the system master and type:

```
CALL -151
B942:18 N 8039
```

and FID all the files to the new disk, by using "=" the wildcard file name. That's all there is to it! Both sides work the same way, and are then listable on COPYA.

Finally, a suggestion; How about some tutorials on finding nibble counts. I can usually copy disks to regular DOS, but I have lots of trouble from then on!

Mike Basford
B.C. Canada

Double ROMs Update

Just a follow-up on my letter on the 32K ROM Mod. If pin #6 (or the trace to it) on chip #5, a 74LS20 at the top of the row next to the ROM, is isolated from the rest of the circuit, either by bending the pin out, or better yet cutting the trace to pin #6, and a wire soldered to the pin or pin-side; run it to a simble on-off switch, and connect a wire to ground (edge connector 26) from the other side. This then allows you to select either the F8 on the motherboard or the one on the 16K card. This is all you have to do if the switch is on the card, but I mounted mine at the front of the case near the keyboard, and the extra length of the wire required caused timing problems, which I solved by mounting a 74LS74 on the spare pad at the front of the main board, as described on

pg. 6-19 of "Understanding the Apple][" by Jim Satter.

Also while your soldering iron is hot, run wires from pins 29 and 26 through a 100 ohm resister and momentary push button switch to give you a NMI.

As you can see, there is lots of potential in the 16K card, and I'm now playing with a couple of language (EPROM) cards. If anyone has had any luck using these to load RWTS automatically on boot-up, I'd love to hear from you.

Now for another softkey. I use COPYB, but I'm sure Super IOB with swap controller would work.

Milliken Comprehension Series, twelve disks, has a very annoying protection, and the fact that you can't write protect them means that they are susceptible to damage. However the softkey is easy. Initialize twelve disks with a fast DOS using BOOT as a HELLO program. Delete the file "Boot" on all of them. Then boot anyone of the Milliken disks. Reset into the monitor with any Non-Autostart F8 ROM and move the RWTS out of the way:

```
8000<B700.BFFFM
```

Now boot COPYB and answer all the prompts. Copy tracks 3-34 with error checking off. Don't let the program re-initialize the disk. Copy all the series the same way.

Then load your favorite sector editor and edit tracks $11 Sector $00 Byte $01 from $F1 to $11. Although this is the same on all the disks, the rest of the sector is different, so make sure you read from each disk, do the sector edit, and write back to the same disk.

Thats all there is!

Keep up the good work.

Mike Basford
B.C. Canada

A Bit of History

I have a program I would like added to the Most Wanted List, but first some background.

AppleWorks from Apple Computer, Inc., is the biggest selling program for the Apple in history. No other program, including VisiCalc, (which was responsible for the tremendous growth of the popularity of Apples in the early days) ever dominated the Apple software sales charts the way AppleWorks has done over the past few months. The point of all this is that AppleWorks is a completely unprotected piece of software.

input

Now comes along another wonderful piece of software for the Apple][series of computers, also from Apple Computer Inc. The program is Apple Instant Pascal, and with this version of Pascal, even I may be able to learn Pascal, after trying for these many years.

Apple Instant Pascal uses the Macintosh environment on Apple][computers, with pull-down menus and either mouse control or keyboard control of the cursor. This environment is so spectacular and easy to use that after using the program for a few days, I am seriously considering the purchase of a Macintosh. If Apple Instant Pascal can make a confirmed Apple][user like me consider dumping five years of accumulated software and data for an entirely different computer, and assuming Apple Computer Inc. wants to sell Macintosh computers, I would think they would want as many copies of Apple Instant Pascal in the hands of Apple // users as possible. Now the problem, despite the success of the unprotected AppleWorks, Apple Computer Inc. decided to issue Apple Instant Pascal on, *God Help Us*, protected disks. Not only protected disks, but disks that must be used on both sides.

I have two Apple][drives that are both over five years old. They have never had one bit of maintenance and I have never had one minute of trouble with them. I have never used both sides of disks, except to copy material from the back side of a program disk onto a single-sided disk. I realize the argument continues as to the use of both sides of disks, but for Apple Computer, Inc. to force me to use both sides with a protected program is outrageous.

Thus, I would like to request that Apple Instant Pascal, a super program for learning Pascal, be placed on the Most Wanted List.

When will software producers understand that copy protection only frustrates and annoys honest users and in no way even slows down professional pirates. If anyone doubts this, just take a look at the hundreds of "protected" program available on unprotected disks from Reliant Software in Hong Kong.

Thomas E. Militello, M.D.
Rancho Palos Verdes, CA

Mr Militello: We think you'll be pleased with the article on page 26 of this magazine.

Another Windham Classic

After seeing Allan J. Migdal's softkey for Windham classics in COMPUTIST No. 29, I realized that while we were deprotecting the same program we were dealing with two different protection schemes. My version, tested only on Alice in Wonderland, has the following files: Hello, Windham, Game, and Gmae (a deleted file)

In my version, there is no program data on track 2. To make a deprotected disk copy both sides of Alice with COPYA and add a normal DOS to side one. That's all!

Finally, yours is easily the best Apple magazine on the market. Without it I'd have given up long ago on understanding this machine. Keep it up!

John Della Pia
Bolling AFB, DC

Unjamming Eddie's Hatch

Enclosed, please find my check for another six issues. Again I thank and commend you for publishing the finest magazine on Apple computers to be found anywhere. Keep up the good work.

I read Rob Muscoby's Wizardry APT in COMPUTIST No. 29 and have an answer for his question regarding the Hitchhiker's Guide to the Galaxy.

1) Take all objects offered to you during the game and keep them if possible.

2) Things fit well in something that you don't know what it is. This is not necessarily true for fluff.

3) Hungry? Eat whatever you might grow during your travels. Consult the Guide about fluff.

4) If the Hatch is jammed by a robot, why not ask a very sulky one to open it for you? After all, he is programmed to be a servile domestic. He may ask you for something you don't have. In that case, you either blew it or lost it and have to start somewhere earlier to get it.

5) Things have a bad habit of disappearing.

6) A flippy gives more room to save to disk the various parts of the game.

7) Don't Panic.

In other words, keep all the tools and weird stuff you find in the game. Put it all in the thing your aunt gave you that you don't know what it is. Sometimes as the game progresses, things will disappear. Inventory your goods frequently. Save to disk frequently. Invariable, the tool or bit of fluff that you need will disappear unless you keep track of it. When you get all four bits of fluff, plant them in the flower pot that you find in the whale. When the plant

sprouts, enter the sauna and then examine the plant. It will have a fruit. Eat the fruit. You will have a revelation wherein you will see Marvin and he asks you for a tool. You better have the tool in your inventory or you will have to restart somewhere in the game to get it. Go to the Screening Door and show it Tea and No Tea simultaneously. You better get thirsty about here. Drink the tea, as it will cheer you up. Enter and ask Marvin to open the Hatch. Begrudgingly, he will acquiesce. Enter the access space and wait for Marvin to shuffle in. When he asks for the tool, give it to him. The Hatch will open. Leave the Heart of Gold and stand on the surface of Magrathea.

John Gubbins
Aurora, CO

A Pirate Speaks Out

I am writing this letter in reply to the "Holy man", Mr. Donald G. Moses (COMPUTIST No. 28). Face it, copying is in! (to quote the Black Rose). Anyone who insists on buying originals is an idiot. Mr. Moses should check with current economic policies of the world. Demand for programs is great, therefore, the criminals running the software companies will raise their prices. To compete directly with this, my "heroes" in Asia are selling programs at approximately 1/10 of the price. The demand for the Software Companies' goods at large prices will decrease, while demand for the Asian product will increase. Anyone looking for programs should write to Hong Kong, Singapore, or Malaysia.

I think you are disgusting (and demented) if you think the public will buy originals rather than copies. It's not a matter of stealing, but supply and demand! Whereas the consumers can find cheaper substitute goods, he will - the basis of our "free enterprise" system.

All I can say is that most software companies charge sometimes hundreds of dollars for programs worth 1/10 of that! Now that is criminal!

The Agent
Soviet Software

input

The American Challenge

Here is a softkey I have recently completed. The program is The American Challenge, a sailing simulation. Here is the cookbook method.

The American Challenge

1) Load COPYA and hit ☐C when it asks you for the slot and drive numbers.

2) Delete line 70.

   70

3) Enter the monitor.

   CALL-151

4) Change $B942 from $38 to $18 so that COPYA will ignore any errors.

   B942:18

5) Re-enter Basic.

   3D0G

6) Run COPYA and copy the disk.

7) With a sector editor, find the sequence 4C 00 C6 on track 0, sector B. Change these to 4C DB 43 and you are done.

> John Cotter
> Bay City, MI

Robotron APT

Here is an APT for Robotron 2084 by Atarisoft. With this APT, you can get infinite men (I think). First, obtain a cracked Robotron. (The softkey in COMPUTIST No. 8 will work.) Then do the following:

1) Boot up standard DOS.

2) Load the broken Robotron file.

   BLOAD ROBOTRON

3) Enter the monitor and modify Robotron for infinite men.

   CALL -151
   326A:99

3) Start up the game.

   2DFDG

To make the last human family move much faster, substitute **3208:99** in place of **326A:99** in step 3.

> Mad Max & The Fool

Where in the World is Murder by the Dozen

I have found that Ronald Wilson's softkey for ''Where in the World is Carmen Sandiego?'' (COMPUTIST No. 25) did not work on my //e (with absolute reset into monitor) due to nothing in memory at location $2000 (step 3). I preceded this step with 2000<0000.07FFM and it worked perfectly.

I have also found that Randy Ramirez's softkey for ''Murder by the dozen'' (COMPUTIST No. 29) does not work as written. Whenever I try to copy files using FID (step 3), I always get a ''No Files Selected'' message. HELP!!

I find your magazine very helpful and informative. Thank you.

> Robert Hall
> APO New York, NY

Mr Hall: To copy every file using FID, you should select option 1 (the copy files option) and then use ''='' as the filename. If you still get a ''No Files Selected'' message then you have a different version of this program.

---

bugs

COMPUTIST No. 32:

Revisiting Music Construction Set:
The sector edits under step 2 should read:

Track	Sector	Byte	From	To
D	F	36	20	18
D	F	37	00	60

readers' softkey & copy exchange

Edward Hauff's antique softkey for...

Graphic Adventure #1

Requirements:
FID
A blank disk

Adventureland from Adventure International, while being an older graphic adventure-type program, is still one of the best made. It can be easily copied with FID, however the first time you try to pick something up, the program does a check for the original disk. After this one check, no other checks are made unless a previously saved game is played, in which case it will do it on the first pickup of that session.

This would seem to mean that a flag byte is used as a reference for the check. I compared the bytes in the game save area just before and right after disk access to see which bytes were changed. Only a few were changed, and as I replaced them one by one with their original values. Location $4165 which was changed from $FF to $00, caused a disk check to be reinvoked. The program "M2," which is responsible for drawing all the rooms, keeping track of where you are and what you are doing, saves all its information in the area where the saved games are loaded. A LDA $FF followed by a STA $4165 was found at $6B16 of the M2 program. Changing the $FF to $00 prevents the disk check from occurring.

In short:

1) INIT both sides of a disk with the name HELO (or was it HELLO?) and delete that program.

2) Enter the monitor and disable DOS' error checking.

        CALL-151
        B942:18

3) Get FID started and copy all files on both sides.

        BRUN FID

4) Load M2 from the boot side.

5) Enter the monitor and change byte $6B17.

        CALL -151
        6B17:00

6) Save the program back to disk.

        BSAVE M2,A$6860,L$0DF0

Charles Taylor's softkey for...

Sensible Grammar & Extended Bookends

Requirements:
COPYA
A sector editor
A disk searcher
A blank disk

Sensible Grammar goes one step beyond proof-reading documents for correct spelling. It checks your manuscripts for correct punctuation and searches for commonly misused phrases to find pompous, informal, cliche, vague, wordy, repetitive, and sexist phrases. It finds all uses of apostrophes, right or wrong, so you had better know how to use apostrophes for posessive use.

Extended Bookends makes full use of a 128K Apple.

Sensible Software has come full circle from very difficult copy protection to very simple protection. (Sensible Grammar can be copied by the "sector copy" from Copy || Plus.) Both programs are readily deprotected by the Mr. Grande's method in COMPUTIST No. 27, page 8. I found the offending code for Sensible Grammar on track 4, sector 4, and for Extended Bookends on track 0, sector $F.

In summary,

1) Copy the original disks with COPYA.

2) Search for the byte sequence BD 8C C0 10 FB 49 D5 D0 F7. Change the final byte ($F7) to $56 and write it back to the disk.

David Kirsch's softkey for...

Chipwits

Requirements:
Apple][
Super IOB 1.5
A blank disk

Chipwits are computerized robots you program to fend for themselves in eight different adventure environments. You can teach a Chipwit to do just about anything: move in any direction, avoid enemies, attack enemies, search for food using different senses, even sing songs!

The Protection

The first thing I do when I realize that a program will be writing to the disk is attempt to copy that disk. After getting this game that is what I tried to do. Copy II Plus 6.0 was no help, neither with parameters nor manual bit copy. Although it told me that tracks 0 - $21 were standard format I still couldn't copy track $22.

A track monitor (a hardware device that tells you what track your disk drive is on) can come in handy sometimes. A while back COMPUTIST ran ads for one that sold for $99.00. I thought that $99.00 was a little too expensive, so I built one for about $25.00.

Back to Chipwits: while booting the program the head first goes to track 0 then to $B and goes backward from $B to 1, jumps back to $B, then to the infamous track $22. Track $B looked like a likely spot to look to see if the copy-protection could be bypassed. Using the sector editor in Copy II Plus I found a strange routine on sector $D that even included the "hang" routine that branched to itself after writing out a disk error message.

At this point I felt that I was very close and that maybe in a few days I would have it bypassed. I tried bypassing the "hang" routine, but that didn't work, and after looking further I realized that it couldn't have worked. Just for fun I put a RTS ($60) at the beginning of the sector and booted the disk.

Hurrah! It worked just fine and booted faster because the track $22 check is now completely taken out.

Procedure

1) Type in the controller below into Super IOB 1.5.

2) Start the program and let it format the blank disk with a volume number of 254.

3) Use it to copy tracks 0-$21 of Chipwits. Super IOB will do the sector edit for you.

4) And there you have a copyable Chipwits that each of your children can play with without destroying each other's robots, or copying each other's logic panels.

5) I almost forgot. The write protect notch must NOT be covered to run the program, since even simple things such as changing robots write to
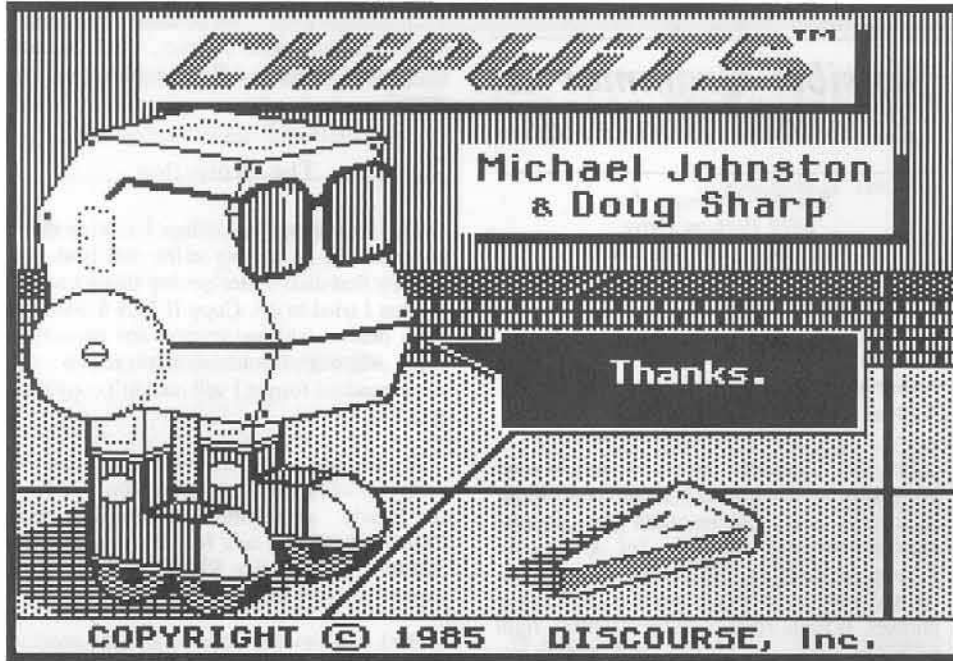
the disk so that the next time you boot up, the program is just as you last left it.

controller

```
1000 REM CHIPWITS CONTROLLER
1010 TK = 0 : LT = 34 : ST = 15 : LS = 15 : CD = WR : FAST
     = 1
1020 GOSUB 490 : GOSUB 610 : T1 = TK : TK = PEEK (TRK
     ) - 1 : RESTORE : GOSUB 310 : TK = T1
1030 GOSUB 490 : GOSUB 610 : IF PEEK (TRK ) = LT
     THEN 1050
1040 TK = PEEK (TRK ) : ST = PEEK (SCT ) : GOTO 1020
1050 HOME : PRINT "COPYDONE" : END
1100 DATA 1^ CHANGES .11 .13 .0 .96
```

Larry Rando's softkey for...

King's Quest II

Sierra On-Line
Coarsegold, CA 93614
$35.95

Requirements:
128K Apple //e or //c
Fast copy program
Sector editor

This is the sequel to Sierra On Line's King's Quest which uses three-dimensional double hi-

res graphics and has animated characters that you control with either keyboard or joystick. Sierra On-Line is known for putting out terrific animated graphic adventures and the first adventure game that utilizes double hi-res on the 128K Apple //e or //c. If you liked the first King's Quest you will enjoy this one.

Procedure:

1) Copy sides 1-5 with any fast copier or COPYA. (Sides 2-5 are not copy-protected).

2) Get out your sector editor and change side 1, track $11, sector $0F, byte $05, from $A9 to $60.

3) Write it back out to the disk.

That's it!

Doni G. Grande's softkey for...

The World's Greatest Baseball Game

Epyx
1043 Kiel Court
Sunnyvale, CA 94089

Requirements:
48K
Super IOB 1.5
A blank disk

This program allows you to select from twenty-five classic baseball teams and pit them against one another. You can select the line-up and manage your team's strategy. You can manage and control the players or just manage the team and let the computer statistically determine how each player plays.

The Super IOB controller below will make a COPYAable version of the World's Greatest

Baseball Game. Just install it into Super IOB v1.5 and copy the disk. That's all there is to it!

controller

```
1000 REM WORLD'S GREATEST BASEBALL GAME
1010 TK = 0 : LT = 35 : ST = 15 : LS = 15 : CD = WR : FAST
     = 1
1015 POKE 47405 ,24 : POKE 47406 ,96 : POKE 47497
     ,24 : POKE 47498 ,96 : REM IGNORE EPILOGS
1020 GOSUB 490 : GOSUB 610 : T1 = TK : TK = PEEK (TRK
     ) - 1
1025 RESTORE : GOSUB 310 : TK = T1
1030 GOSUB 490 : GOSUB 610 : IF PEEK (TR ) = LT
     THEN 1050
1040 TK = PEEK (TRK ) : ST = PEEK (SCT ) : GOTO 1020
1050 HOME : PRINT "COPY^ DONE" : END
5000 DATA 1^ CHANGES
5010 DATA 0 ,5 ,172 ,24
```

controller checksums

1000	– $356B	1030	– $6B89
1010	– $2544	1040	– $9C15
1015	– $7A56	1050	– $A6C6
1020	– $EE04	5000	– $CC10
1025	– $6367	5010	– $211E

Jeff Lucia's softkey for...

Hardball

Accolade, Inc.
20863 Stevens Creek Blvd., B-5/E
Cupertino, CA 95014

Requirements:
COPYA
A disk search utility
A sector editor
A blank disk
Hardball (or other Accolade program)

Accolade is a new company founded in 1985 by a group of Activision veterans (namely Bob Whitehead and Alan Miller).

Accolade's programmers must be given credit. The authors take the time and effort to get their programs on the market. Since Accolade is a new company, many people have not yet been given the chance to see the quality and enjoyment that Accolade's programs bring.

Hardball has got to be the best computer baseball game I've ever seen (it gets my vote for game of the year)! Its graphics and realism just cannot be beaten by any other baseball game on the market.

Because Accolade is producing high quality programs like Hardball, they want to make their disks extremely hard to back up. However, it's not that hard.

The Protection

Accolade's main protection is quarter-tracking. Upon booting the program, the disk will check for a series of quarter-tracks. While checking for the quarter-tracks, it scans for certain byte patterns. Because of this, Copy II Plus, EDD III, Locksmith 6.0 and Nibbles Away II will have trouble making a backup.

Because I'm always concerned about my original disks, I decided to defeat Accolade's protection.

The following code came from Hardball:

```
B260- CLC            clear carry flag
B261- BCC $B272      branch if carry clear
...
B272- SEC            set the carry flag
B273- BCS $B2F3      branch if carry is set
```

This is the tricky part. The code is intentionally clearing the carry flag. Then the program will jump to other locations that set and clear the carry flag. this is their way of making it harder to follow. (Note that the branches shown here will always be taken.) Finally, after following their branches, you will find the main protection. Although the location of this code will be different from program to program, it is the start of the routines that will check the disk for quarter tracks. The way to defeat this is simple. We will insert code that will clear the carry and return to the calling subroutine. Once this is done, the disk will be deprotected.

The Fix

1) Use Copy II Plus (or whatever copy program you have) and copy the disk.

2) Start your disk searcher and look for the byte sequence **$A0 09 B9**.

3) Use your sector editor to replace them with **$18 60 B9** and write the sector back to the disk, and you're all done.

I have tested this method on Dam Busters, Law of the West, and Hardball. Usually, if the same protection is on three or more disks, the company is using it for all their programs (that's just a theory). Some companies use a protection on two to four disks and then change it.

softkey for...
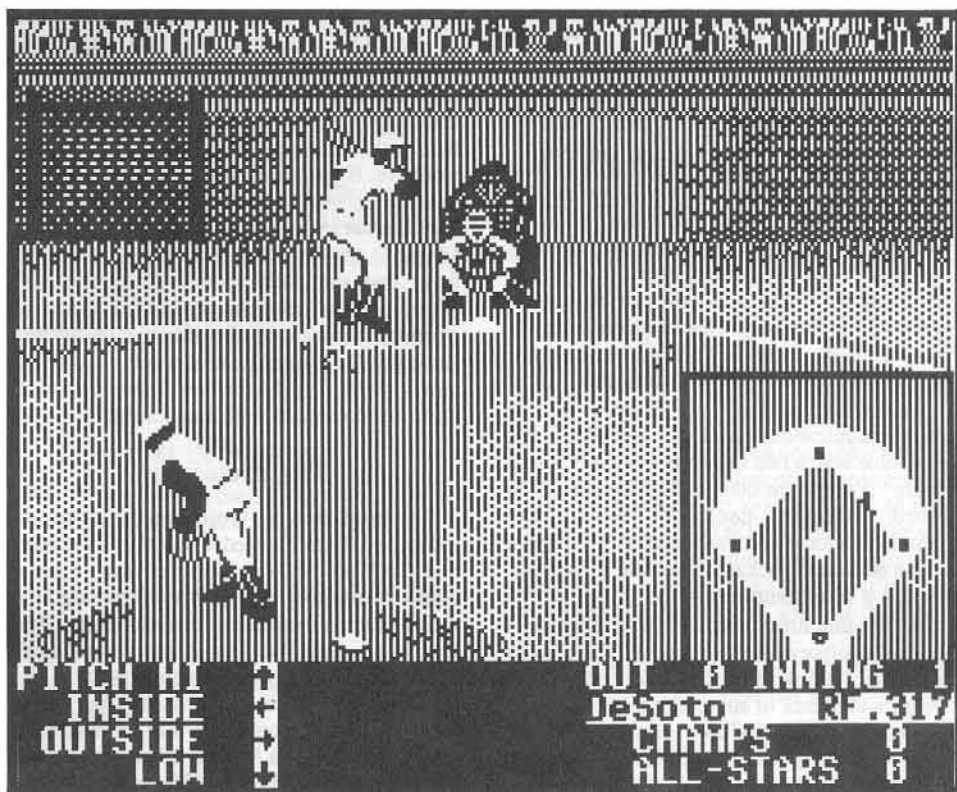
Word Juggler

by John Nice

Quark, Inc.
2525 West Evans, Suite 220
Denver, CO 80219
(303) 934-2211
List price $189

Requirements:
Apple //e
ProDOS User's Disk
DOS 3.3 48K slave disk
1 blank disk
Word Juggler //e

In my opinion Word Juggler is, by far, the best word processor for the Apple //e. It has several features that are not matched by any other program on the Apple line. Its most unique feature is the keycap replacements. When you purchase Word Juggler, you recieve a plastic tray holding 19 new keycaps to put on your //c keyboard. These keys have the function words on the fore front. This allows you to get typing right away. You don't have to read and re-read, then look at the quick reference chart, to see what the command is to "delete a block" of text. You simply look at the keyboard. And, as most people, I do that a lot!

The simple commands aren't the only neat feature either. Word Juggler is actually a system, not just a word processor. Included with the package is an extensive spelling checker. You can either check the whole document for misspelled words or choose the Word Guess function. This function looks for the word in the dictionary that the cursor is on. A good feature for quick documents.

Another neat accessory is a telecommunications package included. It's accessible with a two key combination (Closed-Apple "5") like the other extra functions in the Word Juggler package. Now, the communications software couldn't compete with a dedicated package, like ASCII Express, but it's a nice feature anyway.

Before this turns into a review, why am I writing a softkey for this product? The answer is simple, I am writing this softkey because of Quark's total lack of adherence to the ProDOS standard, unneccesary copy protection, Word Juggler's load speed, and diskette replacement costs.

I don't use any of the original disks I purchase because of the possibility of bombing, the enhanced resale value, and the general one-sidedness of commercial products. On bombing, Quark charges $10 to $15 (up to $25 for Lexicheck, the spelling checker) to replace or repair the disk. If I ever wish to sell a used piece of software, a thrashed-upon original would definitely lower the prospective price. Another reason is the supplied disks from Quark are all one sided. That means you need two of their disks just to operate the word processor. A double-sided disk works just fine and doesn't clutter up my desk.

The boot process on a backup disk bit copied with Copy][Plus, ranges into the minute-plus category. The cracked version loads in seconds.

Quark copy protected this package even though the protection isn't needed. With the keycap replacement, the command strip, and the quick reference triangle, any use of this software on the machine it wasn't set up on is nearly impossible. I have even tried to use Word Juggler on another machine, and have found it troublesome to remember the commands. Since they are usually right in front of my face, when I am writing seriously, I don't try to remember them.

The biggest reason for the crack is because of ProDOS itself. A big goal of Apple Computer in rewriting its disk operating system was to make it device independent. You shouldn't need to worry about what DOS to have in the machine to access each specific drive you have. If you have a hard drive and a floppy, you can access them all in a similar method. This is what I wanted to do with Word Juggler. I want to put my word processor on my hard drive and stop fiddling with these floppies. Word Juggler requires at least 2 disks to operate, because of the sheer size of this package. On a hard drive I wouldn't have to worry about flipping the disk before I want to scan my document with LexiCheck.

The Protection

The protection on Word Juggler is not too extensive. When I booted up my /USERS.DISK and CAT'ed the Word Juggler disk, I got a complete directory listing. When I examined the catalog listing I saw some unusual things. The first was that the Word Juggler program wasn't the startup system file (the first .SYS in the catalog). So, this signaled that they put on the copy protection after Word Juggler had been written a good sign. If Word Juggler was the startup application it would probably have

demanded an extensive dissembly to find the special loading routines.

Next, I got Filer up and compared the ProDOS system file, on the Word Juggler disk, to a known normal ProDOS system file. This turned out OK (it's a good thing too, as Apple Computer says modifying ProDOS is an extreme no-no unless it's for driver installation). Then I scanned for bad blocks on the disk and found out that tracks 1 & 2 (blocks 8-23) were bad. Next I tried to copy the WJ2E.2.8.3 file and WJ2E.SYSTEM file to another disk (since ProDOS doesn't have a verify command like DOS 3.3, you have to use a roundabout method of checking for corrupted file formats). Well, as it turned out the WJ2E.2.8.3 was the bad (copy protected) file and the WJ2E.SYSTEM file was OK. This made sense as the first 'xxx.SYSTEM' file ProDOS comes across during the booting process is the file used to initialize the system. If this file was bad, the ProDOS would have had to be modified to read its special format but I knew that ProDOS was intact and unaltered.

I then copied the rest of the files to test them out too. All of them copied perfectly. That left WJ2E.SYSTEM as the file to disassemble. I knew that WJ2E.SYSTEM would have to load the protected Word Juggler application file, which was partly on tracks 1 & 2. After an extensive and fun dissembly I determined the main logic of WJ2E.SYSTEM to flow as follows:

a) Figure out what the load address is (for Word Juggler //e this is $2000).

b) Copy this loader to upper RAM (from $2000-$26FF to $B000-$B6FF).

c) Determine if the computer is a //e.

d) Redirect RESET, BRK, and ProDOS System death pointer to go to a memory wipe handler.

e) Initialize the system hardware to alternate character set, text mode, and 40 columns.

f) Clear the screen using quick clear method.

g) Print the introduction screen.

Then a jump is taken to the relocated code section in upper RAM. This code makes up the routines that load in the special sectors of the WJ2E.2.8.3 file. The logic continues as such:

h) Set some Word Juggler system variables (i.e. slot*16, drive index, etc.)

i) Make sure it's in a //e, otherwise rebuke the booter, then kill the system.

j) Check for an Apple 80 column card. Whether it is or not, continue on. (The requirement that

only Apple's 80 column card be used was dropped, so this code was invalid. It is now never used.)

k) Set some more variables and pointers.

l) Read in the key block at $00/$0F (Track/Sector). (All this information about WJ2E.2.8.3 is ignored by WJ2E.SYSTEM. This read is mainly to position the disk arm to track 0)

m) In a roundabout way compute the number of sectors (not blocks!) to read for the WJ2E.2.8.3 file. This is all just code protection. It could have easily been done with an immediate load.

n) Read in a sector using RDSCT ($B21B) routine (see below).

o) Combine a checksum of a page of the loader (WJ2E.SYSTEM) with a checksum of the sector just read.

p) Bump the block (thus the t/s) number and buffer pointer.

q) Loop back to (n) until done.

r) When done, read in one more sector ($00/$00) to ensure there was no fiddling with the loader code.

s) Jump to the start of WJ2E.2.8.3.

The RDSCT routine logic starts by turning on the drive and seeks the correct track and sector. When the data field is loaded in, the sector is decoded using the physical sector number EOR'ed with $6B. While this is done, a checksum is computed that will be used in another checksum of the entire file.

The other noteworthy code in WJ2E.SYSTEM is in the RDADDR (read address) routine and the SEEK (seek disk arm to proper track) routine. The code in RDADDR checks to see if it's reading a sector in on tracks 1 or 2, and jumps to a special read address section if true; otherwise it reads the address field in as a normal one. This confirms the BAD BLOCK detections by the Filer utility earlier. The SEEK routine contains some logic to bomb the program if certain conditions aren't met. These are, that the carry is set whenever the RDSCT routine is called. If it isn't, then that is supposed to be the last call to the RDSCT routine, so the file checksum is tested.

In all cases, if an error is detected a section of code will print an error message. Then some clear memory code is moved to the Zero Page and loops until a Reset forces a reboot.

The Crack:

The first thing to do in cracking this program, is to get an image of the main file into memory. To do this we must modify a copy of WJ2E.SYSTEM. The modifications will tell the loader code to ignore any checksum errors, and redirect the jump to the start of Word Juggler to go to the system monitor instead.

After this is done, it can be executed from memory and it will load in the main Word Juggler code. After the boot we will have an image of WJ2E.2.8.3 in memory. Then we have to get it to disk. Unfortunately, ProDOS

can't be booted around a file as large as Word Juggler, so a 48K DOS 3.3 slave disk must be booted to save the file to disk. A slave disk boots in such a way that it won't disturb most of main memory. After DOS 3.3 is booted, you can BSAVE the Word Juggler file image to disk.

Once we have a copy of the file on DOS 3.3, we must CONVERT it over to a ProDOS disk. Then, we must change it from an application file to a system file. To do this we will CREATE a system type file called WJ.SYSTEM, then BLOAD the Word Juggler image into memory and BSAVE it to the system file we just created.

Now we need to make a disk with Word Juggler as the boot system file. Just format a blank ProDOS disk using Filer, then copy ProDOS as the first file on the disk. Copy WJ.SYSTEM as the 2nd, and the rest of the utility files on the original WJ disk (especially the parameter files!!) to the newly created disk and you have yourself a cracked version of the best word processor on the market.

Step By Step

The following is a step-by-step method for cracking Word Juggler v2.8.3. Write-protect your original disk and follow along.

1) Format a blank 48K DOS 3.3 slave disk and delete the HELLO program.

2) Boot ProDOS and get the Filer utility running.

3) Format a blank ProDOS disk with a volume name of /WJ.

4) Copy "ProDOS" to /WJ.

5) Quit Filer back to /USERS.DISK/BASIC. SYSTEM (that is, get BASIC up and running).

6) Create a system file on your new disk to save the Word Juggler main program to later.

CREATE /WJ/WJ.SYSTEM,TSYS

7) Insert the original disk and load in the Word Juggler loader.

BLOAD /WORD.JUGGLER/ WJ2E.SYSTEM,TSYS,A$2000

8) Drop into the monitor to modify the loader.

CALL-151

9) Patch the exit of the loader to turn off the language card and jump to the monitor rather than to the start of the program. (STA $C081; JMP $FF59).

21F0:8D 81 C0 4C 59 FF

10) Kill the code in the SEEK routine that will reboot if the file checksum isn't 0 (use 12 NOPs).

242C:EA EA EA EA EA EA EA EA EA EA EA EA

11) Execute the loader. The program will load and then drop into the monitor.

2000G

12) Insert a 48K DOS 3.3 slave disk into the drive and reboot.

13) Save the Word Juggler application to the DOS 3.3 disk.

BSAVE WJ.BINARY,A$2000,L$6900

14) Insert /USERS.DISK and boot it.

15) Select Convert from the Startup menu.

16) Convert WJ.BINARY on DOS 3.3 to /WJ/WJ.BINARY (move the binary file from DOS 3.3 to ProDOS on your new /WJ disk).

17) Quit back to BASIC.SYSTEM on your /USERS.DISK and hit B to go to BASIC from the Startup menu.

18) At this point you should have PRODOS, the empty file WJ.SYSTEM, and the file WJ.BINARY on your new disk. Now copy the code from WJ.BINARY to WJ.SYSTEM.

BLOAD /WJ/WJ.BINARY,A$2000
BSAVE /WJ/
WJ.SYSTEM,TSYS,A$2000,L$6900

19) Delete the binary file.

DELETE WJ.BINARY

20) Insert your /USERS.DISK and use the "Smart RUN" command to enter FILER.

-FILER

21) Copy all the files except PRODOS, WJ2E.2.8.x, and WJ2E.SYSTEM from the original disk to your new disk /WJ.

Summary

You now have a cracked version of Word Juggler //e v2.8. You should probably clean up the directory by formatting another disk and copying only the needed files (PRODOS, WJ.SYSTEM, WJ.CONSOLE, WJ.PARAMS, TS.PARAMS, all EXT files, and your single printer driver file). With this all done your word processor will boot in seconds, instead of the minute it took on the original.

Another pressing problem is the #9 option (QUIT) from Word Juggler's main menu. The normal quit procedure for ProDOS applications seems to have been ignored by the people at Quark. I don't know if it's because they need the quit code space ($D100-$D3FF in 2nd bank of the language card) for Word Juggler code, or that they just want to keep protecting until the end. I'll be proceeding to wade through the K's of ML code disassembling like a maniac to see if this is true.

I have only encountered one minor problem with the cracked version of this program. It involves ProDOS prefixes. You have to make sure the prefix is /WORD.JUGGLER (or whatever your disk's volume name is) before you run WJ.SYSTEM. If you don't, then WJ.SYSTEM won't be able to find your parameter files!

I have NOT tested it using WJ.SYSTEM on a hard drive. So, if you put it on your hard drive, be careful. Make sure you have as much as possible backed up, just in case.

softkey for...

Tink!

by Pete Ashdown

Tonk! in the land of Buddy-Bots
Mindscape, Inc.
3444 Dundee Road
Northbrook, IL 60062

Requirements:
48K
Super IOB 1.5
A disk searcher
A sector editor
A blank disk

Working for teachers has certain advantages. The main advantage I have found is that the teachers buy lots of protected software. This allows me to deprotect their software when they receive it. Unfortunately, most of the software is early-age educational. But it is still educational for crackers!

One day, I was handed "Tink! Tonk!: Tonk in the Land of Buddy-Bots". The title didn't sound very appealing, but the protection did. Mindscape has been putting out some very good protections recently. So I set out to deprotect it.

I tried to bit copy the disk first. When I booted the copy, it would just reboot continually without loading any further tracks. This is a tip-off to the fact that there was a check on track 0.

Looking at the disk with a nibble viewer shows normal address and data headers, with the trailers changed from DE AA to FF FF. Turning the disk into a normal format in this case can be as easy as turning off the error-checking in DOS and using COPYA. This is the first thing that should be tried in deprotecting any disk. You would be surprised at just how many programs can be converted in this way. Change the SEC ($38) at $B942 to a CLC ($18).

```
CALL-151
B942:18
```

or

```
POKE 47426,24
```

Now use COPYA or another sector copier to copy the original disk to the blank disk. I used Super IOB and a modified controller. There was a longer routine in Super IOB 1.0 for this purpose. I wanted to use the new fast copy feature in Super IOB 1.5 so I simply inserted the POKE in my controller.

The protection gets a little bit tricky from here. I tried booting the disk I had just copied and found that it just continually rebooted like the bit copy. What needed to be done was what I like to call the "Simplest Boot Trace." It involves pressing Reset during the boot. It can usually be done with no problem on continual reboots. But some programs will change the Reset vector to point to a freeze routine or clear & reboot. You would have to use a definite way into the monitor for these. "Tonk" was not one of these, however. To do it, insert the original disk and boot it.

```
PR#6
```

Now listen for the drive head to move past track 0, then immediately hit Reset. If you found yourself in BASIC, enter the monitor with:

```
CALL-151
```

The Boot ROM on the drive controller card loads track 0 sector 0 into page $08 ($0800). This is what is called a "bootstrap loader". This loader loads the other tracks. Thus pulling the program up "by its own bootstraps". Upon inspection of page $08 I found it was pretty normal. It loaded in the rest of Track 0 then did an indirect JMP to the address at $08FD. Location $08FD contained a $00 and $08FE contained a $B7. This meant that the indirect JMP went to $B700. I listed $B700. The first thing I found was a JSR to $BB00. So on a hunch, I NOP'ed the JSR.

```
B700:EA EA EA
```

I then inserted the normal copy I had made previously into the drive and started the routine.

```
B700G
```

The rest of the disk loaded just fine and the program worked perfect. This meant that the only check was at $BB00. So I used the Core Disk Searcher to find the JSR $BB00. This is done by searching for the hex string "2000BB". I searched the whole disk just to be safe and found only once on track 0 Sector 1 at byte 00. I then used a sector editor to remove it.

Tonk!

Track	Sector	Byte#	From	To
0	1	$00	$20	$EA
0	1	$01	$00	$EA
0	1	$02	$BB	$EA

That's it! You now own a deprotected copy of "Tonk in the Land of Buddy-Bots." It is possible that the other "Tink! Tonk!" programs are protected in a similar manner. I know of three more of them right now; "Tuk Goes to Town," "Tink's Adventure" and "Tinka's Mazes". I have not tried this procedure on them, so this is only a thought.

Below is the controller for Super IOB 1.5. It does everything needed to deprotect the disk.

controller

```
1000 REM BUDDY-BOTS CONTROLLER
1010 TK = 0 : LT = 35 : ST = 15 : LS = 15 : CD = WR : FAST
     = 1
1020 POKE 47426 ,24 : GOSUB 490 : GOSUB 610 : IF
     TK = 0 THEN GOSUB 310
1030 POKE 47426 ,52 : GOSUB 490 : GOSUB 610 : IF
     PEEK (TRK ) = LT THEN 1050
1040 TK = PEEK (TRK ) : ST = PEEK (SCT ) : GOTO 1020
1050 HOME : PRINT "COPYDONE" : END
1100 DATA 3ª CHANGES
1110 DATA 0 ,1 ,0 ,234
1120 DATA 0 ,1 ,1 ,234
1130 DATA 0 ,1 ,2 ,234
```

controller checksums

1000	– $356B	1050	– $0582
1010	– $2544	1100	– $46E8
1020	– $297C	1110	– $02AB
1030	– $A469	1120	– $2C24
1040	– $84D6	1130	– $E422
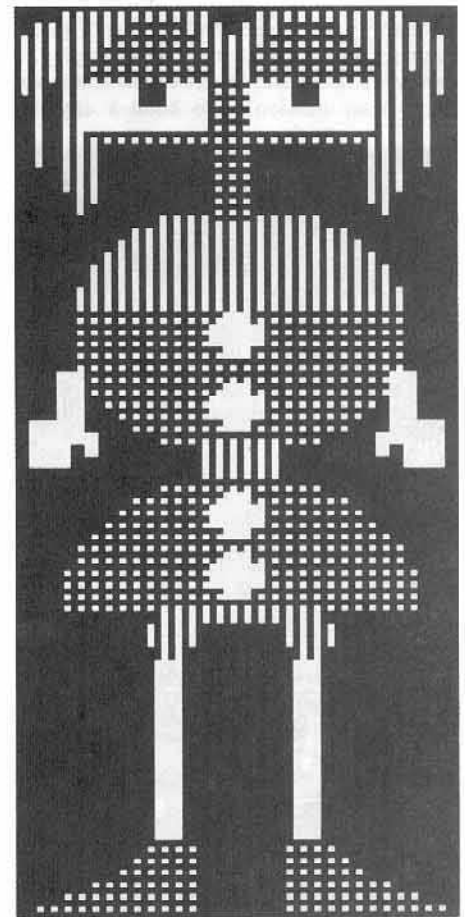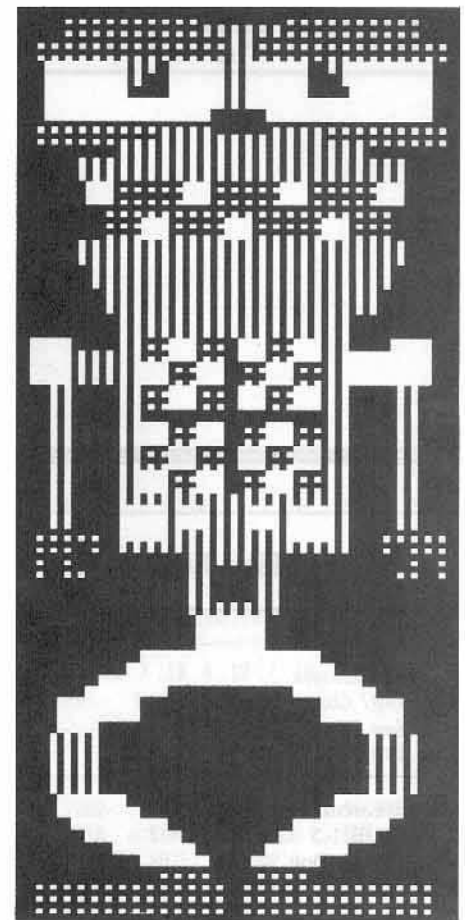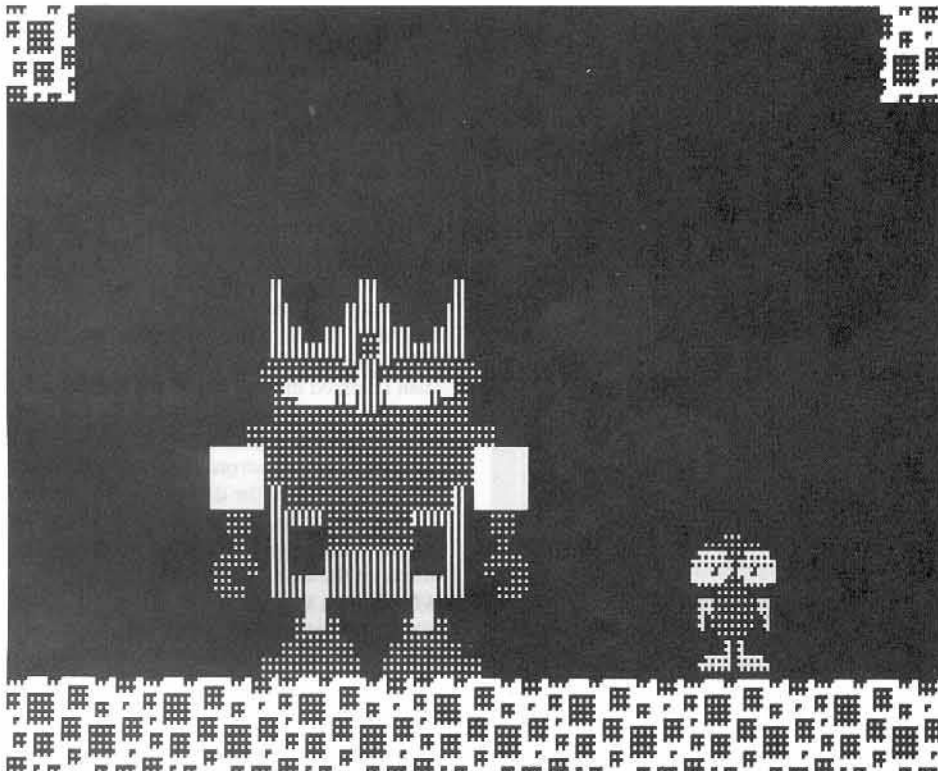
softkey for...

Sundog

by Marc Lirette

Requirements:
Super IOB 1.5
Original SunDog v2.0
2 blank disks (or front and back of one)

SunDog is a fantastic action/strategy adventure game. It has sharp, realistic, quality graphics which makes it attractive to play. It involves strategy and creates excitement as you play. Your mission is to build a city and transport frozen colonists to your city. This may sound easy, but is becomes tricky when space pirates try to steal your money or your valuable cargo. The worst thing that could happen to a person playing the game is to have the disk "crash". This not only costs money to replace, but also puts your mission on hold for a month or two. That is the reason I decided to make a COPYAable version of my disk.

The Protection

The first thing I did when trying to unprotect SunDog was to try to copy the disk. I used Locksmith's fast disk backup because it tells me which tracks and sectors are protected and which are not. Much to my surprise, I found that the front side actually copied without any problems.

I then tried the back side of the disk and found that it too was unprotected except for track $22. Instantly I came to the conclusion that track $22 must be a nibble count or nibble search track.

After taking my disk drive apart and booting the copy I had made, I found that my conclusion was correct. Shortly after inserting the back side of the disk, the program went to track $22. After not being able to find the correct sequence of bytes, the disk arm recalibrated and moved over to track $23. After not finding the correct sequence of bytes again, it recalibrated and the arm moved over to track $22. It did this a few times, stopped, and put me into no man's land. I concluded that track $23 was supposed to be identical to track $22 and was put there as a backup system in case track $22 failed to read properly.

Satisfied that track $22 was the nibble search track, I got out TRAX, from Bag of Tricks, and examined the raw dump of track $22. The dump revealed the following sequence of bytes: D5 D7 FF E7.

The Search

Well, now that I knew what the program searched for, I searched for the program that did the search. I first used ZAP, from Bag of Tricks, and searched for a routine that read the disk and compared with D5 D7 FF E7. I searched both sides of the disk for the sequence BD 8C C0 10 FB C9 D7 (LDA C08C,X; BPL; CMP #$D7) which would be part of the nibble search routine, but I did not find any. I then booted the original disk and waited for it to do its nibble search. As soon as it went to track $22, I pressed Reset until the drive stopped. Then I entered the monitor and started to search memory for a nibble search routine. After a little searching, I found it in memory around location $1000. I wrote down a sequence of bytes and searched the disk for them, but I was not able to find them on either side of the disk. I came to the conclusion that the program was stored on the disk in a coded form. After a lot of hard work and countless hours of searching, I found that the coded program lay on track $18, sectors $01-$05, $0F on the front and also another on track $05, sectors $09-0D on the back. Both of these programs are nibble search routines. Our objective is to defeat these nibble search routines.

v2.0

The Procedure

After examining the program, I found that when it wanted to perform a nibble search, it would jump to location $0F40. Fortunately when I searched the disk for this routine, it was not coded and was found on track $13, sector $07. To disable the routine, I put a ReTurn from Subroutine (RTS, or $60) in byte $00 of track $13, sector $07. This defeated the nibble search routine and let me play the game except I could not LIFTOFF. Every time I tried to liftoff, it would tell me:

*** Lift clearance denied ***
*** due to pirate activity ***

After some more searching, I found that a successful nibble search would store the letters SHIP in locations $1801-$1804. So to defeat the nibble search routine, we make it store the letters SHIP in locations $1801-$1804 and return to the rest of the game.

The Controller

The controller is set up to copy both sides of the disk and asks you which side you wish to copy. While copying the front side of the disk, it will copy tracks $00-$22 and sector edit track $13, sector $07 starting at byte $00, changing from:

```
68 85 00 68 85 01 68 85
02 68 85 03 A5 01 48 A5
00 48 A0 00 B1
```
to:
```
A9 53 8D 01 18 A9 48 8D
02 18 A9 49 8D 03 18 A9
50 8D 04 18 60
```

While copying the back side, it will copy tracks $00-$21, ignoring the unreadable nibble search tracks.

Step by Step

1) Install the controller at the end of this article into Super IOB v1.5.

2) Copy both sides of the disk, specifying Front or Back when the program asks.

3) Enjoy your unprotected version of SunDog v2.0!

---

controller

```
1000 REM SUNDOG V2 CONTROLLER
1010 HOME : A$ = "FRONT^ OR^ BACK?" : GOSUB 450 :
     GET I$
1020 IF I$ < > "F" AND I$ < > "B" THEN 1010
1030 TK = 0 : LT = 35 : ST = 15 : LS = 15 : CD = WR : FAST
     = 1
1040 IF I$ = "B" THEN LT = 34
1050 GOSUB 490 : GOSUB 610
1060 IF I$ = "F" THEN T1 = TK : TK = PEEK (TRK) -
     1 : RESTORE : GOSUB 310 : TK = T1
1070 GOSAUB490 : GOSUB 610 : IF PEEK (TRK) = LT
     THEN 1090
1080 TK = PEEK (TRK) : ST = PEEK (SCT) : GOTO 1050
1090 HOME : PRINT "COPY^ DONE" : END
5000 DATA 21^ CHANGES
5010 DATA 19 ,7 ,0 ,169 ,19 ,7 ,1 ,83 ,19 ,7 ,2
     ,141
5020 DATA 19 ,7 ,3 ,1 ,19 ,7 ,4 ,24 ,19 ,7 ,5 ,169
5030 DATA 19 ,7 ,6 ,72 ,19 ,7 ,7 ,141 ,19 ,7 ,8 ,2
5040 DATA 19 ,7 ,9 ,24 ,19 ,7 ,10 ,169 ,19 ,7 ,11
     ,73
5050 DATA 19 ,7 ,12 ,141 ,19 ,7 ,13 ,3 ,19 ,7 ,14
     ,24
5060 DATA 19 ,7 ,15 ,169 ,19 ,7 ,16 ,80 ,19 ,7 ,17
     ,141
5070 DATA 19 ,7 ,18 ,4 ,19 ,7 ,19 ,24 ,19 ,7 ,20
     ,96
```

controller checksums

1000	– $356B	1090	– $FC0C
1010	– $107A	5000	– $25D2
1020	– $B7E1	5010	– $8051
1030	– $E091	5020	– $C120
1040	– $CABC	5030	– $5754
1050	– $6D26	5040	– $EB5A
1060	– $3586	5050	– $A3F8
1070	– $44A2	5060	– $0049
1080	– $7E43	5070	– $D27D
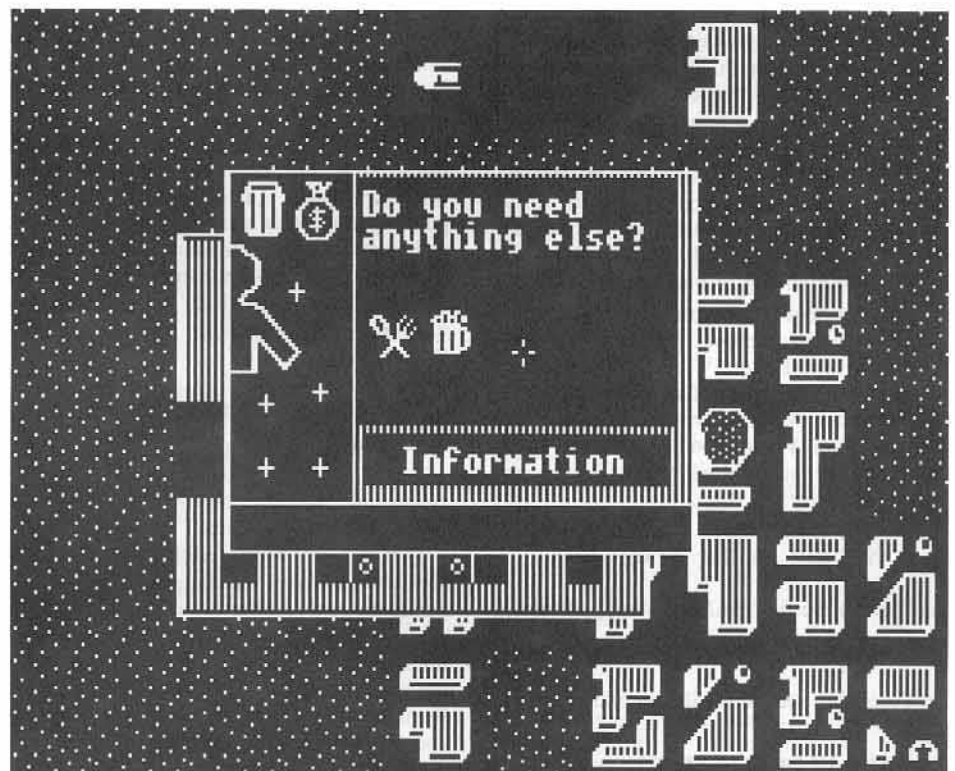
Ultima IV

By Jim S. Hart
and Ray Darrah

Requirements:
DOS 3.3
One disk drive

The new release in the Ultima series is finally out and it is an excellent game. I have had the chance to play it and would definitely recommend it to anyone who enjoys D&D style games. The object of the game is to become enlightened in the eight virtues of the Avatar in your quest to destroy the Abyss of Evil. Along the way you pick up companions, solve quests, explore deep dungeons, and generally learn of the virtues needed to win the game.

There are times, however, when going through an elaborate sequence of moves to get somewhere may not be your cup of tea. Maybe it's just that you want to "customize" the world to fit your needs or possibly you want to see what the world map of Ultima IV really looks like (without having to map it by hand).

Fear not bold adventurers for here are two programs to help you out. The first program, "Ultimainland Editor IV" allows the editing of the entire world map. The second program, "Ultimapper IV" will print out a map of the entire Ultima IV world on your printer. These programs are only limited by what you choose to use them for!

Keying in the Programs

1) Type in the first BASIC listing and save it

SAVE ULTIMAINLAND EDITOR IV

2) Type in the second BASIC program and save it

SAVE ULTIMAPPER IV

3) Key in the first hex code using the instructions on the inside front cover of this magazine and save it

BSAVE SECTOR.ZAP, A$300, L$21

4) Key in the second hex code and save it

BSAVE OBJ.IV,A$300,L$C6

5) Make sure to put these four programs on the same disk!

Using The Programs

To use the printer program, ready your printer with at least 16 sheets of paper, RUN the program and insert the Britannia disk when prompted. The program will then proceed to print out, in four sections, the entire Ultima IV world map. This process will take over one hour.

When the program is finished, tape the four sections together from left to right and viola`, there is your map!

The mainland editor is almost just as easy to use. First, RUN it and insert your Britannia disk when prompted. Table 1 shows all of the possible things that can be on the surface and if the editor supports the manipulation of that character. Note: some printers will not print the delete character (ASCII 127) so an ampersand was substituted in its place in the Ultimapper IV program.

Movement and Editing

To move your cursor about in the editing window, type the usual "I,J,K,M" escape characters. Note that on //e's the cursor will not flash when atop lava.

Directly below the editing window, you should see a line of the characters that you can add to the map. Under one of these characters there should be a carat "^" which is highlighting the current character. By pressing return, the current character will be placed on the map where the cursor is. Pressing the left and right arrow keys will highlight a different character or you can press the key of the character you wish to highlight. For example, pressing the percent key will instantly highlight the percent character.

The Ultimainland Editor divides the map that the Ultimapper IV program makes into four quadrants. They are numbered as follows:

```
0 | 1
--+--
2 | 3
```

You may read from the disk or write to the disk one of these quadrants by pressing "R" for read and "W" for write.

Hopefully these two programs will make the Ultima IV game more enjoyable. For you hackers out there I suggest looking at the three unprotected disk sides with a program such as Copy][Plus and see what you can find. Believe me, there is a lot of information to look at, much of it NOT in file form, and decipher. Happy adventuring!!

Ultimainland Editor

```
10 REM * -------------------- *
20 REM *    ULTIMA IV          *
30 REM *     MAINLAND EDITOR   *
40 REM *   BY : RAY DARRAH     *
50 REM * -------------------- *
60 REM
70 REM SET UP CONSTANTS
80 HIMEM: 8191 : CLEAR
90 IF PEEK (768) < > 169 OR PEEK (769) < > 7 THEN
   PRINT CHR$ (4) "BLOAD^ OBJ.IV"
100 X = 5 : Y = 1 : B1 = 0 : B2 = 32 : TRK = 47084 : SCT
   = 47085 : TK = 0 : ST = 0 : CC = 1 : RW = 47092 : Q
   = 0
110 DIM C(27) : FOR A = 1 TO 27 : C$ = C$ + CHR$ (
   PEEK (938 + A) - 128) : C(A) = PEEK (911 +
   A) : NEXT
120 K$ = "IJKM" + CHR$ (13) + CHR$ (8) + CHR$ (21
   ) + CHR$ (27) + "RW"
130 REM SET UP SCREEN DISPLAY
140 TEXT : HOME : POKE 48 ,32 : VLIN 0 ,33 AT 3 :
   VLIN 0 ,33 AT 36 : INVERSE : HTAB 5 : VTAB 17
   : PRINT SPC( 32) : NORMAL
150 VTAB 19 : HTAB 8 : PRINT C$ : GOSUB 370 : VTAB
   20 : HTAB 5 : PRINT "" : HTAB 36 : PRINT ""
160 REM GET INTIAL QUADRANT
```

```
170 VTAB 23 : HTAB 1 : PRINT "INSERT^ BRITANNIA^
    DISK^ AND^ PRESS^ RETURN"
180 WAIT - 16384 ,128 : GET A$ : IF A$ < > CHR$ (13
    ) AND A$ < > CHR$ (27 ) THEN 180
190 IF A$ = CHR$ (27 ) THEN HOME : END
200 VTAB 23 : HTAB 1 : PRINT SPC( 39 )
210 REM READ IN NEW 16K AREA
220 POKE RW ,1
230 POKE TRK ,TK : POKE SCT .ST : POKE 47093 ,0
    : POKE 47083 ,0 : CALL 768
240 REM PUT COMMANDS ON SCREEN
250 VTAB 22 : HTAB 1 : CALL 64578 : PRINT
    "I,J,K,M^ =^ MOVEMENT" TAB( 23 ) "RETURN^
    =^ PUT^ CHAR"
260 VTAB 24 : PRINT "R^ =^ READ^ QUADRANT" TAB(
    22 ) "W^ =^ WRITE^ QUADRANT" ;
270 REM PRINT CURRENT MAP
280 POKE 252 ,B1 : POKE 253 ,B2 : CALL 833
290 REM WAIT FOR COMMAND
300 VTAB Y : HTAB X : GET A$
310 FOR A = 1 TO LEN (K$ ) : IF A$ < > MID$ (K$ ,A
    ,1 ) THEN NEXT : GOTO 340
320 ON A GOTO 390 ,490 ,530 ,440 ,570 ,600 ,620
    ,190 ,640 ,660
330 REM TEST FOR MAP CHAR
340 FOR A = 1 TO LEN (C$ ) : IF A$ < > MID$ (C$ ,A
    ,1 ) THEN NEXT : GOTO 300
350 CC = A : GOSUB 370 : GOTO 300
360 REM HIGHLIGHT CURRENT CHAR
370 VTAB 20 : HTAB 8 : PRINT SPC( 27 ) ; : HTAB CC
    + 7 : PRINT " ^" : RETURN
380 REM MOVE CURSOR UP
390 IF Y = 1 AND B1 = 0 AND B2 / 8 = INT (B2 / 8 )
    THEN 300
400 Y = Y - 1 : IF Y > 0 THEN 300
410 Y = 8 :B1 = B1 - 128 : IF B1 > - 1 THEN 280
420 B1 = 128 :B2 = B2 - 1 : GOTO 280
430 REM MOVE CURSOR DOWN
440 IF Y = 16 AND B1 = 0 AND (B2 - 7 ) / 8 = INT ((B2
    - 7 ) / 8 ) THEN 300
450 Y = Y + 1 : IF Y < 17 THEN 300
460 Y = 9 :B1 = B1 + 128 : IF B1 < 256 THEN 280
470 B1 = 0 :B2 = B2 + 1 : GOTO 280
480 REM MOVE CURSOR LEFT
490 IF X = 5 AND B2 < 40 THEN 300
500 X = X - 1 : IF X > 4 THEN 300
510 X = 20 :B2 = B2 - 8 : GOTO 280
520 REM MOVE CURSOR RIGHT
530 IF X = 36 AND B2 > 79 THEN 300
540 X = X + 1 : IF X < 37 THEN 300
550 X = 21 :B2 = B2 + 8 : GOTO 280
560 REM PUT CHAR ON MAP
570 PRINT MID$ (C$ ,CC ,1 ) ; : IF X > 20 THEN POKE
    B1 + B2 * 256 - X - (Y - 1 ) * 16 + 2027
    ,C(CC ) : GOTO 300
580 POKE B1 + B2 * 256 + X - 5 + (Y - 1 ) * 16
    ,C(CC ) : GOTO 300
590 REM MOVE POINTER LEFT
600 CC = CC - (CC > 1 ) : GOSUB 370 : GOTO 300
610 REM MOVE POINTER RIGHT
620 CC = CC + (CC < 27 ) : GOSUB 370 : GOTO 300
630 REM READ NEW QUADRANT
640 POKE RW ,1 :A$ = "READ" : GOTO 680
650 REM WRITE NEW QUADRANT
660 POKE RW ,2 :A$ = "WRITE"
670 REM GET NEW QUADRANT NUMBER
680 VTAB 22 : HTAB 1 : CALL 64578 : PRINT
    "QUADRANT^ TO^ " A$ "^ (0-3)^ "; CHR$ (8 ) ;
690 GET B$ : IF (B$ < "0" OR B$ > "3" ) AND B$ < >
    CHR$ (13 ) AND B$ < > CHR$ (27 ) THEN 690
700 IF A$ = CHR$ (27 ) THEN 250
710 A = VAL (B$ ) : IF B$ = CHR$ (13 ) THEN A = Q
```

Source Code for OBJ.IV

B7F5-	ERROR	.EQ $B7F5	DOS ERROR CODE
B7ED-	SECTOR	.EQ $B7ED	DOS SECTOR NUMBER
B7EC-	TRACK	.EQ $B7EC	DOS TRACK NUMBER
B7F1-	BUFMSB	.EQ $B7F1	WHERE THE SECTOR GOES OR COMES FROM
FDF0-	COUT	.EQ $FDF0	PRINT OUT ROUTINE IN MONITOR
20-	LFTMAR	.EQ $20	LEFT MARGIN OF TEXT WINDOW

Table 1:
Surface character representations

Byte	Item	edit?	Char.
$00	Deep water	Y	~
$01	Regular water	Y	space
$02	Shallow water	Y	-
$03	Swamp	Y	#
$04	Grass	Y	.
$05	Brush	Y	-
$06	Forest	Y	+
$07	Hills	Y	/
$08	Mountains	Y	*
$09	Dungeon	Y	D
$0A	City	Y	c
$0B	Castle	Y	C
$0C	Village	Y	V
$0D	L.Side of Main Castle	Y	[
$0E	Middle of Main Castle	Y	@
$0F	R.Side of Main Castle	Y	]
$10	Boat left	N	
$11	Boat up	N	
$12	Boat right	N	
$13	Boat down	N	
$14	Horse left	N	
$15	Horse right	N	
$16	Spider web	N	
$17	Bridge	Y	=
$18	Balloon	Y	%
$19	Top of bridge	N	
$1A	Bottom of bridge	N	
$1B	Up ladder	N	
$1C	Down ladder	N	
$1D	Magincia	Y	M
$1E	Shrine	Y	S
$1F	Ranger	N	
$20-21	Character 'A'	N	
$22-23	Character 'B'	N	
$24-25	Character 'C'	N	
$26-27	Character 'D'	N	
$28-29	Character 'E'	N	
$2A-2B	Character 'F'	N	
$2C-2D	Character 'G'	N	
$2E-2F	Character 'H'	N	
$30	Stone pillar	N	
$31	Bottom left diagonal	N	
$32	Bottom right diagonal	N	
$33	Upper left diagonal	N	
$34	Upper right diagonal	N	
$35	Ship mast	N	
$36	Ship cross	N	
$37	Rock	Y	:
$38	Sick character	N	
$39	Stone fence	N	
$3A	Locked door	N	
$3B	Open door	N	
$3C	Chest	Y	$
$3D	Ankh	N	
$3E	Brick floor	N	
$3F	Middle of bridge	N	
$40-43	Moongate	N	
$44	Green fire	Y	1
$45	Blue fire	Y	2
$46	Orange fire	Y	3
$47	Red fire	Y	4
$48	Solid wall	N	
$49	Secret brick wall	N	
$4A	Dungeon altar	N	
$4B	Campfire	N	
$4C	Lava	Y	delete
$4D	Dungeon eyeball	N	
$4E-4F	Attack shots	N	
$50-51	Guard	N	
$52-53	Merchant	N	
$54-55	Bard	N	
$56-57	Jester	N	
$58-59	Beggar	N	
$5A-5B	Child	N	
$5C-5D	Bull	N	
$5E-5F	Lord British	N	
$60-79	Alphabet A to Z	N	
$7A	Parallel lines	N	
$7B	Shape of ']'	N	
$7C	Shape of '['	N	
$7D	Window	N	
$7E	Dark space	N	
$7F	Brick wall	N	
$80-83	Pirate Ship	N	
$84-85	Nixie	N	
$86-87	Squid	N	
$88-89	Sea serpent	N	
$8A-8B	Seahorse	N	
$8C-8D	Whirlpool	N	
$8E-8F	Tornado	N	
$90-93	Rat	N	
$94-97	Bat	N	
$98-9B	Spider	N	
$9C-9F	Ghost	N	
$A0-A3	Slime	N	
$A4-A7	Troll	N	
$A8-AB	Gremlin	N	
$AC-AF	Treasure chest	N	
$B0-B3	Reaper	N	
$B4-B7	Insects	N	
$B8-BB	Eyeball	N	
$BC-BF	Phantom	N	
$C0-C3	Orc	N	
$C4-C7	Skeleton	N	
$C8-CB	Rogue	N	
$CC-CF	Snake	N	
$D0-D3	Ettin	N	
$D4-D7	Headless	N	
$D8-DB	Cyclops	N	
$DC-DF	Wisp	N	
$E0-E3	Mage	N	
$E4-E7	Skull	N	
$E8-EB	Lava lizard	N	
$EC-FF	Zorn	N	
$F0-F3	Daemon	N	
$F4-F7	Three headed Dragon	N	
$F8-FB	Dragon	N	
$FC-FF	Balron	N	

```
720 Q = A :TK = 8 * (A > 1) :ST = 8 * (A / 2 <
  > INT (A / 2) ) : HTAB 1 : CALL 64578
730 PRINT : PRINT LEFT$ (A$ ,4) "ING^QUADRANT^
  " A : GOTO 230
```

Ultimainland Editor checksums

10	- $BADD	380	- $9C32
20	- $9B13	390	- $A264
30	- $4D3B	400	- $19E7
40	- $AD92	410	- $443A
50	- $C899	420	- $B8E0
60	- $FF65	430	- $A84B
70	- $A3BF	440	- $04AF
80	- $646B	450	- $FFEE
90	- $EC83	460	- $03A3
100	- $5BF9	470	- $F612
110	- $A345	480	- $DA62
120	- $4D8B	490	- $2CC8
130	- $75F3	500	- $BA53
140	- $FE5A	510	- $73A9
150	- $E9D6	520	- $0222
160	- $F428	530	- $082C
170	- $F791	540	- $6EF5
180	- $7FAC	550	- $D999
190	- $FCCB	560	- $422C
200	- $8CFA	570	- $4ABD
210	- $885C	580	- $F137
220	- $8EA3	590	- $EC3B
230	- $8DC6	600	- $38DD
240	- $C474	610	- $564C
250	- $2516	620	- $DEB2
260	- $317B	630	- $E5CE
270	- $97C0	640	- $7E24
280	- $CE8F	650	- $DB44
290	- $FB65	660	- $300C
300	- $A1DD	670	- $7340
310	- $FEF7	680	- $A397
320	- $8580	690	- $9FDC
330	- $30F8	700	- $B187
340	- $12DA	710	- $E097
350	- $4A59	720	- $2838
360	- $7DEC	730	- $63F9
370	- $24CC		

Ultimapper IV

```
10 REM *--------------------*
20 REM *  ULTIMA IV         *
30 REM *   MAINLAND PRINTER *
35 REM *  BY JIM S. HART    *
40 REM *--------------------*
50 REM
60 REM *--------------------*
70 REM *  SET UP VARIABLES  *
80 REM *--------------------*
90 REM
100 HIMEM: 38144
110 IF PEEK (768) < > 169 OR PEEK (769) < > 3 THEN
    PRINT : PRINT CHR$ (4) ; "BLOAD^
    SECTOR.ZAP, A$300"
120 CLEAR : BUF = 38143 :RWTS = 768 :SEC = 0 : DIM
    LOC(16 ,256) ,V(27) ,V$(27)
130 HOME : D$ = CHR$ (4) :P0$ = D$ + "PR#0" ;P1$
    = D$ + "PR#1"
140 FOR I = 1 TO 27 : READ V(I) ,J :V$(I) = CHR$
    (J) : NEXT : IV$ = "ULTIMA^ IV"
150 REM
160 REM *--------------------*
170 REM * PRINT OUT TITLE SCRN*
180 REM *--------------------*
190 REM
```

```
*  ------------------------------------------------------- *
*               READ OR WRITE 16K OF DATA                  *
*  ------------------------------------------------------- *

0300: A9 07          LDA #7      READ OR WRITE 8 TRACKS
0302: 85 FE          STA CNTR1
0304: A9 58          LDA #$58    FIRST SECTOR GOES AT $5800
0306: 8D F1 B7       STA BUFMSB
0309: AD FD B7  RW1  LDA SECTOR
030C: 85 FF          STA CNTR2   SAVE ENDING SECTOR NUMBER
030E: 18             CLC
030F: 69 07          ADC #7      GO BACKWARDS FOR SPEED
0311: 8D ED B7       STA SECTOR
0314: A9 B7     RW2  LDA #$B7    READ OR WRITE A SECTOR
0316: A0 E8          LDY #$E8
0318: 20 D9 03       JSR $3D9
031B: AD ED B7       LDA SECTOR  DONE?
031E: C5 FF          CMP CNTR2
0320: F0 0E          BEQ NXTTRK  YES, DO NEXT TRACK
0322: CE ED B7       DEC SECTOR  GET NEXT SECTOR
0325: AD F1 B7       LDA BUFMSB  NEXT SECTOR IS STORED 8 PAGES LESS
0328: 38             SEC
0329: E9 08          SBC #8
032B: 8D F1 B7       STA BUFMSB
032E: B0 E4          BCS RW2     .ALWAYS

0330: EE EC B7  NXTTRK INC TRACK NEXT TRACK NUMBER
0333: AD F1 B7       LDA BUFMSB
0336: 18             CLC
0337: 69 39          ADC #$39    NEXT SECTOR IS HERE
0339: 8D F1 B7       STA BUFMSB
033C: C6 FE          DEC CNTR1   DONE WITH ALL THE TRACKS?
033E: 10 C9          BPL RW1     NOPE, CONTINUE
0340: 60             RTS

*  ------------------------------------------------------- *
*             PRINT CURRENT STUFF ON SCREEN                *
*  ------------------------------------------------------- *

0341: A9 04          LDA #4      CHANGE TEXT WINDOW
0343: 85 20          STA LFTMAR
0345: A9 FF   PRINT3 LDA #$FF    START AT TOP OF SCREEN
0347: 85 25          STA CV
0349: A0 00          LDY #0      START WITH FIRST BYTE
034B: A9 0F          LDA #15     SIXTEEN LINES/SECTOR
034D: 85 FF          STA CNTR2   DO 256 BYTES
034F: A9 8D   PRINT1 LDA #$8D    GO TO LEFT EDGE OF NEXT LINE
```

```
200 HOME : INVERSE
210 VTAB 10 : PRINT "^ ULTIMA^ IV^ MAINLAND^
    PRINTER^ "
220 NORMAL : VTAB 20
230 PRINT "Insert^ Britannia^ &^ Press^
    [RETURN]"
240 PRINT "("; SPC( 9); "[ESC]^to^exit" ; SPC(
    9) , ")^ " ;
250 GET A$ : ON (A$ < > CHR$ (13) AND A$ < > CHR$
    (27) ) GOTO 250
260 PRINT A$ : IF A$ = CHR$ (27) THEN TEXT : HOME
    : END
270 HOME : VTAB 10 : INVERSE
280 PRINT "^ THIS^ WILL^ TAKE^ ABOUT^ 1^ HOUR. ^ "
290 NORMAL
300 REM
```

```
310 REM *--------------------*
320 REM * TURN ON PRINTER AND *
330 REM * PRINT OUT MAINLAND  *
340 REM *--------------------*
350 REM
360 PRINT P1$ : PRINT CHR$ (9); "80N"
370 PRINT IV$
380 POKE 790 ,1 : POKE 791 ,0
390 FOR LOOP = 0 TO 15
400 FOR S1 = SEC TO (SEC + 3)
410 POKE 782 ,LOOP : POKE 783 ,S1 : POKE 791 ,0
    : CALL RWTS : POKE 72 ,0
420 FOR J = 1 TO 256 :LOC(S1 ,J) = PEEK (BUF + J
    ) : NEXT J
430 NEXT S1
440 FOR I = 0 TO 15
```

```
0351: 20 F0 FD          JSR COUT
0354: A9 0F             LDA #15        SIXTEEN BYTES/LINE
0356: 85 FE             STA CNTR1
0358: B1 FC     PRINT2  LDA (SECT1),Y  GET A BYTE FROM FIRST SECTOR
035A: A2 1A             LDX #26        27 ITEMS TO COMPARE WITH
035C: DD 90 03  CMP1    CMP ITEMS,X    MATCH?
035F: F0 07             BEQ GOT.IT     YES. TRANSLATE AND PRINT
0361: CA                DEX            NEXT ITEM
0362: 10 F8             BPL CMP1
0364: A9 BF             LDA #$BF       ANYTHING ELSE IS A QUESTION MARK
0366: D0 03             BNE SKIP1      SKIP THE LDA BELOW
0368: BD AB 03  GOT.IT  LDA TRANSL,X   TRANSLATE INTO PRINTABLE CHARACTER
036B: 20 F0 FD  SKIP1   JSR COUT       PRINT IT
036E: C8                INY            NEXT BYTE
036F: C6 FE             DEC CNTR1      DONE WITH LINE?
0371: 10 E5             BPL PRINT2     NOPE CONTINUE
0373: C6 FF             DEC CNTR2      DONE WITH SECTOR
0375: 10 D8             BPL PRINT1     NOPE. DO NEXT LINE

0377: A5 20             LDA LFTMAR     ALREADY PRINTED SECOND SECTOR?
0379: C9 14             CMP #20
037B: F0 0E             BEQ EXIT       YES. WERE DONE
037D: A9 14             LDA #20
037F: 85 20             STA LFTMAR
0381: A5 FD             LDA SECT1+1    NEXT SECTOR TO PRINT IS +8
0383: 18                CLC
0384: 69 08             ADC #8
0386: 85 FD             STA SECT1+1
0388: 4C 45 03          JMP PRINT3

038B: A9 00     EXIT    LDA #0         FIX TEXT WINDOW
038D: 85 20             STA LFTMAR
038F: 60                RTS

0390: 00 01 02
0393: 03 04 05
0396: 06 07 08
0399: 09 0A 0B
039C: 0C 0D     ITEMS   .HS 00.01.02.03.04.05.06.07.08.09.0A.0B.0C.0D
039E: 0E 0F 17
03A1: 18 1D 1E
03A4: 37 3C 44
03A7: 45 46 47
03AA: 4C                .HS 0E.0F.17.18.1D.1E.37.3C.44.45.46.47.4C
03AB: FE A0 DE  TRANSL  .HS FE.A0.DE
03AE: A3 AE AD
03B1: AB AF AA
03B4: C4 E3 C3
03B7: D6 DB C0
03BA: DD BD A5
03BD: CD D3 BA
03C0: A4 B1 B2
03C3: B3 B4             .AS -"#.-+/*DcCV[@]=1MS.$1234"
03C5: FF                .HS FF
```

```
450 FOR S1 = SEC TO (SEC + 3 )
460 FOR J = 1 TO 16
470 FOR K = 1 TO 27
480 IF LOC(S1 ,J+(16 * I ) ) = V(K) THEN PRINT
V$(K); .K = 28 : NEXT K : GOTO 500
490 NEXT K ; PRINT "?" ;
500 NEXT J
510 NEXT S1 : PRINT "^ "
520 NEXT I
530 NEXT LOOP
540 IF SEC<> 12 THEN PRINT CHR$ (12 ) . SEC = SEC
+ 4 : GOTO 370
550 TEXT . HOME . PRINT P0$ : END
560 REM
```

```
570 REM *---------        --------*
580 REM *  VALUE OF ITEM &       *
585 REM *  CHR REPRESENTATION  *
590 REM *---------------------*
600 REM
610 DATA 0 ,126 ,1 ,32 ,2 ,94 ,3 ,35 ,4 ,46 ,5 ,45
,6 ,43 ,7 ,47 ,8 ,42
620 DATA 9 ,68 ,10 ,99 ,11 ,67 ,12 ,86 ,13 ,91 ,14
,64 ,15 ,93
630 DATA 23 ,61 ,24 ,37 ,29 ,77 ,30 ,83 ,55 ,58
,60 ,36 ,76 ,38
640 DATA 68 ,53 ,69 ,54 ,70 ,55 ,71 ,56
```

Ultimapper IV checksums

10	- $BADD	330	- $458B
20	- $9B13	340	- $6F1D
30	- $4D3B	350	- $D9A2
35	- $8081	360	- $AE07
40	- $971F	370	- $23B7
50	- $A151	380	- $6CC2
60	- $5F38	390	- $74D2
70	- $8A25	400	- $5A3D
80	- $13B9	410	- $D956
90	- $1F6D	420	- $B4AA
100	- $36BB	430	- $BA04
110	- $9A92	440	- $BDA9
120	- $1CC1	450	- $E1DD
130	- $8B6F	460	- $2E2B
140	- $C0FD	470	- $D85F
150	- $E384	480	- $D06E
160	- $BEAE	490	- $ADC6
170	- $FCC2	500	- $C1C3
180	- $4863	510	- $7244
190	- $E554	520	- $EA47
200	- $CC25	530	- $8EE4
210	- $0B40	540	- $FFC1
220	- $F5BC	550	- $57DA
230	- $F844	560	- $7D1C
240	- $5CC4	570	- $DB03
250	- $750F	580	- $70C0
260	- $8AD3	585	- $D25D
270	- $0082	590	- $3640
280	- $C77E	600	- $2729
290	- $A93D	610	- $DF2B
300	- $81F1	620	- $0E73
310	- $05E8	630	- $1C9F
320	- $D0B9	640	- $A17F

Sector.Zap Hexdump

```
0300. A9 03 A0 0A 20 D9 03 60    $6D5D
0308: 00 00 01 60 01 00 00 00    $37AD
0310: 1B 03 00 95 00 00 01 00    $08D4
0318: 00 60 01 00 01 FF D8 00    $C3C5
0320: 00                         $4343
```

OBJ.IV Hexdump

```
0300: A9 07 85 FE A9 58 8D F1    $9A1D
0308: B7 AD ED B7 85 FF 18 69    $7DC6
0310: 07 8D ED B7 A9 B7 A0 E8    $D101
0318: 20 D9 03 AD ED B7 C5 FF    $4634
0320: F0 0E CE ED B7 AD F1 B7    $2750
0328: 38 E9 08 8D F1 B7 B0 E4    $164A
0330: EE EC B7 AD F1 B7 18 69    $D905
0338: 39 8D F1 B7 C6 FE 10 C9    $5BC5
0340: 60 A9 04 85 20 A9 FF 85    $487E
0348: 25 A0 00 A9 0F 85 FF A9    $998D

0350. 8D 20 F0 FD A9 0F 85 FE    $F904
0358: B1 FC A2 1A DD 90 03 F0    $4386
0360: 07 CA 10 F8 A9 BF D0 03    $CA4B
0368: BD AB 03 20 F0 FD C8 C6    $8985
0370: FE 10 E5 C6 FF 10 D8 A5    $822C
0378: 20 C9 14 F0 0F A9 14 85    $D0CE
0380: 20 A5 FD 18 69 08 85 FD    $0CE1
0388: 4C 45 03 A9 00 85 20 60    $B032
0390: 00 01 02 03 04 05 06 07    $B092
0398: 08 09 0A 0B 0C 0D 0E 0F    $5092

03A0: 17 18 1D 1E 37 3C 44 45    $C9F8
03A8: 46 47 4C FE A0 DE A3 AE    $12D1
03B0: AD AB AF AA C4 E3 C3 D6    $6CB2
03B8: DB C0 DD BD A5 CD D3 BA    $E683
03C0: A4 B1 B2 B3 B4 FF          $1A5B
```

How to be the...

Sound Master

by William Wingfield Jr.

Requirements:
Low-wattage soldering iron
Small gauge hook-up wire
Audio Jack (RCA or 1/8" type)
500k potentiometer and 10 ohm resistor
Phillips Head screwdriver
Impedance transformer (optional)

> *Note: The procedure described below requires modification of your computer and may void any warranty. COMPUTIST will not be held responsible for any damages incurred while following this procedure.*
>
> This procedure involves taking apart your computer. If you are new at this, you should read the entire article carefully before doing anything and then read it a second time while following the procedure.

Have you ever noticed just how loud that little speaker that's inside your Apple can be when you're playing your favorite game late at night when there are people trying to sleep? To remedy the situation, you hit ⌂S and when this works, you find playing without sound to just not be the same.

Perhaps you have a music program or two and would like to hear your creations on something besides that little speaker in your Apple. Or maybe you've decided connect your Apple to your stereo and then someone mentions that due to the way the Apple drives its speaker you may blow the inputs on your stereo. What do you do?

This article will tell you how to add a volume control, external speaker or stereo-level output to your Apple at very little cost. You'll be able to play your favorite games with sound at night that only you can hear or blast it out your stereo so that the neighbors down the street can hear.

Adding the External Speaker

As it comes, the Apple simply has a speaker hooked to the motherboard with a two conductor connector. To add an external speaker all that's needed is a jack to plug it into. I recommend the closed-circuit type (the type that are used for earphone jacks on radios). Closed-circuit earphone jacks automatically turn off the computer's speaker when the external speaker is plugged in. This way the internal speaker is still there and ready if you disconnect the external one.

About the Volume Control

A 500 ohm potentiometer (pot) and 10 ohm resistor are all that are needed to make a good volume control. The resistor provides a load for the sound output from the Apple when the volume is turned completely down (this prevents shorting which could damage your Apple). Although audiophiles would probably reccomend an audio taper pot because it is more adjusted to your hearing, it really doesn't matter whether you use a linear or audio taper pot. A 1000 ohm pot may be used also.

Getting Your Feet Wet

If you intend to mount the volume control or speaker jack I recommend removing the body shell from the chassis first. Remove the power cord, open your case and remove everything that's not your Apple (cards, game controllers, etc). Turn the computer over and remove the screws that attach the body shell to the chassis from the bottom of the computer, refer to Fig. 1. Take care not to remove any other screws such as those that hold the power supply or the motherboard in place.

Holding the body shell to the bottom, turn the computer back upright and lift the shell up

a few inches. Look under the keyboard and you will see a jumper connecting the keyboard to the motherboard and a small two-pin connector that attaches the speaker to the motherboard. Unplug the jumper and the speaker connector and remove the shell. If you haven't cleaned the inevitable dust off of your motherboard for a while, this is a perfect opportunity to do so.

You should now decide where you want the external speaker jack and volume control located. I put my jack in the back of the computer out of the way and the volume control to the top left of the keyboard. If you decide to mount the jack in the case of your computer it may be necessary to counter-sink the mounting hole due to the thickness of the case. If you are going to mount the jack or volume control, do so now. Of course you can add just the jack or just the volume control by skipping the connections to the one you omit.

Solder Time

Solder the 10 ohm resistor to the CCW terminal, or the low volume side of the pot. Next remove the leads from the speaker in the computer and solder one to the free end of the 10 ohm resistor and the other to the wiper terminal of the pot. This should be the center terminal if they aren't marked on your pot. Refer to fig. 2 for a schematic or fig. 3 for a wiring diagram.

Now cut two lenghts of wire long enought to reach from the pot to the output jack. Solder one of these to the 10 ohm resistor along with the lead that came off of the speaker. Solder the other to the remaining terminal of the pot, the CW or high volume side.

Now look closely at the audio jack, you should see three terminals. Two of the leads normally make a connection through the jack so that the internal speaker will get current but when a plug is inserted the circuit is opened and the internal speaker is cut off. We'll call the

one that makes contact with the tip of the plug terminal #1. The one that becomes disconnected is terminal #2. The last of these is the connection to the ring part of the jack, this is the ground connection, we will call this terminal #3.

Solder the lead from the CW terminal of the pot to the #1 terminal of the jack. Attach the lead from the resistor to the #3 terminal of the jack, don't solder it yet.

Now cut two more lengths of wire long enough to reach from the jack to the speaker. Solder one to terminal #2 of the jack (the one not yet used) and the other to terminal #3 along with the wire from the resistor.

Now check over all of your work, reassemble your computer and get ready to test out the modification. Turn the volume control to the middle of its rotation and turn on your computer. It should beep as usual, just not as loud. Turn the volume up and hit reset, the beep should be at normal volume. Turn the volume down and hit reset again, the sound should be completely off. If the volume control works backwards then you have the CW and CCW leads reversed on the pot.

Now plug in the external speaker. It should now work just as the internal speaker did and the internal speaker should be off. If the internal speaker didn't work but the external one does then you probably have leads #1 and #2 reversed on the jack.

Going All the Way

If you would like to keep the case so that it is easily removable from the chassis you should use a connector between the output jack and the speaker rather than soldering directly to the speaker terminals on the motherboard. Any two-conductor connector will do for this.

If you want to hook your Apple audio into your stereo system you'll need to isolate the output jack from the Apple with a transformer. An 8 ohm to 8 ohm isolation transformer installed as shown in fig. 2 will allow the use of an external speaker or the option of plugging into the auxiliary input of your stereo.

If you want a dedicated output for your stereo an 8 ohm to 2000 ohm impedance transformer can be used. Connect the leads from the pot to the 8 ohm side and the 2000 ohm side to the output jack. Note that the internal speaker isn't used in this configuration. You may want to use an RCA type connector for an output dedicated to your stereo since this is the standard stereo type connector. The volume control isn't needed in this configuration but it will allow you to adjust the volume being put out by your stereo from your Apple.

A third option is to use a 2000 ohm to 2000 ohm isolation transformer in parallel with the speaker, see fig. 4. The speaker in the Apple and the high-impedance output will always be active.

Which ever way you decide to configure your sound I'm sure you'll find it to be an improvement over the simple speaker that your computer came with. Now you're ready to blast away with Skyfox even if you don't have a Mockingboard.
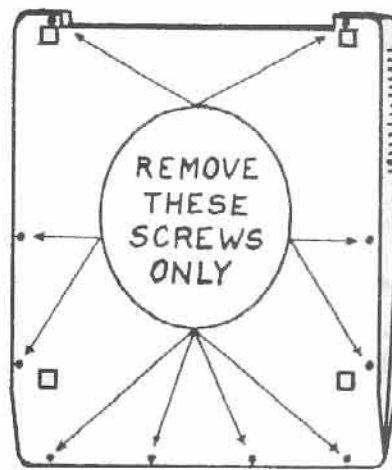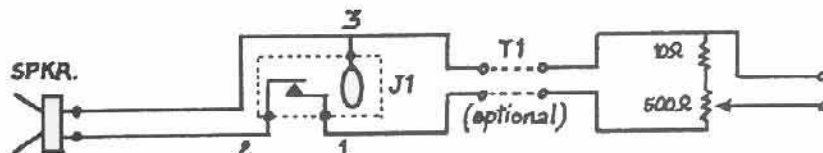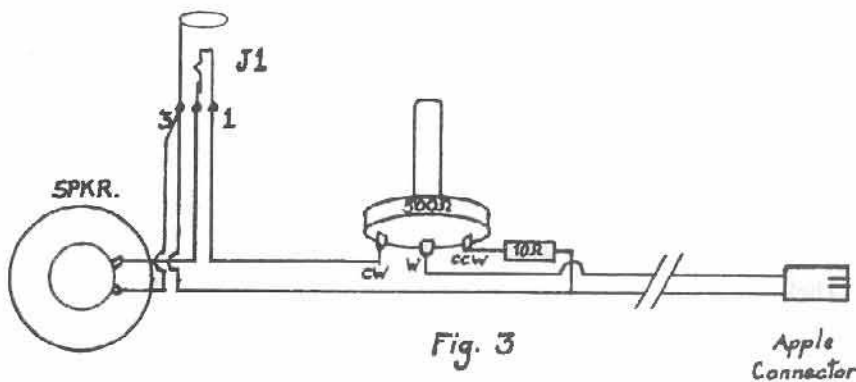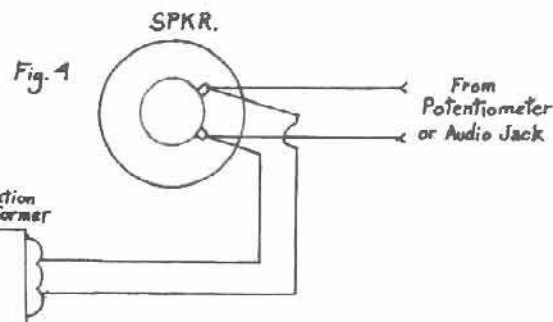

Fig. 1

Fig. 2

Fig. 3

Apple Connector

Fig. 4

From Potentiometer or Audio Jack

Isolation Transformer

G.I. Joe &
Lucas Film's Eidolon

by Larry Rando

Epyx
1043 Kiel Court
Sunnyvale, CA 94089

Requirements:
COPYA from the DOS 3.3 master
Sector editor
Disk searcher (optional)

G.I. Joe

Like the typical Epyx protection scheme, you have to first disable the DOS error checking routine in order to get the disk into a COPYA format. When I booted the disk I noticed it rebooted after sitting there a few seconds, just like the Winter Games protection (COMPUTIST No. 29). I then got out COPY II Plus and went into its sector editor in which I used the search function to search for $4C 00 C6 (JMP $C600, the boot ROM), which I found on track 0, sector 1. I temporarily replaced JMP $C600 with a JMP $FF59 (jump to monitor). I booted the disk, got into the monitor and searched the boot code before the JMP $C600 (where I had a hunch the protection was). After examining the boot code further, I noticed a bunch of branches going to the same location in memory. The code all these branches went to was indeed the main part of the protection.

Snooping through the boot code I found the main loader of G.I. Joe, in which I would have to insert a little routine to disable the protection check where it would be appropriate, but not interfere with the boot process.

To sum it all up I NOPed out anything that went to $992 then I inserted my routine that loaded in the rest of the program. I replaced a BNE $918 that was of no use to the program/boot, but part of the protection, with a JMP $D476.

1) Boot normal DOS 3.3

2) Enter the monitor and disable DOS's read error checking routine.

```
CALL-151
B942:18
```

3) RUN COPYA. Copy both sides.

4) Get out your sector editor and make the following changes to the boot side.

TRACK	SECTOR	BYTE	FROM	TO
$00	$01	$1A	$F0	$EA
		$1B	$76	$EA
		$1F	$B0	$EA
		$20	$71	$EA
		$25	$D0	$4C
		$26	$F1	$76
		$27	$A0	$D4
		$2F	$F0	$EA
		$30	$61	$EA
		$3D	$F0	$EA
		$3F	$53	$EA
		$4A	$D0	$EA
		$4B	$46	$EA
		$53	$D0	$EA
		$54	$3D	$EA
		$62	$F0	$EA
		$63	$2E	$EA
		$71	$D0	$EA
		$72	$1F	$EA
		$96	$4C	$EA
		$97	$00	$EA
		$98	$C6	$EA

5) Write the sector back out.

Lucas Film's Eidolon

To deprotect this one I used the same procedure as in G.I. Joe to find the protection. I searched for $4C 00 C6 and found it on track $1, sector $F so by snooping through the code that was executed before the JMP $C600, I found the routine (nibble count, in this case) that went to the JMP $C600 after it was executed. So, by NOPing out the nibble count it went on to work perfectly.

1) Boot normal DOS 3.3

2) Enter the monitor and disable DOS's read error checking routine.

```
CALL-151
B942:18
```

3) RUN COPYA and copy the disk.

4) Get out your sector editor and make the following changes.

TRACK	SECTOR	BYTE	FROM	TO
$01	$0F	$00	$20	$EA
		$01	$FD	$EA
		$02	$0E	$EA
		$05	$4C	$EA
		$06	$00	$EA
		$07	$C6	$EA

5) Write the sector back out to the disk.

Some Tips on EPYX

I will now give you some general tips for deprotecting software made by Epyx and Lucas Film. The first thing you must do is to disable DOS's read error checking routine as I did in these cases. Boot the disk up and see how far it gets. If you are lucky that is all you need for deprotecting whatever Epyx disk you tried it on. In case it does not work, see if the disk reboots at some point. If so, then use a disk search utility and search for $4C 00 C6.

You should find it somewhere on Track $00. When you locate it, try NOPing it out with three EA's and boot it up. Your computer should either just sit there or run the program. In the first case where it just sits there, try snooping through the code before the JMP $C600. You are likely to find the nibble count routine close by, which can be anything that will jump to the JMP $C600 or do something of that sort.

If you find something you believe is a nibble count (performs disk reads in the form LDA $C08C,X) try NOPing it out and boot up. If that does not work try setting up a boot trace and look for the program loader or start address, and when you find it replace the nibble count routine with a JSR to the program start address.

With the methods described above you can easily deprotect the following programs by Epyx: Worlds Greatest Baseball, Robots of Dawn, Temple of Apshai Trilogy, Lucas Film's Koronis Rift and Rescue on Fractalus. I have not tried any of these methods on any other Epyx programs that were not mentioned here, but they should be similar so give it a shot. It will also be wise to try these methods on any future programs to come from Epyx.

With some practice and becoming familar with the Epyx boot code you should become an expert in deprotecting Epyx software in no time. Don't forget to send your softkeys to COMPUTIST!

by Matt Evans

Epyx
1043 Kiel Ct.
Sunnyvale, CA 94089

Requirements:
64K Apple][
COPYA
A sector editor
A blank disk (2 sides)

As everyone knows, a good game disk is used many times. That's why I decided to dig in and find a softkey for Summer Games II. What follows is a few crude steps in boot code tracing. I started by booting DOS 3.3 and entered the following to make DOS ignore checksum errors, and copied the disk with COPYA.

```
CALL -151
B942:18
3D0G
RUN COPYA
```

would let the boot go the next step so I could examine the next stage at $1500. At $96F8, I entered:

```
96F8:A9 4C 8D 39 08 A9 59 8D
9670:3A 08 A9 FF 8D 3B 08 4C
9708:01 08
```

Which disassembles to

```
96F8- A9 4C     LDA #$4C
96FA- 8D 39 08  STA $0839
96FD- A9 59     LDA #$59
96FF- 8D 3A 08  STA $083A
9702- A9 FF     LDA #$FF
9704- 8D 3B 08  STA $083B
9707- 4C 01 08  JMP $0801
```

This puts a break point (JMP $FF59) at the JMP $1500 in page $8 where sector 0 got loaded, and continues the boot. Now, after the $1500 part gets loaded in (which happens to be the rest of track 0), I get the cursor back. I turned off the drive again.

I then looked at the next stage of the boot at $1500. Nothing strange happens early until you get to $1531. There, a routine begins to read bytes off the disk and check to see if they are of a certain value and then simply returns. Hmmm... I thought this was strange and

I booted the disk again by typing 9600G and stopped the drive. After sector 0 gets read in, my routine puts a JMP $970A at $0839. The boot continues at $801 where the rest of track 0 is read into memory. A jump occurs at $0839 to my routine at $970A. This puts a JMP $1580 at $1531 to skip the strange routine, and continues booting at $1500. The boot proceeds and finishes. Lo and behold, the disk booted with no problems save for the minor JMP at $1531 I put in. Now remember I am using the COPYA version so I know the COPYA version will boot.

Now all I had to do was find out where on the disk $1531 was located so I could insert my new jump. Since I knew it was on Track 0 somewhere, I simply wrote down the first few bytes at $1500 and began searching for those bytes by reading in each sector of track 0 and looking for them. I didn't have to look far, the $1500 routine was on sector 1 which meant that the my new jump would be placed at +$31 bytes into the sector. So I changed the A0 00 BD (LDY $BD00) on the disk to 4C 80 15 (JMP $1580) which would jump over the strange routine and wrote the sector back to the disk, and there I had it! I then tested it to see if all the events and things would work, and they did!

softkey for...

Summer Games II

Well, the disk copied fine, but it wouldn't boot, or at least I didn't hear any noises letting me know the head moved off track 0. Therefore, it appeared to have a normal DOS format, but something was going on during the first few seconds of the boot before much was done.

I decided to boot code trace since it appeared that I wouldn't have to go far. I copied the boot ROM that actually boots the disk, from $C600 to $9600 by entering:

```
CALL -151
9600<C600.C6FFM
```

I then changed the JMP $801 at $96F8, which usually continues the boot process at $801, to a JMP $FF59 by entering:

```
96F8:4C 59 FF
```

I then booted the disk:

```
9600G
```

When the cursor came back, I stopped the drive:

```
C0E8
```

I now had track 0 sector 0 in memory. I started examining the code and the first JMP I saw was a jump to $1500. So, I decided I

seemingly of no value to the boot. I found the end of the strange routine at $157F. I decided to modify my little boot routine so it would modify this part of the boot and force it to skip over this strange code. Before $1500 executed, I stored a JMP $1580 at $1531 so that it would skip the strange code. I entered:

```
96F8:A9 4C 8D 39 08 A9 0A 8D
9700:3A 08 A9 97 8D 3B 08 4C
9708:01 08 A9 4C 8D 31 15 A9
9710:80 8D 32 15 A9 15 8D 33
9718:15 4C 00 15
```

Disassembled:

```
96F8- A9 4C     LDA #$4C
96FA- 8D 39 08  STA $0839
96FD- A9 0A     LDA #$0A
96FF- 8D 3A 08  STA $083A
9702- A9 97     LDA #$97
9704- 8D 3B 08  STA $083B
9707- 4C 01 08  JMP $0801
970A- A9 4C     LDA #$4C
970C- 8D 31 15  STA $1531
970F- A9 80     LDA #$80
9711- 8D 32 15  STA $1532
9714- A9 15     LDA #$15
9716- 8D 33 15  STA $1533
9719- 4C 00 15  JMP $1500
```

The Procedure

1) Boot normal DOS 3.3.

2) Make DOS's checksum routine ignore errors.

```
CALL -151
B942:18
3D0G
```

3) RUN COPYA and copy both sides of the disk.

4) Sector edit track 0, sector 1 and change these bytes:

Track	Sector	Byte	From	To:
0	1	$31	$A0	$4C
		$32	$00	$80
		$33	$BD	$15

You now have a COPYA version of Summer Games II!

By the way, the high scores are kept on track 0, sector 2 if you want to change those to your advantage...

softkey for...

Thief

by Paul Wilson

Datamost

Requirements:
A way to Reset into the monitor
A blank disk
DOS 3.3

Thief, a game written by Bob Flanagan and marketed by Datamost in 1981, is based on the popular arcade game Berzerk. It is marketed as having seven levels, and a man to rescue on the seventh level. However, I have several reasons to believe that there are only four levels. Writing to Bob Flanagan about levels 5-7 didn't get a reply, so I decided to explore the code to check out the possibility of their non-existence. Alas, Thief is heavily protected. To remove the anti-user protection and to have a backup against the original disk wearing out or getting damaged are the two goals.

First, initialize a blank disk with DOS 3.3 and the following short HELLO program. Try some volume number other than 254. Too many disks use that number.

```
10  TEXT: HOME
20  PRINT " D BLOAD ROBOT,A$2000"
30  POKE -16304,0: POKE -16302,0
40  POKE -16300,0: POKE -16297,0
50  PRINT " D EXEC MZ.EXEC"
```

This is in fact, the HELLO file on the Thief disk. Replace the copy disk with the original disk. Boot the disk with PR#6. As the disk boots, wait until the screen goes black right after

the Applesoft prompt appears before pressing Reset. Do not try to engage DOS yet. Turn off the RUN Flag with (while in the monitor)

D6:0

This will keep the program from restarting when you go into BASIC. Now you can enter BASIC and list the HELLO program if you like.

Back in the monitor, the I/O routines of DOS have been sabotaged by patching a JMP $BF00 at $A676. The reset vector is set to $BF08. This means that a Reset in Applesoft or even an attempt to use DOS will access a memory wipe routine that will flush memory. The following will prevent any trouble after this point:

A676:68 38 60
BF08:60

The I/O routines *should* work now, but they don't. Attempts to access the disk result in groans and useless I/O ERROR messages. Why? The reason turned out to be simple, but clever. Some of the byte values of the prologues and epilogues of the address and data fields were stored in a volatile manner: in bytes $31 and $48. Commands like CMP #$31 and CMP #$48 accessed them. The normal user operations, even POKEing these bytes, will overwrite and destroy this data irretrievably, and the DOS will not work again.

Fortunately I was able to read track 0 to Thief with the ZAP utility of "Bag of Tricks". Sector 0 was in 16 sector format and the rest of track 0 was in 13 sector form, save for a few bizarre sectors that gave me "DAMAGED" or "UPDATED" messages. In sector 1 of track 0 I found code that stored $AE at $31 and $DE at $48. I was fortunate that this clue was so readily obtained, as Bag of Tricks is biased towards the normal disk and TRAX refused to examine any track in Thief other than track 0.

"UNABLE TO INTERPRET DATA" messages didn't help in trying to check for any other formatting tricks. This is a serious deficiency in Bag of Tricks' performance, but fortunately, further DOS format research proved unnecessary. It proved necessary to write a short driver that was totally portable to run the CATALOG function from the strange DOS.

```
300- A9 AE      LDA #$DE
302- 85 31      STA $31
304- A9 DE      LDA #$DE
306- 85 48      STA $48
308- 4C 6E A5   JMP $A56E
```

Often DOS gave me trouble with groaning, garbaged text screen parameters, etc., but a few times outputted the following (the addresses and lengths of the binary files were added later).

```
A 002 HELLO                   A$      L$
T 002 MZ.EXEC
B 007 TBLGEN.OBJ0     800     50C
B 015 DROUTS.OBJ0     6000    DD7
B 013 MZ.OBJ0         4000    B52
B 005 MZS.OBJ0        5C00    3EE
B 009 MSSL.OBJ0       800     7B9
B 034 ROBOT           2000    1FFF
I 002 APPLESOFT
```

The starting address and length data of the various files were obtained by patching in a BLOAD finder routine, even though DOS crashed after loading in the first sector of each file. Now the problem was to work around the erratic behavior of this heavily modified and messed up DOS 3.2 to get all the files loaded into memory that I wished to transfer.

I hit upon the following gimmick with trial and error; first I wrote the following driver.

```
300- AD 81 C0    LDA $C081
303- 20 51 A8    JSR $A851
306- A9 AE       LDA #$AE
308- 85 31       STA $31
30A- A9 DF       LDA #$DF
30C- 85 48       STA $48
30E- 4C D1 A4    JMP $A4D1
```

In Applesoft, I replaced the hello program with a one-liner to BLOAD the desired file into memory. Then I would type CALL 768. The disk would spin, groans would occur and the operation would stop with an error message like FILE TYPE MISMATCH or ROGRAM TOO LARGE. If I got the latter, I typed RUN. Believe it or not, *this* loaded the file without any trouble; don't ask me why. I had wormed under the DOS' malfunctioning, with whatever the pseudo-driver did.

The following instructions will get all of the machine code for Thief into memory. Ignore the error messages you get after CALL 768.

```
10 PRINT "□D BLOAD DROUTS.O
   BJ0"
CALL 768
RUN
10 PRINT "□D BLOAD TBLGEN.O
   BJ0"
CALL 768
RUN
CALL -151
5000<800.D0CM
□C
10 PRINT "□D BLOAD MZ.OBJ0"
CALL 768
RUN
10 PRINT "□D BLOAD MZS.OBJ0"
CALL 768
RUN
10 PRINT "□D BLOAD MSSL.OB
   J0"
CALL 768
RUN
CALL -151
7000<800.FB9M
```
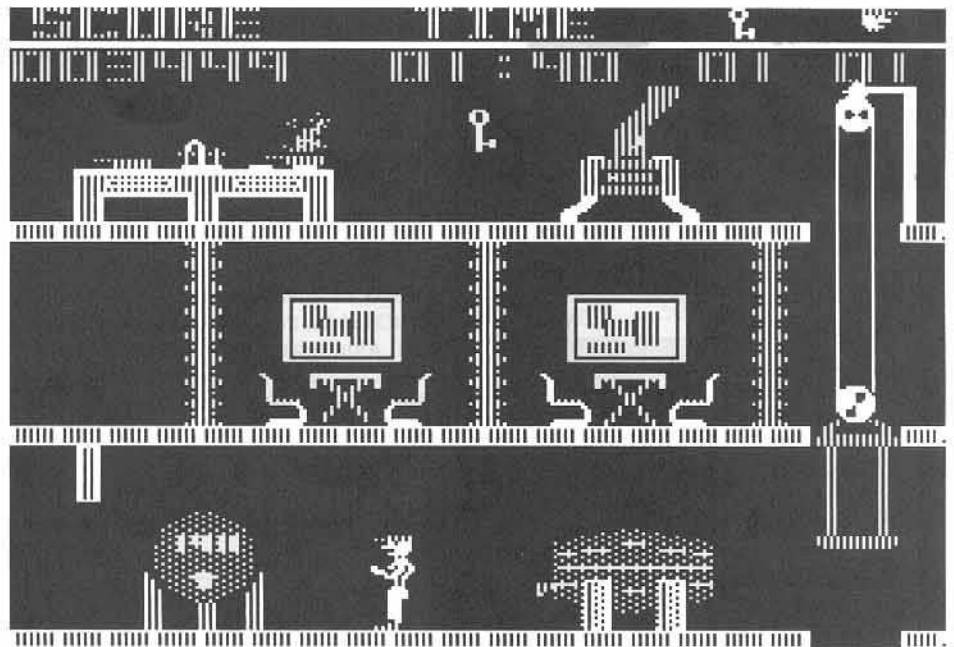
□C
```
10 PRINT "□D BLOAD ROBOT,A
   $2000"
CALL 768
RUN
```

After confirming that all files are in memory, you are now finished with the erratic Thief DOS and can boot a DOS 3.3 48K slave disk (the disk you just INITialized). Two files above were moved as they BLOAD into the same area, and this same area is also overwritten when DOS is booted. All the files are in the safe area above $2000 now.

Now perform the following BSAVES to create your deprotected user-friendly copy of Thief.

```
CALL-151
BSAVE ROBOT,A$2000,L$1FFF
800<5000.550CM
```

```
BSAVE TBLGEN.OBJ0,A$800,L$50C
BSAVE DROUTS.OBJ0,A$6000,L$DD7
BSAVE MZ.OBJ0,A$4000,L$B52
BSAVE MZS.OBJ0,A$5C00,L$3EE
800<7000.77B9M
BSAVE  MSSL.OBJ0,A$800,L$7B9
```

Last of all, return to BASIC with □C and type in this program to create an EXEC file.

```
10 PRINT "□D OPEN MZ.EXEC"
20 PRINT "□D WRITE MZ.EXEC"
30 PRINT "BLOAD DROUTS.OBJ0"
40 PRINT "BRUN TBLGEN.OBJ0"
50 PRINT "BLOAD MZS.OBJ0"
60 PRINT "BLOAD MSSL.OBJ0"
70 PRINT "BRUN MZ.OBJ0"
80 PRINT "□D CLOSE"
```

You have now created (or re-created) the MZ.EXEC file. I doped out its commands by entering MON C,I,O before having the deranged Thief DOS EXEC that file. After each echo of a command, the screen filled with rubbish, but a Reset regained control for the next command.

All should boot properly now, and loading should even sound the same, with the same arm movement sounds. The same tough game should follow. The big difference is that the code is now open to examination, backup via COPYA, and experimentation (to see if you can make the on-screen man immune to robot fire and even touching those deadly walls!). Even nicer is that the memory wipe that occurs on reset no longer works. This and the reset vector to it were built into the crazy Thief DOS, on the original disk. Your original can now be stored away while you play with the copy.

Thief has no hidden commands, nor can you restart the game by □R or whatever. The fourth level is murder! I can hardly imagine a fifth, let alone a sixth or seventh level! But now, Thief is open to check.

© 1985 THINK Technologies, Inc.
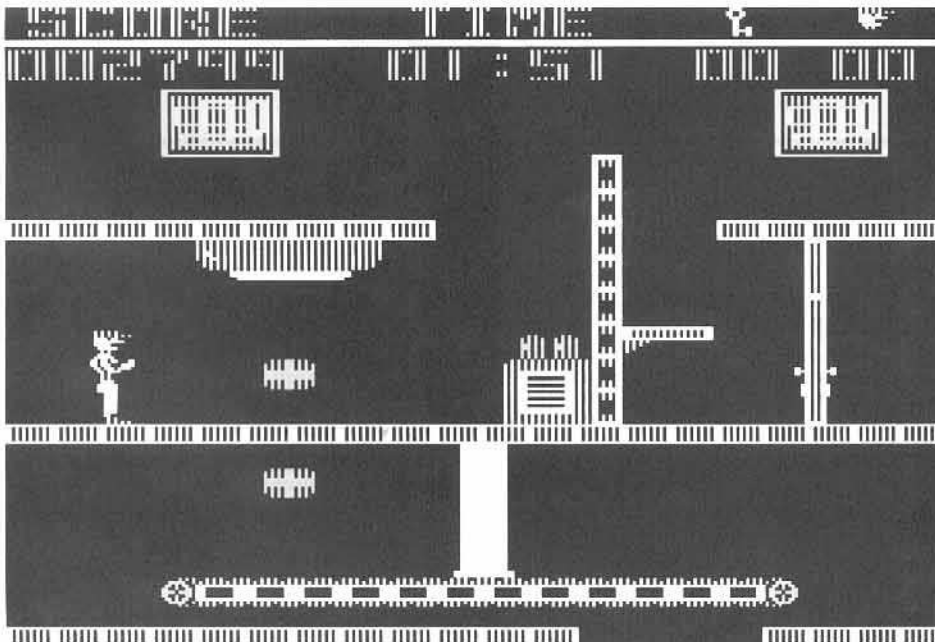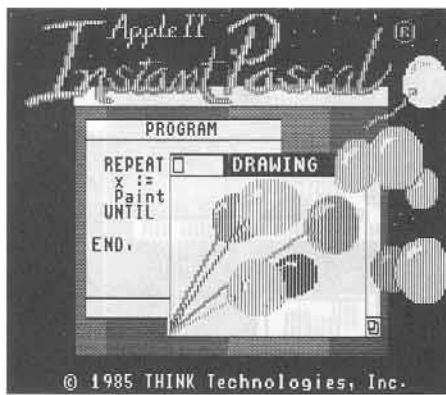
by Jonathan Maiara
and Jeff Lucia

Apple Computer, Inc.
20525 Mariani Ave.
Cupertino, CA 95014
(408) 996-1010

Requirements:
128K Apple //e or //c
Instant Pascal disk
A sector editor
COPYA
Two blank disks

Apple Computer has recently released Instant Pascal, an implementation of ANSI Pascal that uses ProDOS and takes advantage of double hi-res graphics, pull-down menus, and Macintosh-style windows. Instant Pascal (IP) is distinguished by its impressive sixteen-color double hi-res output (even text is displayed on the graphics screen) and its innovative interpreted execution. However, these features also have several very serious side effects. First, IP is extremely slow. So slow in fact, that Lethargic Pascal may be a more appropriate name for it. In addition, available memory is severely limited and disk access seems slow for ProDOS. Finally, there is a very annoying and nasty protection scheme that will, of course, be addressed here.

Since IP is ProDOS based, one might expect a special, arcane type of copy protection. This is not really the case, however; IP is protected
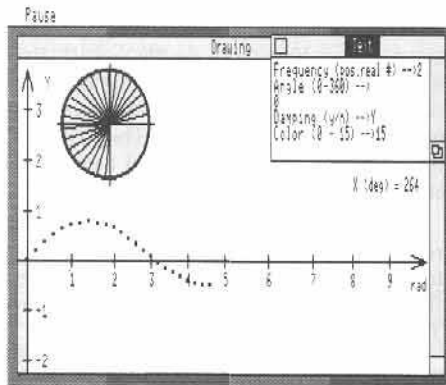
in a method that has become standard for many ProDOS based programs and isn't as unusual as it might be. Most of the disk is unprotected and has readable files. However, track $1 on side 2, has a signature (a unique sequence of bytes specially written) that is verified by a portion of the program. Aside from the signature verification routine, which uses the disk drive directly through the I/O addresses in the utility page ($C000-$C0FF), nearly all of IP's disk access is done through the MLI, ProDOS's file manager. With IP, as with many ProDOS disks, there are no changes to ProDOS itself. Although the basic method of protection is mainstream, the protection itself is very effective and very difficult for bit copiers to reproduce.

Like all ProDOS system programs, IP's interpreter (named IP.SYSTEM) is loaded at $2000 and then executed. However, IP.SYSTEM is only a loader for the rest of the system. Resident programs (programs that stay in memory all the time) that handle mouse action, memory use, disk access and display are loaded in first via IP.SYSTEM. Included in these programs is the signature verification code. This code is used to check the disk in drive one to see if it is the original IP master disk. After the resident programs are loaded, a segment file is read in pieces. This file contains the Pascal language commands, the filer commands, editing commands, and other system functions and is not protected. The protection is contained only in one part of the

resident programs.

Instant Pascal's protection is very effective for several reasons. First, the code that verifies the disk is very unusual, and has been placed in the middle of text page 1 and in the bank switched area of auxiliary memory, making it difficult to detect. The signature itself is also very unusual, and needs special code to read it because it has been written in a wierd way with a special program. The signature itself has been designed to foil copiers, as it has quite successfully (we haven't been able to get a copy of track $1 yet), and the splitting of the verification code has been designed to foil crackers (which it hasn't. Hee! Hee!).

The verification code has two segments; one is in the text page, and the other is in auxiliary memory. The first segment does the actual reading of the signature bytes (400 of them, actually) from the disk with nonstandard timing. The second segment checks the data that has been read to see if the disk in the drive is the original. The reading code is very unusual, and the hackers out there may want to eyeball some of it.

If you are at all familiar with Apple disk manipulation, you will know tha the I/O locations are usually accessed using X-indexed absolute addressing, where the x index contains the controller slot times sixteen. For example, to start the drive, the instruction LDA $C089,X would be used. If the x index has a $60 in it, the final address is $C0E9. IP's loading routine uses normal absolute addresses, which speeds

Instant

up execution very slightly. Bit copiers have trouble copying IP's protected track because the bytes on the track have been written too quickly, and the copiers miss some of the bytes due to their slower addressing techniques. The head phase timing is also nonstandard and the settling time is a little less than what is normally allowed.

The second segment of the verification doesn't read from the disk, but checks the information that has already been read. Hidden away in auxiliary bank-switched memory there are several routines that check some of the bytes that were loaded in by segment one. If the bytes are not correct, one of these routines creates a break instruction and jumps to it. Under normal operation, IP's break vector points to

routine and modified accordingly.

Now we'll give you the good stuff. Here's a step by step explanation of the deprotection procedure:

1) First, you must make copies of both sides of th IP disk. Start by running COPYA and copy the first side normally.

2) Hit ⊟C followed by a Return when asked for another copy.

3) Change line 70 to read:

70 POKE 47426,24

(This kills two birds with one stone. It keeps COPYA.OBJ0 from being reloaded and disables DOS' error checking so we won't have any trouble with track 1.)

Pascal

a routine that prints "System error #(1 or 2)," and makes you start up again.

In order to deprotect IP, it is necessary to bypass the section that creates the break instruction. Also, the disk reading routines should be removed to eliminate any unnecessary disk checking and to allow the use of IP with other kinds of disk drives, such as hard drives or RAM type devices (Note: this will require some additional modification to the program). The reset and break vectors are changed in IP.SYSTEM and can be fixed as you like (explained later). It is possible, after using the reset vector to stop execution, to preserve the contents of both verification segments in memory for examination. After this, the correct routines can be located on the disk with a search

4) RUN and copy the other side of the disk.
For the next steps you will need your sector editor.

5) Make the following changes to the **copy** of side one:

Track $B, Sector $1:

Byte(s)	Change From	To:
$0-$2	AD EC C0	B8 50 29
$2C-$2D	10 FB	A9 FF
$3A-$3C	AD EC C0	BD DD 07
$59-$5B	AD E3 C0	B8 50 18

Track $B, Sector $3:

Byte(s)	Change From	To:
$3C-$3F	AD EA C0	B0 50 1B

```
        PROGRAM
   B : INTEGER;
   I, J : INTEGER;                TEXT
   WORDS : ARRAY[1..9
 BEGIN
   B := 70;
   PENPAT(BLACK);□        OBSERVE
   PAINTRECT(0,  TYPE EXPR:
   PENPAT(MAGENTA
   PAINTRECT(0, 1
   PENPAT(BROWN);
   PAINTRECT(0,
   PENPAT(ORANGE)
   PAINTRECT(0, 42, B,
   PENPAT(DARKGREEN);
   PAINTR
   PENPAT       CLIPBOARD
   PAINTR
   PENPAT(GREEN);
```

Track $15, Sector $F:
Byte(s)	Change From			To:		
$65-$69	F0 13 A9 00			A9 00 F0 11		

6) Write protect the two disks (or sides).

7) Raid the refrigerator in the ecstasy of accomplishment.

After you have copied IP, you may change the reset and break vectors if you want to. To do this, follow these steps:

1) Start up ProDOS BASIC with the Users Disk or System Utilities.

2) Load IP.SYSTEM by typing

BLOAD IP.SYSTEM, A$2000, TSYS

3) Enter the monitor by typing CALL -151.

4) To change the reset vector, store the low byte of the target address in $2151 and the high byte in $2153. For example, to go directly to the monitor ($FF69) on reset:
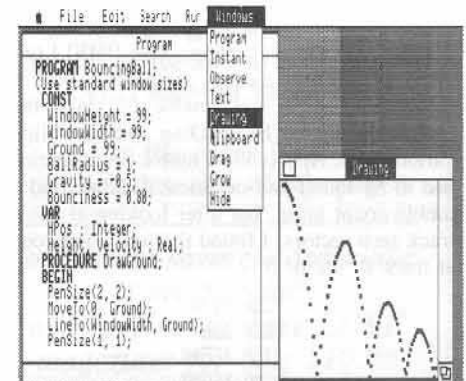
2151:69
2153:FF

5) To change the break vector, change locations $216F and $2171. If you want the registers displayed after a break ($FA59),

216F:59
2171:FA

6) Save IP.SYSTEM back to the disk by typing

BSAVE IP.SYSTEM, A$2000, TSYS

```
⌘ File Edit Search Run  Windows
                              Program
        Program               Instant
PROGRAM BouncingBall;         Observe
(Use standard window sizes)   Text
  CONST
    WindowHeight = 99;        Drawing
    WindowWidth = 99;         Clipboard
    Ground = 99;              Drag
    BallRadius = 1;           Grow
    Gravity = -0.5;           Hide
    Bounciness = 0.80;
  VAR
    HPos : Integer;                    Drawing
    Height, Velocity : Real;
  PROCEDURE DrawGround;
  BEGIN
    PenSize(2, 2);
    MoveTo(0, Ground);
    LineTo(WindowWidth, Ground);
    PenSize(1, 1);
```

World's Greatest Football Game

by Charles S. Taylor

Requirements:
Super IOB 1.5
Copy][Plus
Two blank disk sides

This is another excellent sports game from a long line of Epyx sports games. It is mostly for the football fanatic, as it is a very involved game. It is primarily a two player game. Each player selects offensive and defensive plays from a list of hundreds on the disk. If you find the selection of plays too limited, you can use the chalkboard option to design your own.

After my first attempt to copy failed I determined that the protection involved address and data epilogues altered from DE AA to FF FF. After copying the disk to standard formatting, I was able to catalog the disk, and thought my job was done. The "HELLO" program was only a decoy, or perhaps part of the protection. I was unable to make the program work by BLOADing and BRUNing various files. Apparently a nibble count routine had to be found and defeated. I didn't find a nibble count jump, but after looking at some track zero sectors, I found the following code at track 0, sector 5:

```
LDA #$00
LDX $2B
JSR $EEA0
JMP $EC00
```

```
DEC $F0
BNE $09B1
JMP $C600
```

I knew I didn't want the disk to reboot (JMP C600) so I substituted the previous JMP destination (JMP EC00) for JMP C600. It worked!

Summary

1) Copy the boot side with the controller below in Super IOB. The address and data epilogues are changed from DE AA to FF FF, and track 0, sector 5, byte B0 is changed from C6 to EC during the copy process.

2) Use COPYA to copy the back side of the disk.

controller

```
1000 REM WG FOOTBALL GAME CONTROLLER
1010 TK = 0 : LT = 35 : ST = 15 : LS = 15 : CD = WR : FAST
     = 1
1020 RESTORE : GOSUB 170 : GOSUB 490 : GOSUB 610
1030 T1 = TK : TK = PEEK (TRK) - 1 : GOSUB 310 : TK
     = T1
1040 GOSUB 230 : GOSUB 490 : GOSUB 610 : IF PEEK
     (TRK) = LT THEN 1060
1050 TK = PEEK (TRK) : ST = PEEK (SCT) : GOTO 1020
1060 HOME : PRINT "COPYᐃ DONE" : END
5000 DATA 255 ,255 ,255 ,255
5010 DATA 1ᐃ CHANGES ,0 ,5 ,176 ,236
```

controller checksums

1000	– $356B	1050	– $20D2
1010	– $2544	1060	– $0747
1020	– $73B0	5000	– $45C9
1030	– $04A7	5010	– $E0CA
1040	– $3D3D		

PLANETFAL

● To use the Shuttle Car: first, raise the speed (push lever) up to 20. Wait for some turns. When there's a sign saying "15", pull lever and wait, you'll come to a station smoothly and slowly.
● You don't need light.
● The thing in the grating is steel; the U-bar is a magnet.
● The monsters in the computer like warm things.
● There's something behind the painting.....
● Floyd must die, but don't cry, you'll see him again!
● Interface #2 isn't okay.
● There's something behind the robot door, but you won't fit in, think about someone who will.
● Don't give anything to Floyd.
● Play with Floyd in the elevator; you have extra time.
● You'll not need the helicopter.
● The colored lamp in The Communications Room means that that color of liquid is needed.
● Don't fight against the mutants, just fool them.
● The magnet will do no good to Access Cards.
● You can sleep on the floor, if you're tired enough.
● The strange language isn't too hard, and it's not extremly necessary.
● Don't go back with the Shuttle Car that you came with.

By Aapo Puskala

ESCAPE FROM RUNGISTAN

● Think about NESSEN while flying.
● Rungistanians DO drink, you don't need a drink.
● Booze does the same thing to other people than it does to you. Save the game, and drink it.
● Skiing is quite hard; save the game when you've skied enough.
● The bear has relatives.
● Try "DROP WATER".
● The guard in the tower is interested in helicopters, run when he's looking at one.
● The cat likes mice; give one to him.
● Light the dynamite with the magifying glass.
● The fence is electric.
● What do you do when the gate is down?
● The boy likes chocolate.

By Aapo Puskala

Description of Available Back Issues

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 I | ....

25 *Softkeys* | DB Master 4.2 | Business Writer | Barron's Computer SAT | Take 1 | Bank Street Speller | Where In The World Is Carmen Sandiego | Bank Street Writer 128K | Word Challenge | *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 | ..

23 *Softkeys* | Choplifter | Mufplot | Flashcalc | Karateka | Newsroom | E-Z Draw | *Readers' Softkeys* | Gato | Dino Eggs | Pinball Construction Set | TAC | The Print Shop: Graphics Library | Death In The Caribbean | *Features* | Using A.R.D. To Softkey Mars Cars | How To Be The Writemaster | *Core* | Wheel Of Money | ....

22 *Softkeys* | Miner 2049er | Lode Runner | A2-PB1 Pinball | *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 | .....

17 *Softkeys* | The Print Shop | Crossword Magic | The Standing Stones | Beer Run | Skyfox | Random House Disks | *Features* | A Tutorial For Disk Inspection and the Use Of Super IOB | S-C Macro Assembler Directives (reprint) | *Core* | The Graphic Grabber For The Print Shop | The Lone Catalog Arranger v1.0 Part 2 | ....

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 | ....

13 *Softkeys* | Laf Pak | Beyond Castle Wolfenstein | Transylvania | The Quest | Electronic Arts | Snooper Troops (Case 2) | DLM Software | Learning With Leeper | TellStar | *Core* | CSaver: The Advanced Way to Store Super IOB Controllers | Adding New Commands to DOS 3.3 | Fixing ProDOS 1.0.1 BSAVE Bug | *Review* | Enhancing Your Apple | *Feature* | Locksmith 5.0 and Locksmith Programming Language | ....

11 *Softkeys* | Sensible Speller | Exodus: Ultima III | *Readers' Softkeys* | SoftPorn Adventure | The Einstein Compiler v5.3 | Mask of The Sun | *Features* | Copy][Plus (4.4C): Update Of An Old Friend | Parameter List For Essential Data Duplicator | *Core* | Ultimaker III | The Mapping of Ultima III | Ultima II...The Rest Of The Picture | ....

7 *Softkeys* | Zaxxon | Mask of the Sun | Crush Crumble & Chomp | Snake Byte | DB Master | Mouskattack | *Features* | Making Liberated Backups That Retain Their Copy Protection | S-C Assembler: Review | Disk Directory Designer | *Core* | Corefiler: Part 1 | Upper & Lower Case Output for Zork | ....

4 *Softkeys* | Ultima II | Witness | Prisoner II | Pest Patrol | Adventure Tips for Ultima II & III | Copy II Plus PARMS Update | *Feature* | Ultima II Character Editor | ....

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 |

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 not listed are no longer available.
But disks are still available for
ALL sold-out issues !

Use the order form
on the other side of this page

*For special savings, order our
'Core Special'
and receive all three CORE
magazines for only $10.00*

Issue	Mag $4.75	Disk $9.95	Both $12.95
33	☐	☐	☐
32	☐	☐	☐
31	☐	☐	☐
30	☐	☐	☐
29	☐	☐	☐
28	☐	☐	☐
27	☐	☐	☐
26	☐	☐	☐
25	☐	☐	☐
24	☐	☐	☐
☆ 23 . .	☐	☐	☐
22	☐	☐	☐
21	NA	☐	NA
20	☐	☐	☐
19	☐	☐	☐
18	NA	☐	NA
17	☐	☐	☐
16	☐	☐	☐
15	NA	☐	NA
14	NA	☐	NA
☆ 13 . .	☐	☐	☐
12	NA	☐	NA
☆ 11 . .	☐	☐	☐
10	NA	☐	NA
9	NA	☐	NA
8	NA	☐	NA
☆ 7 . . .	☐	☐	☐
6	NA	☐	NA
☆ 4 . . .	☐	☐	☐
3	NA	☐	NA
Core 2 .	☐	☐	☐
2	NA	☐	NA
1	☐	☐	☐
Core 1 .	☐	☐	☐
Core 3 .	☐	☐	☐
Computing 3	☐	NA	NA
Best of Hardcore Computing . .	NA	☐	NA

Core Special $10.00 ☐
(All three CORE magazines)

Special "Both" disk & magazine combination orders apply to one issue and its corresponding disk.

Some disks apply to more than one issue and are shown as taller boxes.

☆ We have a limited supply of these issues.

BACK ISSUES
and
LIBRARY DISKS

of COMPUTIST (*formerly Hardcore COMPUTIST*) *are still available, though some issues (marked NA) are sold out, library disks are available for ALL issues of COMPUTIST.*

LIBRARY DISKS
are perfect companions for
COMPUTIST

Documentation for Library Disks is in the corresponding issue.

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

Name _____ ID# _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone _____

[VISA] [⬤] _____ - _____ - _____ Exp. _____

Signature _____ CP33

Send check or money order to: COMPUTIST PO Box 110846-T Tacoma, WA 98411. Most orders shipped UPS so please use street address. Offer good while supply lasts. In Washington state: add 7.8% sales tax.

Back Issue Rates For Foreign Orders

FOREIGN MAGAZINE ORDERS ••••••••••••••••••••••••••••••
Price for each magazine includes shipping.

	1 - 2 copies	3 to 4 copies	5 or more copies
Canada/Mexico	..... $8.00 each	..... $7.00 each	..... $6.00 each .....
Other Foreign	..... $14.25 each	..... $13.25 each	..... 12.25 each .....

FOREIGN DISK ORDERS •••••••••••••••••••••••••••••••••••

Disks are $11.94 each (includes shipping). Special "Both" disk and magazine combinations shown do NOT apply to Foreign orders. US funds drawn on US banks. All foreign orders sent AIR RATES.

Who are COMPUTISTs and what do they want?

The editors at COMPUTIST are in the process of evaluating our service to you, the readers, and would like to get your opinion on this subject.

To help us provide you with the kinds of articles and columns that you want, please complete the following questionnaire and return this form along with your name and address to:

COMPUTIST
PO Box 110846-T
Tacoma, WA 98411

Background

1. What is your age?
- [1] Under 20
- [2] 20—24
- [3] 25—29
- [4] 30—39
- [5] 40—59
- [6] 60 and over

2. Sex:
- [1] male
- [2] female

3. Level of education:
- [1] high school
- [2] vocational school
- [3] college

4. Head of household?
- [1] yes
- [2] no

5. Personal or family annual income:
- [1] under 10,000
- [2] 10—15,000
- [3] 15—20,000
- [4] 20—25,000
- [5] 25—30,000
- [6] 30—40,000
- [7] 40—50,000
- [8] 50—75,000
- [9] over 75,000

6. Do you have an Apple or clone at home?
- [1] yes
- [2] no

7. Do you have an Apple or clone at work?
- [1] yes
- [2] no

8. Do you belong to a users' group or club?
- [1] yes
- [2] no

9. How many (waking) hours do you spend with your computer each week?
- [1] under 5
- [2] 5-10
- [3] 10-20
- [4] More than 20

10. How did you find out about COMPUTIST?
- [1] friend
- [2] mentioned on BBS (which?)
  _____
- [3] mentioned in magazine article:
  _____
- [4] magazine ad (where?)
  _____

11. How many additional people read your issue of COMPUTIST?
- [1] 0
- [2] 1
- [3] 2-3
- [4] 4+

12. What do you do with your issue of COMPUTIST?
- [1] Keep it in an underground vault
- [2] Give it to a friend
- [3] Give a friend a photocopy of it
- [4] Let your friend photocopy it

Knowledge

13. Rate your knowledge in the following subjects:

[1] Novice   [2] Intermediate   [3] Expert

	1	2	3
BASIC	☐	☐	☐
machine language (ML)	☐	☐	☐
DOS 3.3 from BASIC	☐	☐	☐
ProDOS from BASIC	☐	☐	☐
DOS 3.3 from M.L.	☐	☐	☐
ProDOS from M.L.	☐	☐	☐
Software protection	☐	☐	☐
Hardware stuff	☐	☐	☐

14. Can you read & write BASIC?
- [1] yes
- [2] no

15. Can you read & write machine language?
- [1] yes
- [2] no

16. Can you build a circuit from a schematic?
- [1] yes
- [2] no

Interests

17. Rate your interest level in the following:

[1] None   [2] Some   [3] A lot

	1	2	3
BASIC	☐	☐	☐
machine language (M.L.)	☐	☐	☐
DOS 3.3 from BASIC	☐	☐	☐
ProDOS from BASIC	☐	☐	☐
DOS 3.3 from M.L.	☐	☐	☐
ProDOS from M.L.	☐	☐	☐
Software protection	☐	☐	☐
Hardware stuff	☐	☐	☐

18. I want to see Fewer, the Same, or More of the following:

Softkeys for:

	Fewer 1	Same 2	More 3
Business software	☐	☐	☐
Utility software	☐	☐	☐
Game software	☐	☐	☐

Reviews of:

	1	2	3
Utility software	☐	☐	☐
Deprotection hardware	☐	☐	☐
Peripherals	☐	☐	☐
Game software	☐	☐	☐

Articles:

	1	2	3
The CORE section	☐	☐	☐
Machine language	☐	☐	☐
Utility programs	☐	☐	☐
Fun programs	☐	☐	☐
DOS tricks	☐	☐	☐
Hardware tricks	☐	☐	☐
ROM modifications	☐	☐	☐

19. Would you like to see tutorials on:
- BASIC [1] yes [2] no
- Machine language [1] yes [2] no
- Bit copiers [1] yes [2] no
- Hardware [1] yes [2] no

20. Would you like to see other computers covered?
- [1] yes
- [2] maybe
- [3] no

21. Would you like to see more pages?
- [1] yes
- [2] maybe
- [3] no

22. Would you like to see readers' classified ads?
- [1] yes
- [2] maybe
- [3] no

23. Would you be interested in some cooperative purchasing deals (buyers' groups?)
- [1] yes
- [2] maybe
- [3] no

If 'yes' or 'maybe', what products? _____
_____
_____
_____
_____
_____
_____

Circulation

24. Are you receiving your current magazines regularly?
- [1] yes
- [2] no

25. How quickly do you receive your product orders?
- [1] Less than 1 week
- [2] 1 to 2 weeks
- [3] 2 to 3 weeks
- [4] more than 3 weeks

Service

26. How would you grade (A—F) COMPUTIST:

	A 1	B 2	C 3	D 4	F 5
EDITORIAL CONTENT					
useful	☐	☐	☐	☐	☐
informative	☐	☐	☐	☐	☐
easy to understand	☐	☐	☐	☐	☐
accurate	☐	☐	☐	☐	☐
up-to-date	☐	☐	☐	☐	☐
unique information	☐	☐	☐	☐	☐
reference value	☐	☐	☐	☐	☐
holds interest	☐	☐	☐	☐	☐
VISUAL PRESENTATION	1	2	3	4	5
readable text	☐	☐	☐	☐	☐
presentation of articles	☐	☐	☐	☐	☐
layout	☐	☐	☐	☐	☐
size	☐	☐	☐	☐	☐
neatness	☐	☐	☐	☐	☐
print quality	☐	☐	☐	☐	☐

27. Do you intend to renew your subscription?
- [1] Yes
- [2] No

28. If not, do us a favor and tell us why below. Thank you.
_____
_____
_____

Literature

29. What 'Apple' literature do you have?
- 1 ▢ Apple // Reference Manual
- 2 ▢ DOS 3.3 Manual
- 3 ▢ Beneath Apple DOS / Beneath Apple ProDOS
- 4 ▢ book on Apple machine language
- 5 ▢ Applesoft or DOS source code
- 6 ▢ Reverse-engineered the schematics

30. What 'Apple' magazines do you read?
- 1 ▢ Nibble
- 2 ▢ Incider
- 3 ▢ A+
- 4 ▢ Call -APPLE
- 5 ▢ Open-Apple
- 6 ▢ Apple Assembly Lines
- 7 ▢ other:_____

31. What other computer magazines do you regularly read?
- 1 ▢ BYTE
- 2 ▢ Dr Dobbs Journal
- 3 ▢ Computer Shopper
- 4 ▢ Radio Electronics
- 5 ▢ Compute!
- 6 ▢ InfoWorld
- 7 ▢ other:_____

32. What Bulletin Boards do you use?
- 1 ▢ Compuserve
- 4 ▢ The Source
- 2 ▢ Pirates Harbor
- 5 ▢ Local BBSs
- 3 ▢ other:_____

Equipment

33. Which Apple or clone do you have?
- 1 ▢ Apple][
- 6 ▢ Apple][Plus
- 2 ▢ Apple //e
- 7 ▢ Apple //c
- 3 ▢ Franklin 1000
- 8 ▢ Franklin 2000
- 4 ▢ Laser 128
- 5 ▢ other:_____

34. What accessories does your computer have?
- 1 ▢ printer
- 2 ▢ modem
- 3 ▢ 80 column card
- 4 ▢ second 5-inch floppy
- 6 ▢ Unidisk 3.5
- 7 ▢ dedicated RAM disk
- 8 ▢ Apple 1 Meg card
- 9 ▢ other RAM cards
- 10 ▢ copy card
- 11 ▢ modified ROMs
- 12 ▢ other:_____

35. What microprocessor do you use (motherboard)?
- 1 ▢ 6502
- 4 ▢ 65C02
- 2 ▢ 65802
- 5 ▢ 65816
- 3 ▢ other:_____

36. What coprocessor(s) do you use in your computer?
- 1 ▢ 65816
- 5 ▢ 68000
- 2 ▢ 6809
- 6 ▢ 8088 or 8086
- 3 ▢ Z80 (including CP/M)
- 4 ▢ other:_____

37. Do you have a language card?
- 1 ▢ yes
- 3 ▢ no
- 2 ▢ built-in (64K motherboard)

38. Besides a language card, what extra RAM do you have?
- 1 ▢ language card style (Know-Drive, Titan, etc.)
- 2 ▢ extended 80 column card style (Ramworks, Multiram, etc.)
- 3 ▢ other style or RAM disk
- 4 ▢ not sure, but I have some

39. What is your primary DOS for the Apple?
- 1 ▢ 3.3
- 4 ▢ ProDOS
- 2 ▢ CP/M
- 5 ▢ Pascal
- 3 ▢ other:_____

40. How do you use the source code printed in COMPUTIST?
- 1 ▢ rarely use them.
- 2 ▢ study them carefully
- 3 ▢ enter them with my assembler

41. What software do you have?
- 1 ▢ word processor
- 2 ▢ Applesoft program line editor
- 3 ▢ assembler
- 4 ▢ Locksmith
- 5 ▢ Copy II Plus
- 6 ▢ EDD
- 7 ▢ Echo Plus
- 8 ▢ other bit copier:_____
- 9 ▢ sector editor
- 10 ▢ disk search utility
- 11 ▢ other:_____

42. How much did you spend on your hardware?
- 1 ▢ under $1000
- 3 ▢ $1500—$200
- 2 ▢ $1000—$1500
- 4 ▢ over $2000

43. How much on your software?
- 1 ▢ under $250
- 4 ▢ $750—$1000
- 2 ▢ $250—$500
- 5 ▢ over $1000
- 3 ▢ $500—$750

44. How much do you intend to spend on hardware in the next year?

$_____

45. How much do you intend to spend on software in the next year?

$_____

46. What computer other than an Apple or clone do you own?
- 1 ▢ I don't
- 2 ▢ IBM or clone
- 3 ▢ CP/M-based
- 4 ▢ Macintosh
- 5 ▢ Atari ST
- 6 ▢ Amiga
- 7 ▢ other Commodore
- 8 ▢ other Atari
- 9 ▢ other toy-level computer
- 10 ▢ something from Radio Shack
- 11 ▢ something from Cray

47. Would you like COMPUTIST to carry products recommended in articles?
- 1 ▢ yes
- 2 ▢ no

'How-to' Hardware Projects

This is an important section to us. We want to know how **interested** you are in hardware oriented articles and projects.

48. What level of interest do you have in hardware modification projects:
- 1 ▢ none
- 2 ▢ some
- 3 ▢ high

49. What about hardware construction projects:
- 1 ▢ none
- 2 ▢ some
- 3 ▢ high

50. And what about your computer or its peripherals:
- 1 ▢ You want to learn to modify it.
- 2 ▢ You plan to modify it.
- 3 ▢ You have modified it.
- 4 ▢ You've built your own.

51. I would like to see COMPUTIST publish hardware projects / modifications in the cost range of:
- 1 ▢ under $25
- 2 ▢ under $75
- 3 ▢ under $150

52. Check the subjects you would like to see covered in COMPUTIST.

Things to plug into the joystick port
- 1 ▢ simple data transfer
- 2 ▢ real world interfacing
- 3 ▢ gizmos with flashing lights
- 4 ▢ gizmos with neat sounds
- 5 ▢ other:_____

Things to plug into slots
- 1 ▢ deprotection aids
- 2 ▢ RAM cards
- 3 ▢ ROM cards
- 4 ▢ memory
- 5 ▢ computer-on-a-board

Things to do with common peripherals
- 1 ▢ modems
- 2 ▢ video
- 3 ▢ sound
- 4 ▢ disk drives
- 5 ▢ other:_____

Things to do to your motherboard
- 1 ▢ keyboards
- 2 ▢ speed enhancements
- 3 ▢ memory enhancements
- 4 ▢ alien processors
- 5 ▢ I/O devices
- 6 ▢ common peripherals
- 7 ▢ building a Cray into a toaster

53. Your comments:

_____
_____
_____
_____
_____
_____
_____
_____
_____

Thank you for telling COMPUTIST who you are and what you want from future COMPUTISTs.

Writer's Guide

COMPUTIST

is a monthly magazine dedicated to the serious user of the Apple (or compatible) computer. COMPUTIST welcomes articles on a variety of subjects in all levels of technical difficulty but requires accurate data, technical competence, correct English usage, readable style, and fully defined jargon and buzzwords.

MANUSCRIPT MECHANICS

All manuscripts must be typed or printed on one side of the paper. Text should be double-spaced.

Printouts should use a non-compressed font with both upper and lower case. A letter quality mode is preferred, with each page torn at the perforation only. Pages need not be stapled together. The cover page of each manuscript should contain the following data:

TITLE OF WORK
FULL NAME OF AUTHOR
ADDRESS
PHONE NUMBER

Each page of the manuscript and program listing should include the author's name, the title of the work, and the page number in the upper right hand corner.

The article and any accompanying program **SHOULD BE SUBMITTED AS A STANDARD TEXT FILE ON A DOS 3.3 DISK.** Label the disk with the title of the work and the author's full name and address. **ON DISK, TEXT MUST BE SINGLE-SPACED ONLY.** Please identify your editing program.

Original disks are always returned as soon as possible. Other materials will be returned only when adequate return packaging and postage is enclosed. We are not responsible for unreturned submissions. We *will guarantee* the return of original commercial disks mailed to us for verification of an accompanying softkey.

You will be notified of the status of your submission within 4 to 6 weeks after it is received if the article is a softkey accompanied by an original disk. Please submit completed manuscripts directly; do not query first. Previously published material and simultaneous submissions are not accepted.

SUBJECTS

We prefer material on these topics:

1) Original program/article combinations
2) General articles (Apple computing)
3) Softkeys
4) Advanced Playing Techniques (APT's)
5) Hardware modifications
6) DOS modifications
7) Product reviews (hardware and software)
8) Utilities
9) Bit Copy Parameters

WRITING YOUR ARTICLE

Observe the following points of style.

A. Always assume that your reader is a novice and explain all buzzwords and technical jargon. Pay special attention to grammar and punctuation; we require technical competence but also good, readable style.

B. Whenever appropriate, a list of hardware and software requirements should be included at the beginning of the manuscript. When published, this list will be offset from the main text.

C. Include the name and address of the manufacturer and the price when a commercial program is mentioned. This is of particular importance in PRODUCT REVIEWS.

D. When submitting programs, first introduce the purpose of the program and features of special interest. Include background information describing its use. Tips for advanced uses, program modifications, and utilities can also be included. Avoid long print statements and use TABs instead of spaces.

Remember: A beginner should be able to type the program with ease.

E. A PROGRAM is not accepted for publication without an accompanying article. These articles, as well as articles on **hardware** and **DOS modifications** MUST summarize the action of the main routines and include a fully remarked listing.

F. GENERAL ARTICLES may include advanced tips, tutorials, and explorations of a particular aspect of Apple computing.

G. SOFTKEYS of any length are acceptable and must contain detailed step-by-step procedures. For each softkey, first introduce the locking technique used and then give precise steps to unlock the copy-protected program. Number each step whenever possible. We accept articles which explain locking techniques used in several programs published by the same company.

H. When altering game programs, the changes made are sometimes extensive enough to warrant the title of ADVANCED PLAYING TECHNIQUE (APT). APTs can deal with alterations to a program, deleting annoying sounds, acquiring more points in play and avoiding hazards. Again, provide step-by-step instructions to complete each APT and explain each step's function. APT's of 100 words or more are preferred.

AUTHOR'S RIGHTS

Each article is published under the author's byline. As a rule, all rights, as well as one-time reprint rights are purchased. Purchase of exclusive rights to programs is required; however, alternate arrangements may be made with individual authors depending on the merit of the contribution.

PAYMENTS

COMPUTIST pays upon publication. Rate of payment depends on the amount of editing required and the length of the article. Payment ranges from $20 to $50 per typeset page for an article. We also pay $10 to $20 for short softkeys and APT's. A fully explained softkey accompanied by the commercial disk for verification may earn up to $50 per typeset page.
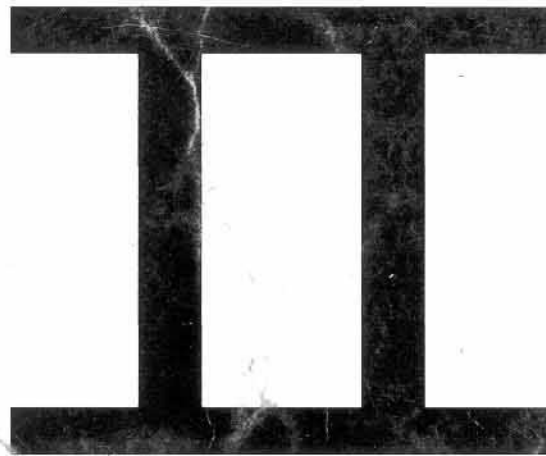
Please mail your submissions to:

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