84 COMPUTIST

Canada & Mexico $7

U.S. $3.75

Contents

COMPUTIST

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

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

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

• COMPUTIST does NOT purchase editorial material. The entire editorial content consists of information submitted to COMPUTIST for publication in the shared interests of all COMPUTISTs.

SUBSCRIPTIONS: Rates (for 8 issues):

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

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

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

Issues missed due to non-receipt of change of address may be acquired at the regular back issue rate.

We are not responsible for missing issues 90 days after mailing date. If you do not receive an issue at the usual time each month, please call or write.

Readers Data EXchange

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

What is a softkey, anyway?

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

Commands and control keys

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

6 ctrl P

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

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

Software recommendations

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

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

Super IOB and Controllers

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

Reset into the Monitor

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

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

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

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

Apple //e, //c: Install a modified CD ROM on the computer's motherboard that changes the open-apple ctrl reset vector to point to the monitor. *(This will void an Apple //c warranty since you must open the case to install it.)*

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

Recommended literature

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

Typing Applesoft programs

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

```
10HOME:REMCLEAR SCREEN
```

The LIST will look like:

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

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

NOTE: If you want your checksums to match, only type spaces within quotes or after DATA statements if they are shown as (◊) charactors. **SAVE** the program at periodic intervals using the name given in the article. All characters after a REM are not checked by the checksum program so typing them is optional.

Typing Hexdumps

Machine language programs are printed in COMPUTIST as hexdumps, sometimes also as source code.

Hexdumps are the shortest and easiest format to type in. You must first enter the monitor:

CALL -151

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

3D0G

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

Typing Source Code

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

Computing checksums

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

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

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

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

SSSS.EEEE ctrl Y

Correct the lines where the checksums differ.

Writing to the RDEX editor

RDEX (are-decks) stands for: **R**eader's **D**ata **EX**change. We print what you write. When you send in articles, softkeys, APTs, etc., you are submitting them for *free* publication in this magazine. RDEX does *not* purchase submissions nor do we verify data submitted by readers. If you discover any errors, please let us know so that we may inform our other readers.

Remember that your letters or parts of them may be used in RDEX even if not addressed to the RDEX editor. Correspondence that gets published may be edited for clarity, grammar and space requirements.

Because of the great number of letters we receive and the ephemeral and unpredictable appearance of our volunteer staff, any response to your queries will appear only in RDEX, so it would be more appropriate for you to present technical questions to the readers and ask for their responses which will then be placed in the Apple-RDEX.

How to get a free library disk

Whenever possible, send everything on Apple format (5.25" - DOS/ProDOS or 3.5" - ProDOS) or IBM format (3.5") disks. Other formats are acceptable but there may be some delay as we look for someone to translate it for us. *(If you use a 5.25" disk, when we print your letter, we will return your disk with the current library disk copied onto it.)* Use whatever text editor you like, but tell us which one. Put a label on the disk with your name (or pseudonym) and address (if you want to receive mail). Don't reformat any programs or include them in the text of your letter. Send Applesoft programs as normal Applesoft files and machine language programs as normal binary files. We have programs to convert them to the proper format for printing. If you are sending source code files, and you are not using the S-C Assembler, send them as normal text files.

When to include a printed letter

Don't include hardcopy (printout) unless:

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

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

Writing to get help

When writing to request help, be sure to include ALL relevent information. The more information you include, the easier it is to find a solution. There's an old saying that goes "A properly framed question includes 90% of the answer".

How to get mail

If you are interested in receiving mail from other readers, be sure that we have a current address. If you use a pen name and want to receive mail, we need to have your address. Our readers privacy is important, so we will not print your address unless you specifically say too.

How to write to RDEX authors

When writing to one of the RDEX authors. Write your letter and seal it in an envelope. Put your return address, the authors name (as it appears in RDEX) and *the correct postage* on the envelope. Put this envelope into another and send it to RDEX. We will put the correct address on your letter and mail it for you. Check to the right of the authors name to see if the author is writing from a foreign country and include the proper postage.

Help Line

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

Jack Nissel (Disk Protection, 7-10PM EST) (215) 365-8160
Marc Batchelor, 6025 Coker St., Cocoa, FL 32927
Rich Etarip, 824 William Charles Ct. #2, Green Bay, WI 54304-4018

The BBS
(Bulletin Board System)

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

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

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

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

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

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

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

United States Code title 17, §117

COMPUTIST

84

U.S. $3.75

Canada & Mexico $7

Table of Contents

Jeff Hurlburt TX

The PRODUCT MONITOR

RATINGS

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

GD 301: Spring Seminar (11AM session)

Good morning, students. Despite anticipations of impending frolics, I trust we are all prepared for today's topics. First, as scheduled, "Every Game Needs It".

Toward the end of our last meeting, I asked you to turn in one or two ideas as to what "it" is. According to my tally, "Good graphics and sound", and "user friendliness" were the big 'vote-getters', followed closely by "good documentation". Alas, while these features, along with "speed", "attractive packaging", etc., are frequently important, we can easily come up with exceptions. For instance, most of you enjoyed Infocom's text-only "Enchanter" trilogy.

The closest response was "Meaningfulness"— decidedly ambiguous, but, I think, you have the right idea. What every game needs is "Purpose", something you find in the game which makes playing 'feel' worthwhile. Hot-sticking arcades, for example, employ the oldest trick in the book: they teach. We all enjoy learning— really, we do; it's only when classroom instruction, etc. fails to help us learn and develop that we start to believe that "learning" is boring. Well designed arcades teach all kinds of coordination skills; and, in fact, practically every decent game exercises a host of problem-solving abilities. If you're challenged by a puzzle, a maze, a combat situation, or whatever, it's a cinch that success means you have learned something.

People play games for fun. Overcoming challenges is fun; but many games employ yet another "purpose" 'hook'. They present a scenario and try to persuade you that, as the "the Last Starfighter", "dauntless avatar", etc., you have great deeds to accomplish. If it works— if you become involved in the scenario— then completing mission objectives, cracking the puzzle, rescuing the princess, etc. become important, worthwhile achievements.

"Purpose", "meaningful challenge", "involvement", ...there are lots of ways to say the same thing. However you express the idea, it comes down to this: the player has to feel that what happens in the game matters. You can be attracted to a game by fantastic graphics, sound effects, and music; you won't play for long unless you care about what happens.

Wing Commander II: Vengeance of the Kilrathi

★★★★

$79.95 for EGA-VGA 640K PC

Origin

AdLib, Sound Blaster, or Roland sound, joystick, & 512K Expanded Memory recommended

Wing Commander II: Special Operations 1

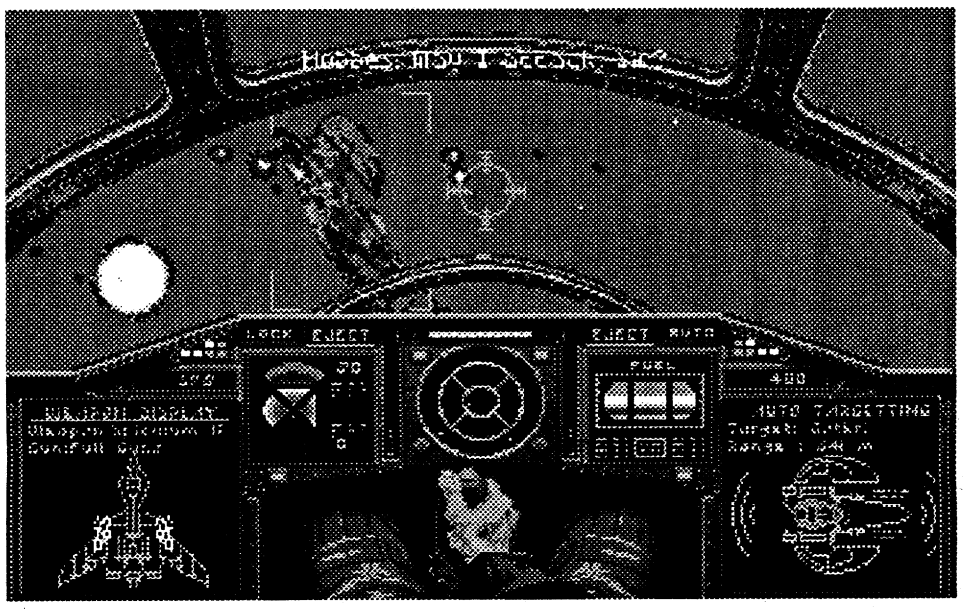
★★★★

$39.95 for EGA-VGA 640K PC

Origin

Requires Wing Commander II

No question about it, defending the fledgling Terran Confederation of Planets (human, nice, peace-loving) from the sprawling militarist Kilrathi Empire (tigers, mean, warlike) qualifies as a worthy occupation. It's too bad the first Wing Commander spends so little program space upon the story and supporting characters. As a TCP Navy pilot, you mainly just fly one mission after another— sort of like stringing together all of the Briefing Room and combat takes from a WWII carrier war flick, pretty dry stuff. I can not really blame Baywoof, who bought WC1 (before the review appeared), for bailing out after only four missions.



Wing Commander II' does it right. Instead of 'a space combat game player', you become a space combat pilot. It starts with a 256-color movie-style introduction— you see and hear the Kilrathi emperor announcing his determination to crush the upstart humans, complete with speech and stirring background music. (You get Sound Blaster speech only during the intro. An optional "Speech Package" supplies voices for the entire game.)

So far, so good; barely five minutes into the game and I was already just a tad chapped by this over-sized pussy cat's attitude. The real 'hook' comes once you get into the missions. Each is set between animated story segments which can run for a minute or more. Usually, this means that a briefing (which may be part of a larger 'intro' scene) is followed by the mission which is followed by another scene where characters and events react to mission results. (When starting from a saved position, you can choose "View Story" to see the sequence again before getting into the next mission.) Everything fits together so smoothly, you hardly notice that the missions you fly have become part of the story.

Once you're in, you're in for the duration. For starters (in the best WWII flick tradition) you have 'something to prove'. When your old ship, the "TCS Tiger Claw" was destroyed, you (one of the few survivors) were held to be partly responsible. The Court Martial finding and demotion to captain were, of course, unjust; but you could not establish that you were delayed by an encounter with cloaked enemy fighters. (No one in Navy command believes Kilrathi "stealth fighters" exist!) Some fellow pilots consider you a coward and the Admiral makes no secret of his distrust— which explains why you, a flying ace respected by the enemy, are stuck with patrol duty on a behind-the-lines space platform.

It turns out the 'lines' are closer than anyone thought! Soon, you get to show you've still got the right stuff and win a transfer to the "TCS Concordia". Great! because "'Concordia" becomes the prime target in an escalating Kilrathi threat to the critical Enigma Sector. Better yet, your squadron commander is Colonel Jeanette "Angel" Deveraux! 'All business' in squad room briefings, the beautiful Jeanette still regards you with an affection which grows even as disaster looms and each mission becomes more perilous.

The clincher is that the feeling is mutual. When, in the briefing for what promises to be a particularly dangerous battle, Jeanette announces that, due to a shortage of pilots, she is going to be flying and as your wingman— well, you will feel like telling her to stay on board and do 'squadron commander stuff'. You won't; (you have no direct control of the dialogue); besides, she's right about needing every available pilot, and Jeanette is one of the best. Rivalries and friendships, the presence of an unknown Kilrathi sympathizer, sabotage, a deadly struggle for power among the Kilrathi, and much more— the second Wing Commander is first-rate 'space opera'; and, YOU are the pivotal character. Your performance decides the flow of events from small storyline shifts through ultimate victory or defeat!

To win in Wing Commander II you will fly about thirty missions; a successful 'Special Operations 1 campaign is nineteen. Some are simple patrols— you and a wingman, usually flying light fighters, auto-pilot through a string of Nav Points and eliminate any enemy craft discovered. The majority, however, have specific objectives such as rescuing a trader, securing a hyperspace Jump Point, intercepting an enemy message module, destroying a task force, knocking out a star base, or defending a Kilrathi planet in rebellion against the Empire.

The stakes are high; and the Navy does its best to see you get the right craft for each assignment. Even so, in the thick of some combats you will swear that whoever ordered Ferrets (or Epees, or whatever) for this mission had to be crazy! Mainly, you will be assigned a medium or heavy fighter like the Rapier or Sabre. Sometimes, you'll draw the Broadsword heavy bomber and, in 'Special Operations 1 you put the Navy's super secret Crossbow attack bomber to the ultimate test. Occasionally, you fly solo; but, mostly, you'll have one or more shipmates as wingmen. Getting the most from their presence is part of the challenge— sometimes a bigger part than you imagine. These are people with abilities, prejudices, and problems which can produce some real surprises. Indeed, so can enemy aces like Prince Thakrath and Admiral Khasra.

Almost always, several outcomes are possible besides outright success or failure. For instance, avoiding an encounter with a large fighter group might improve your chances of completing the mission's main objective, but make for a rougher challenge later on (when the fighters reappear). Going for an attack on a capital ship no one expected to encounter could make finishing a mission impossible; yet, potting it may greatly improve the strategic situation for Earth. Part of the realism and excitement of play comes with the freedom to make risky command decisions which can affect other characters, the course of the war, and story content. A very bad decision, getting yourself killed, or too many non-fatal failures in a row ends the game. You will get to see the awful result (e.g. the "'Concordia" is destroyed, the Kilrathi sweep on to victory, etc.); then, you must restore a saved position (one of up to eight) or restart.

Happily for newcomers, the game does not assume WC1 experience or, even, any particular combat sim expertise. Nor must you pour over the "Play Manual" or study your copy of "Joan's Fighting Spacecraft" before getting into action. The games's designers know you won't anyway; so, the first mission is easy enough to finish with barely a glance at the controls "Reference Card". Later missions reflect a similar willingness to 'ease off' a bit when the player pilots a new craft, first attacks a capital ship, etc..

It's an approach that works. Wing Commander combat savvy is 20% quick-sticking coordination and 80% 'smarts'. Mission after mission, both areas get a good workout so that getting used to your guns, missiles, radar, status displays, Nav Computer, etc. and adjusting to the minor differences in cockpits is nearly automatic. Knowing when to 'hold' or 'release' your wingmen, how to draw out and cut up enemy fighters protecting a Fralthra class cruiser, finding the best approach for a torpedo run— that is, developing tactical 'smarts'— takes a bit longer. By the end of WC2 you're an expert in 'fighting your ship', deadly in a dogfight, and 'not too shabby' at tactics— guaranteed, since you'll never get to the end of WC2 otherwise. By the end of SO1 you're just plain deadly, the best Wing Commander in the Fleet!.

Wing Commander II combat aims for realism; and, at any given moment, what you see is very convincing. Whether you select cockpit, left or right turret, rear turret, "tactical", or some other view,

the 3-D images of ships, missiles, (asteroids, mines, etc.) are continually updated to present the right size and angle vis-a-vis the player.

Combat sound effects rate "pretty good"; there's plenty of room for more variety and 'sock'. The same applies to weapons visuals. Your silvery mass balls, lavender plasma bolts, etc. sail into targets and 'splash' the shields; but you never actually see a hit munch a wing or hole the fuselage. (Ship explosions, however, are quite gratifying— lots of churning fire and debris.) Throughout, insults traded with enemy pilots add to the fun; and, via an SO1 editor utility, you can now create your own taunts. As for the music score skillfully threaded through both combat and story segments, it rates an unqualified "excellent"!

All in all, 'leading edge' stuff; and that usually means it's time for a speed check! Origin recommends "16MHz minimum"; odd, because the number is bound to scare off some 12MHz machine owners. Yet, 16MHz is so far from being fast enough to show WC combat at its best that 4MHz one way or the other scarcely matters. Baywoof, for example, noticed some slowness on his 25MHz '386.

Mainly, there are two effects: Objects such as enemy ships may move less smoothly; and, you may notice delays in control response. Displays for the storyline, take-offs and landings, etc. are not notably affected because these are all in-the-can animated sequences. Most combat images are generated on-the-fly. The more active objects displayed (like shups, missiles, mines, and asteroids) and the closer they approach, the longer it takes to process the images. Of the testers who have given Wing Commander II a try, only one reported "no problem" with speed in combat. He was running the game on a 33MHz '486 system. Yet, I enjoyed the battles playing on a 12MHz '286! Clearly, there is a BIG difference between speed requirements for obtaining "maximum" and "good enough" realism.

All large games have bugs. With one exception, the few you encounter in Wing Commander II and its first supplement are minor. For example, on one occasion, an overlay is misplaced in a story segment. On another, I got multiple-exposure images of my craft during a landing. The exception 'hangs' the game just before landing. In a total of something like a hundred missions (including retries after defeats) I ran into this bug just three or four times. On one encounter, I happened to add "F1" to my key-press recovery attempts and, voila! instead of having to re-boot, I was back in the game! Origin believes I may have gotten a diskette with a badly copied file. (They mailed a free set of replacement diskettes.) Perhaps. In any case, I continued play with the original installation and never encountered the bug again.

The biggest "bug" is not really a bug at all; it's a design flaw. For some reason, WC2 supports both VGA and 16-color EGA; this, despite the fact that VGA has been 'standard stuff' for some time. (Besides, playing the game in anything but 256-color VGA is unthinkable.) As a result, installation must both unpack and process files to load the game to hard disk; an ordeal spanning eight 3.5" HD diskettes and well over two hours! Granted, this is a large program, about 15MB plus another one or two MB for the SO1 supplement; but, there is no excuse for such an un-fun start to such a fine gaming experience. (Incidentally, when installing you do not select the "Save Space", unpack-during-play option— play is far too slow.) Determined to avoid a repetition in case reinstallation should be required, I saved the hard disk files, using PKWare's "PKZip" utility to do the compression, and got everything onto seven diskettes. Loading the game from these diskettes (including unpacking) takes about fifteen minutes!

Wing Commander II is unique. An objective outsider might easily walk by a trade show both, watch someone play for a while, and walk away certain that he or she had the game 'pegged': "Obviously, the attraction is the combat arcade. The music and story stuff is a nice frill. Too bad there's no scoring or High Scores roster." You will know better. The combat 'works' because the scenario 'works' and vice-versa. A High Scores roster would look silly! Your WC2 and SO1 encounters are realistic, addictive, fun because combat graphics and sound 'get the job done', the challenge never falters, AND because you must win. Even on the three or four occasions when defeats can force five, ten, or more attempts— when 'the joy of combat' is wearing pretty thin— you'll say "There's NO WAY to win this thing!" and then go back for more. Earth, peace-loving Kilrathi rebels, your shipmates, and Jeanette are counting on you. (And, that Prince Thakrath guy is REALLY arrogant!) Naturally, winning the tough ones is the most fun of all.

In a recent radio talk show appearance (Doug Johnson Show, KPRC) I mentioned the new Wing Commander as an action entertainment 'good bet'. "Well," Doug interjected, "I can tell by the way your eyes light up, that you like it. It must be a pretty great game." Not a bad summary. Get into Wing Commander II. Expect a software masterpiece, hours of challenging fun, and genuine involvement in "a pretty great game"!

Quest for Clues III

★★★★

$24.99, 8.5" x 11" softcover, 198 pages

Origin

Need help with "Dragon Wars", "Circuit's Edge", "Starflight" 1 or 2? What about "Usurper'", "Windwalker", or "Space Quest III"? If it's a major '88-'89 vintage adventure, the odds are the solution is in Origin's Quest for Clues III.

Each of the forty entries begins with a brief overview/mini-review; but, from then on, format is tailored to fit the game. Sometimes, as in "Loom"'s section, you'll get virtually a 'walkthrough'; while, for "Wizardry V'", the emphasis is upon puzzle-busting; and "'Azure Bonds" coverage highlights key events and encounters. You can expect clear, easy-reading descriptions throughout with, as usual, simple alphabet-shift encoding of critical words to avoid inadvertent spoilage of puzzle challenges. (The code— b = a, ..., a = z— is listed in the book.)

As owners of earlier QFC's can attest, coverage goes far beyond mere clues. If maps will help, it's maps you get, complete and thoroughly annotated. The same goes for charts and tables, whether it's "Space Rogue"'s Places, People, and Items, Spells and ingredients for "Keef'", or "'Magic Candle" Teleporter Combinations and Chants. QFC 'delivers the goods' and, with illustrations plus good use of font sizing and grey-bar highlighting, does it in a handsome, easy-on-the-eyes package.

The book does not claim to supplant game manuals. It doesn't; though it will, often, prove a valuable addendum even when you are not especially looking for hints. For example, just keeping track of "Neuromancer"'s Database locations, codes, passwords, and contents is a major chore. QFC organizes everything in a handy one-page table and adds three pages of charts listing AI strength levels and weaknesses, ICE Breaker ratings with locations, and functions of Chips, Objects, and other Software!

As a game buying guide, QFC rates, at best, fair. One intro/review observes that some "'Azure Bonds" encounters "take hours"— maybe, if you take a break for lunch— and recommends it for "adventurers who love wargames"!? While most of the sketches are more accurate, criticism is uniformly subdued or, simply, absent. You can get some ideas for games to check out; but, that's about it. Remember, the book is published by a major game producer. Then, too, do you really expect to read "this game is a nearly unplayable, boring turkey" just before several pages are devoted to the game's maps and hints?

No problem. The BIG reason you buy QFC is "the clues"; and, as claimed, you get what amounts to a whole collection of Clue Books. Sometimes— SSI is a good example— the maker's books supply notably more detail. Sometimes, as with "Wizardry V'", no Clue Book is available! Whatever the alternatives, you can count upon QFC for more than enough coverage to get the job done. Even if you need/want help with just a few of the forty games, Quest for Clues III ranks among the best bargains in adventure gaming.

Fast Frames, Updates, etc.

Apple Materials

Recently, I called Jessa Vartanian, Apple's APDA contact, to find out about new support products. She quickly returned a copy of the current "APDA Tools Catalog" just crammed with Apple and third-party Macintosh products (e.g. Data Access Language 1.3, SCSI Development Package, CD-ROM Developer's Lab, ...). Obviously, if you are a Mac user, APDA has a lot to offer. There were also some IIgs listings; so, I asked for review copies of two GS/OS references. Tessa forwarded my request to Apple's PR ace, Keri Walker, who forwarded it to Tom Weishaar at Resource Central!

Tom mailed the references along with a cover letter: "... As you may have heard, Resource Central is now the official distributor of Apple's Apple II materials that were formerly sold by APDA. In addition, we have virtually every book in print about the Apple II (and we have the last remaining stock of books that will be 'out of print' when our stocks are gone).

"We think Apple II's will be used in America's education system for years to come. They are classic machines and will be supported by users much like classic automobiles. We'll be here to help the Apple II community in any way we can. ..."

IIgs Disk Fixer

Speaking of "classics", our trusty "Woz" edition IIgs still gets plenty of use. However, it developed a severe 'fuel line problem' some weeks back, when one of its two 3.5" drive's refused to read diskettes. This time, popping in a Head Cleaner diskette did not restore functioning. After five years of faithful service, the heads were gone. (I temporarily swapped in the head assembly from the 'good' drive to verify the problem. Sure enough, now the 'bad' drive worked fine.)

Checking around to obtain the replacement revealed a challenge of greater magnitude than first anticipated. Evidently, nobody sells Apple IIgs (Sony) 3.5" head assemblies, certainly not to individuals. Apple's own "National Parts Supplier" tried to help; but it soon became clear that the notion of selling Apple parts to Apple users remains dangerously novel, not the sort of thing you just spring on someone over the phone. The usual "repair" for any IIgs 3.5" unit consists of replacing the entire drive assembly. Around $180 - $200 seems to be a typical "good deal" for the job.

It is fortunate that I dumped the problem onto the local "Club Apple" BBS. Within a day, the sysop (Glynne Tolar) came back with two shops he thought might fix the drive: one for $104, another for about $75. I called the second place. Connect It quoted a price of $79; and, ZAPPO, the drive was in the mail. Within two weeks it was back, checked out 'good as new'; and our IIgs is once more 'hitting on all fours'!

How come Connect It got the job done for so much less? I called again and spoke with a repairman. He explained that their approach is to try to fix only what is wrong. (Gosh! What a neat idea. No wonder their prices were lower; they were practically cheating!) $79 is the usual price for replacing the head assembly. (By the way, when your 3.5" drive bombs, the problem is virtually certain to be dirty or worn out heads.) When I asked where/how they got the head assemblies, the CI spokesman declined to be specific. He said they buy them by the case; but, that, sometimes, they are kind of hard to find.

Tunnels and Trolls ★★

Okay. On the one hand I have, generally, enjoyed New World's Ultima-style swords & sorcery adventure ($49.95 for CGA-EGA 640K PC). Set on the 64 x 96 Dragon Continent, seas, and isles, T&T sends your dauntless foursome on a quest to squash an evil wizard, free an entombed arch mage, and topple a demoness named Lerotra'hh. The game offers an unusually rich mix of mini-quests and personages spanning several cities, castles, dungeons, and mines. In 640 x 400 16-color EGA, your party shows as a single hero figure on the partial-perspective, top-down display used for city and maze explorations. When moving between cities, exploring the countryside, etc. you guide a dot on the terrain map. Fully a third of the screen is reserved for displaying the well-written, unusually verbose prose passages which accompany the numerous special situations you encounter. Other notable features include full self-mapping, quick

Save/Restore, decent monster variety, fairly good weapons system, some clever ideas for magical artifacts, adequate PC sound, an attractive manual, and a colorful fold-out map.

On the other hand, the game is beset by decidedly alien-looking hero pic artwork, numerous potions and artifacts with vaguely defined properties, a magic system padded with many spells you will never use, cramped single-screen tactical combat, and cumbersome terrain map movement. To which you can add swarms of non-fatal bugs. For instance, you will encounter occasional text over-scrolling which hides useful information; the Red Orc mini-quest which, if you first meet the chief after rescuing his daughter, permits ripping off endless quantities of gold; a glitch which delays journeys past Castle Overkill, even after it's liberated; items incorrectly identified as "useless"; and a good "Dreams of the Dragon" clue book ($19.95) with numbered references to key locations but no numbers on the maps!?

Aside from boosting Experience and character attributes, much of what you can accomplish (good deeds, rescues, missions, etc.) will have no direct bearing on your quest to rid the lands of Lerotra'hh. Yet, some achievements are critical; and telling one from the other is not always so simple. This 'looseness', and a relatively modest tactical challenge mean you have to enjoy mini-quest do-gooding, note-taking, and exploration a lot to get the most from Tunnels & Trolls.

Tunnels and Trolls Tavern Talk

Rumor-wise, your stop at "The Tunneling Troll" has produced surprisingly slim pickings. Everyone seems content with small talk and swilling the fine Rock Dwarf beer. You're on the way out (to try things over at "The Dragon's Dream") when, at last, you overhear something of interest...

"All you new guys make the same assumption; but, around here, staffs are for spell casting, wands are spelled; least-wise, the good ones are. You use 'em like swords. Now, this one (whoosh, whoosh) is your standard two-handed Wizard's Wand. THIS little jewel I call 'Black Beauty' (whish), a genuine one-handed Death Wand. The Wizard' is good for a solid hit in a pinch; but one tap from 'Beauty here plants 'em where the sun don't shine. Now, a young magic user like yourself needs ...

"... maze in Castle Overkill isn't fatal, it's just that any wrong step sends you back to the start— darn, sure wish I had a pencil. Look, just remember these directions: E, that's "east", gets you onto the first square in the maze room. Then you move on toward the east wall S-E-E-N-E-E-E-E-S-E-S-S. Next, you go westward W-W-N-W-W-S-S-W-S-S. Then, back east E-E-N-E-E-S-S-E-S and, viola, you're in the treasure room!

"Yeah, that was Gorbash. He, Igmo, and the rest of his bunch were up around Level 12 when Gorbash figures 'Why not head north and clean up on the dire wolves?' Damn things just kept coming; and Experience kept climbing! Igmo claims everybody gained four or five Levels!!"

"Ain't nobody ever gone wrong buyin' a round for old Yurdlin (slurp!). Now, you're a new bunch, itchin' for questin' but short of 'das geld', right? The place to go is the Gull sewers; the one to see is the Wraith. He plays dice hi/lo and pays off in gems (and Experience)! You can play game after game and build up quite a tidy bankroll. 'Course, losin' is fatal; but, a savvy bunch like yours should be able to figure a way around that. (slurp, glub) Later, once you've got to where you can swat wolves like flies, head for the Arena in Khazan and sign everyone up for the 10-combats contract. If you don't walk out with a couple hundred thousand denars, my name's not ..."

Platter Plague?

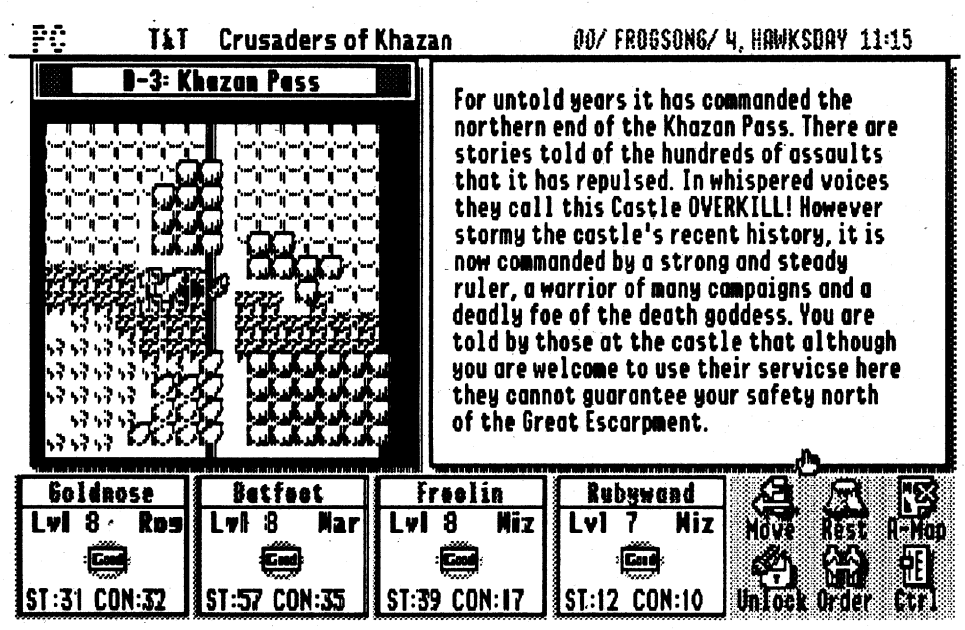
"Mean Time Before Failure" numbers for the average hard disk would seem to guaranteed useful operation for several years. Why, then, do I keep running into users with failing 2-3 year-old drives?

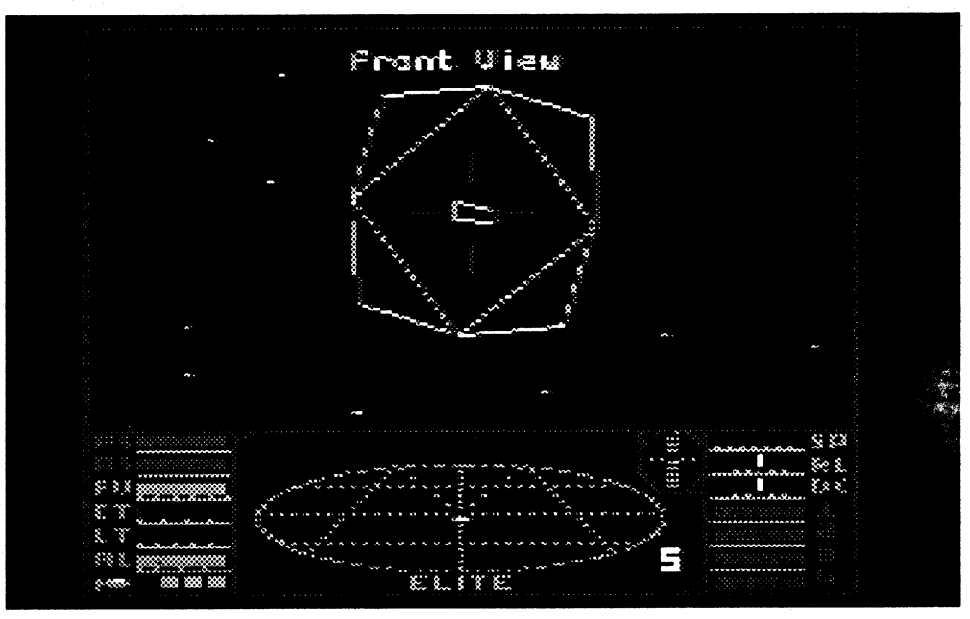
The most frequent complaints are motor-related, beginning with notably more noisy operation. Our own 40MB MFM drive has begun to produce squeals after barely 18 months. Annoying, but we're still better off than users who flip on the power and find that the drive simply fails to start. (i.e. "It don't work!") Apparently, temperature makes a difference— good to know if you need to back-up your stuck drive prior to replacement. Baywoof has noticed that his failing PC drive (60MB RLL) is more likely to 'click-on' when the room is warm. The same was true of the hard drive he used with his IIgs. (Some users report that giving the case a good solid tap helps.) Other complaints include crashes (e.g. a head becomes glued to a platter) and "forgetfulness". The latter may produce program bombing, or 'merely' excessively time-consuming re-reads. Some blocks are weak, but 'work' enough of the time to escape being marked "Bad" during normal checks.

Logically, MBTF numbers for the last (pre-IDE) generation of small computer hard disks had to be estimates. It's beginning to look like these were overly optimistic. If your 'old' drive starts to act funny, don't be too shocked. Get ready to invest in a new unit and, probably, a new controller board.

Next

SSI's Pools of Darkness is guaranteed PLUS exclusive hints for the game's ultra-difficult Final Showdown battle! Then, too, ...





Vendors

APDA: Apple Computer
MS 75-2C
20525 Mariani Avenue
Cupertino, CA 95014
atten: Jessa Vartanian, (408-974-0091, orders: 800-282-2732)

APPLE COMPUTER
MS 48-I
20525 Mariani Avenue
Cupertino, CA 95014
atten: Keri Walker (408-974-2042)

CLUB APPLE
P.O. Box 5338
Pasadena, TX 77508-5338
Attn: Glynne Tolar (713-476-9998)

CONNECT IT
2026 W. Pioneer Parkway, Suite B-7
Arlington, TX 76013
Attn: PR/Mktg. (817-461-9400)

DOUG JOHNSON SHOW
KPRC Radio
P.O. Box 2222
Houston, TX 77252
Attn: Doug Johnson

ELECTRONIC ARTS
1450 Fashion Island Blvd.
San Mateo, CA 94404
Attn: Marci Galea (415-571-7171/ orders 800-245-4525)

NEW WORLD COMPUTING
20301 Ventura Blvd., Suite 200
Woodland Hills, CA 91364
Attn: Scott McDaniel (818-999-0607)
dist. Electronic Arts

ORIGIN SYSTEMS
110 Wild Basin Road, Suite 330
Austin, TX 78746
Attn: Wayne Baker (800-999-4939)

PKWARE
7545 North Port Washington Road, Suite 205
Glendale WI 53217
Attn: PR/Mktg. (414-352-3815)

RESOURCE CENTRAL
P.O. Box 11250
Overland Park, KS 66207
Attn: Tom Weishaar (913-469-6502)

STRATEGIC SIMULATIONS INC.
675 Almandor Ave
Sunnyvale, CA 94086
Attn: Kathleen Watson (408-737-6800)
dist Electronic Arts

Michael A. Horton — WA

Does anyone out there have the instructions for KORONIS RIFT? My set got lost one of the times I moved and I really need some instructions.

An ELITE Craft ... without NMI

Advanced Playing Technique for...

Elite

?

When I saw the APT for ELITE in issue #70 I was happy, but as I read it my happiness began to fade. I soon found out this APT required some way to enter the monitor while the program is executing. I have no way to do this so I couldn't use the APT the way it was, but it did give me a nice place (and some good incentive) to start.

Since Jeff Hurlburt didn't list any of the code at the locations he was modifying I had to look for myself. I got ELITE booted up and running. Well, I have an Apple IIe and no way to enter the monitor, so I have to reboot and then reset the computer. I do this by pressing the following keys: control, open apple, reset. I then release the reset key and then the other two keys. Then I must press the control and reset keys to reset the computer. To some this may seem cumbersome, but if you don't have NMI capability it sure gives you a way to look at most of the program's code.

Warning: Using the above procedure will change the values of some memory locations. So the program may or may not be restartable from memory. You can always reboot the program.

Note: The program names have a ctrl E character buried in them. The ctrl E is not printed to the screen, (it is invisible), but it must be typed to load the programs.

The boot program (the first program that is executed after DOS is in memory) is called E ctrl-E LITE. It is written in BASIC. It runs another program called E ctrl-E LA. This program executes and then returns control to ELITE which then runs a program called S ctrl-E EC3. I have stopped the program after ELA has run. I put the screen in text mode (TEXT), dropped into the monitor (CALL -151) and checked for the code I found at the locations mentioned by Jeff Hurlburt. The code I found when the program was running was not there, so SEC3 must create it. Following this thought, I loaded SEC3 into memory (BLOAD S ctrl-E EC3) to look at it. SEC3 starts in memory at $2000. Here is what I saw:

2000- 20 D7 24 JSR $24D7

```
2003- 20 7C 24 JSR $247C
2006- 20 DC 23 JSR $23DC
2009- 20 9B 24 JSR $249B
200C- 20 5D 24 JSR $245D
200F- 20 DC 23 JSR $23DC
2012- 4C 00 40 JMP $4000
```

The jump to $4000 look like a good place to drop into the monitor from. So I changed the jump to $4000 to jump to the monitor.

2013:59 FF

I started SEC3 running.

2000G

It loaded some stuff into memory then dropped into the monitor. I look at for the code again but it still wasn't there, so SEC3 must need to run a little further. I began to look at $4000. It contained a jump to $456D.

```
4000- 4C 6D 45 JMP $456D
```

I really didn't see anything interesting until $4592.

```
4592- A9 30    LDA #$30
4594- 8D 42 83 STA $8342
4597- EA       NOP
4598- EA       NOP
4599- 20 0F 96 JSR $960F
459C- 4C B9 7EJMP $7EB9
```

The jump at $459C looked like another good place to stop the program. So I change it to drop into the monitor.

459D:59 FF

I then started SEC3 back up.

4000G

It dropped into the monitor after a flash. I then looked for the code and this time I found it. Now comes a slight problem, the code at $456D is not there when SEC3 is loaded but gets there when it is executed. I like to write APTs that are "clean" so the APT doesn't cause problems with the program. If I re-write something from a program I try to have it mimic the original code.

Now all I had to do was write the APT. First, I rewrote the BASIC portion of ELITE. The BASIC portion is the same except it loads in CHEAT.CODE and SEC3 The BASIC program then modifies the jump at $2012 so that it will jump to $0100 (POKE 8212,1). Then it runs SEC3 (CALL 8192). I like using the very bottom of the stack because very few programs use it and it usually is safe.

I wrote a program that modifies the jump at $459C. It makes SEC3 jump back down to the rest of the APT once SEC3 puts the code we looked for in place. The program then patches in those neat APTs.

Step-by-step

1. Format a slave disk with a boot program name "ELITE".

INIT ELITE

2. Type in the 1st program and save it as "ELITE".

```
10 TEXT : HOME
20 PRINT TAB(5) "EEEEE◊◊L◊◊◊
  ◊◊◊III◊◊TTTTT◊◊EEEEE"
30 PRINT TAB(5) "E◊◊◊◊◊L◊◊◊◊
  ◊◊◊I◊◊◊◊T◊◊◊◊E"
40 PRINT TAB(5) "EEEE◊◊◊L◊◊◊
  ◊◊◊◊I◊◊◊◊T◊◊◊◊EEEE"
50 PRINT TAB(5) "E◊◊◊◊◊L◊◊◊◊
  ◊◊◊I◊◊◊◊T◊◊◊◊E"
60 PRINT TAB(5) "EEEEE◊◊LLLL
  L◊◊III◊◊◊◊T◊◊◊◊EEEEE"
70 PRINT : PRINT : PRINT
80 PRINT TAB(10) "1...CHEAT
  ◊VERSION"
90 PRINT : PRINT TAB( 10)
  "2...NORMAL◊VERSION"
100 PRINT : PRINT : PRINT
  TAB(10) "CHOICE:" ;
110 FLASH : PRINT "◊" :
  NORMAL
120 POKE - 16368,0
130 A = PEEK ( - 16384)
140 IF A < 128 THEN 130
150 IF A < > 177 AND A < >
  178 THEN 120
160 VTAB 14: HTAB 17: PRINT
  CHR$ (A - 128)
170 POKE - 16368,0
180 IF A = 177 THEN PRINT
  CHR$ (4) "RUN◊CHEAT◊ELITE"
190 IF A = 178 THEN PRINT
  CHR$ (4) "RUN◊STANDARD
  ◊ELITE"
```

SAVE ELITE

Checksums

```
10-$FB33    80-$AB71   150-$4A15
20-$038F    90-$7FAC   160-$1FD0
30-$5566   100-$2E69   170-$E926
40-$825D   110-$0309   180-$24CC
50-$7709   120-$6C45   190-$203A
60-$557B   130-$A3A7
70-$93DF   140-$1CEA
```

3. Type in the 2nd program and save it as "STANDARD.ELITE".

```
10 PRINT CHR$ (4) "BRUN◊E"
  CHR$ (5) "LA"
20 PRINT CHR$ (4) "BRUN◊S"
  CHR$ (5) "EC3"
```

SAVE STANDARD.ELITE

Checksums

```
10-$45DF    20-$EA15
```

4. Type in the 3rd program and save it as "CHEAT.ELITE".

```
10 PRINT CHR$ (4) "BRUN◊E"
  CHR$ (5) "LA"
20 PRINT CHR$ (4) "BLOAD◊
  CHEAT.CODE"
30 PRINT CHR$ (4) "BLOAD◊S"
  CHR$ (5) "EC3"
40 POKE 8212,1: CALL 8192
```

SAVE CHEAT.CODE

Checksums

```
10-$45DF    30-$1E48
20-$7C4C    40-$CA3F
```

5. Enter the monitor, type the binary program and save it.

CALL-151

```
0100:48 A9 0F 8D 9D 45 A9 01  $F5DE
0108:8D 9E 45 68 4C 00 40 48  $3DF1
0110:98 48 A9 FF 8D 0E 6F A0  $DB12
0118:00 B9 30 01 F0 06 99 AF  $A6B5
0120:6B C8 D0 F5 8D 84 5A 8D  $18D9
0128:D9 7B 68 A8 68 4C B9 7E  $7DE3
0130:A2 FF 8E BC 02 E8 EA EA  $C8E5
0138:EA 8E C3 02 00           $D8EC
```

BSAVE CHEAT.CODE,A$0100,L$3D

6. Copy the files on the deprotected ELITE disk to the slave disk except for the program named "E ctrl-E LITE". (Remember: the ctrl E is invisible.)

Now we all can play ELITE with the added fuel and unlimited "G-hops" or just play normally. Either way it is a great game. So get out there and rack up those "kills". My current rating as of writing this is DEADLY. If anyone out there is rated as ELITE, please let us Computist readers know.

The following is an explanation of the APT's source code.

```
        .OR  $0100   Object code is to start at
                     $0100
        .TF  CHEAT.CODE Send object code to
                     a Target File called
                     CHEAT.CODE
START   PHA          push Acc. value on the
                     stack so we can get it back
                     later
        LDA #INSERT.PATCH  get location low
                     byte of INSERT.PATCH
        STA $459D    store it in Elite so that Elite
                     will run our patcher
        LDA /INSERT.PATCH  get location high
                     byte of INSERT.PATCH
        STA $459E    store it in Elite so that Elite
                     will run our patcher
        PLA          get the old Acc. value off
                     the stack
        JMP $4000    run some more of Elite
INSERT.PATCH
        PHA          push Acc. value on the
                     stack so we can get it back
                     later
        TYA          transfer Y-register to Acc.
        PHA          push Acc. value on the
                     stack, we just put what was
                     in the Y-register on the
                     stack
        LDA #$FF     load Acc. with 255 decimal
        STA $6F0E    make light-year fuel when
                     bought equal 25.5, notice
                     the similarity with 255
        LDY #$00     set index at 0
.2      LDA NEW.CODE,Y get a byte of the
                     patch
        BEQ .1       are we at the end of the
                     patch (00) if we are then go
                     to .1
        STA $6BAF,Y  store the byte in Elite
        INY          increment the index
        BNE .2       branch back to .2 if index is
                     not 0
.1      PLA          get the value off the stack
                     and put it in Acc.
        TAY          transfer Acc. to Y-register,
                     we just got the old Y-
                     register back
        PLA          get the old Acc. value off
                     the stack
        JMP $7EB9    run the rest of Elite
NEW.CODE
        LDX #$FF
        STX $02BC
        INX
        NOP
        NOP
        NOP
        STX $02C3
        .HS 00
```

Notes on Hacker II

Here is something for those of you who play HACKER II. When you reach the "LOGON PLEASE:" prompt, type in one of the following phrases.

TITLE H2SC

This takes you to the title page of the game.

COVER H2SC

This takes you to the "Rolling Stone" magazine cover as if you completed the game.

DEMO H2SC

This is my personal favorite, what it does is it gives you unlimited MRUs. You start with 3 MRUs, when your first MRU gets destroyed you will now have 2 MRUs for the rest of eternity (or until you turn off the game).

And don't forget what to type when you get in front of the filing cabinets to gather intelligence ("ROA").

Advanced Playing Technique for...

Eidolon

?

Now another APT. Don't worry I won't go into detail on this one because it took way to long to develop. Here it is:

Stops EIDOLON clock:

Trk	Sct	Byte	From	To
1D	E	C5	03 4C B5	00 4C 2C

No energy drop:

Trk	Sct	Byte	From	To
1C	8	E5	49 FF 38 6D	EA EA EA AD

The last dragon is hard to kill (7 heads) and personally, I think the ending of this game is stupid, but the last maze looks really neat.

Playing Tip for...

Gemstone Healer

?

Here is some help for Gemstone Healer for those who need it. Each one of the following five sets heals one part of the gemstone. I hope this will help some people complete the game.

Set #1

South
Earth
Hand
The Five Plus
Positive

Set #2

North
Fire
Eye
The Five Square
Negative

Set #3

Orientation
Air
Foot
The Five Circle
Emitter

Set #4

West
Water
Ear
The Five X
Collector

Set #5

East
Elements
Soul
The Five Unite
Transformer

Super Boulderdash APT Explanation

I booted up Super Boulderdash (Front side, Boulderdash I) and then rebooted and reset the computer. I then entered the monitor.

I knew from playing the game that when you press the 'ESC' key the game kills your current Rockford and start the cave over again. This is used if you get trapped and don't want to sit there and wait for time to expire. I figured this would be a good place to start since the game changes the number of Rockfords you have. So I started to see how they were looking at the keyboard.

C000<0.BFFFS

The location to look at the keyboard is $C000. What the above command says is search for C000 from $0000 to $BFFF. $0-$BFFF is the lower 48K RAM. Below is part of the result.

```
0042-
0044-
006F-
0073-
0CC4-
1565-
1627-
1701-
170C-
174A-
```

I don't look at the zero page locations (00xx-) if there are others listed because whatever I am looking for is most often not there. Also never bother looking at 0042 and 0044 because this is where the monitor stores what you are looking for. I started looking at $1558.

1558L

I always start looking several bytes ahead of the listed location to get a better look at the code. Here is some of what I saw.

```
1558-8D A9 F0 STA $F0A9
155B-20 A8 FC JSR $FCA8
155E-CA       INX
```

```
155F-10 03      BPL $1564
1561-4C C9 16   JMP $16C9
1564-AD 00 C0   LDA $C000 ****
1567-10 ED      BPL $1556
1569-8D 10 C0   STA $C010
156C-C9 C1      CMP #$C1
156E-F0 0B      BEQ $157B
1570-C9 CA      CMP #$CA
1572-D0 E2      BNE $1556
```

At $1564 the accumulator is being loaded from the keyboard location. So most likely the program will compare the value in the accumulator to the value of the 'ESC' key ($9B). The command to do this would look like.

```
xxxx- C9 9B     CMP #$9B
```

To look for this in memory we will use the monitor search command. The search command will find a 1 or 2-byte sequence in memory. The value we must search for is 9BC9. The 9B must come first because it is stored in the higher memory location(most significant byte). The C9 comes last because it is stored in the lower memory location(least significant byte).

9BC9<0.BFFFS

```
0042-
0044-
1724-
```

I started looking at memory at $1718.

1718L

```
1718-92         ???
1719-D0 09      BNE $1724
171B-A9 01      LDA #$01
171D-85 E8      STA $E8
171F-85 EE      STA $EE
1721-4C 70 18   JMP $1870
1724-C9 9B      CMP #$9B    *****
1726-D0 03      BNE $172B
1728-4C 18 18   JMP $1818
172B-C9 A0      CMP #$A0
172D-D0 B8      BNE $16E7   Branch to beginning
                            of the keyboard
                            routine
```

Apparently if the accumulator is equal to $9B then the computer will jump to $1818. I then looked at $1818.

1818L

```
1818-A6 26      LDX $26     Escape routine
181A-D6 28      DEC $28,X   Decrement number of
                            Rockfords
181C-A5 07      LDA $07
181E-D0 1F      BNE $183F
1820-A6 26      LDX $26     Advance to next cave
                            routine
1822-A5 27      LDA $27
1824-10 03      BPL $1829
1826-4C 00 08   JMP $0800   Jump to clear screen
                            and reload title
                            screens
```

At $181A, there is a decrement command (subtract 1 from something). This happens to decrement the number of Rockfords that you have. Looking at the code, I was curious as to why the accumulator was being loaded from $07 and if it wasn't 0 then it branched to $183F. So I scanned the disk for A6 26 D6 28. I found it at Track $0E Sector $07. I then changed the 1F to a 00 so that if it branched it would go to the next instruction. After writing the sector back to the disk, I booted up Boulderdash I. When I pushed the 'ESC' key I lost a Rockford and ADVANCED TO THE NEXT CAVE! Wow! Now that is a surprise!

I thought maybe I could write something to add Rockfords and/or advance to the next cave. Only one problem, where to put it in memory? Well I looked through the program and found a great place and something I think is stupid. They (the authors) wrote a hi-res page clear routine that clears both hi-res pages but the routines take up 205 bytes. It stretches from $9FE to $ACA. I'm sorry, but to me that is way to long for a screen clear routine. Sure it runs a little faster than mine but who notices a few milliseconds?! So now I have to write a new screen clear routine and I will have the space I need for an APT. The screen clear routine below modifies itself sightly to conserve space and not use any zero page locations. The hi-res pages are from $2000-$3FFF for page 1 and $4000-$5FFF for page 2.

```
09FE:A9 20      LDA #$20    Load the MSB(Most
                            significant byte) of the
                            address of Page 1
0A00:8D 0E 0A   STA $0A0E   Store it in xx
0A03:A9 40      LDA #$40    Load the MSB of the
                            address of Page 2
0A05:8D 11 0A   STA $0A11   Store it in yy
0A08:A0 00      LDY #$00    Load the index with 0
0A0A:A9 00      LDA #$00    Load the accumulator
                            with 0 the value we
                            are clearing the
                            screen to.
0A0C:99 00 xx   STA $xx00,Y Store it in page 1
0A0F:99 00 yy   STA $yy00,Y Store it in page 2
0A12:C8         INY         Increment the index
0A13:D0 F7      BNE $0A0C   Branch if the index is
                            not 0 to A0C
0A15:EE 0E 0A   INC $0A0E   Increment xx
0A18:EE 11 0A   INC $0A11   Increment yy
0A1B:AD 11 0A   LDA $0A11   Load the accumulator
                            from yy
0A1E:C9 60      CMP #$60    see if it has reached
                            the end of Page 2
0A20:D0 E8      BNE $0A0A   if it hasn't then
                            branch to A0A
0A22:AD 54 C0   LDA $C054   turn on page 1
0A25:60         RTS         return to caller
```

Now we need to write an APT and place it after the screen clear routine. Well I wanted something I could implement simply and select between the APTs or have both at once. So I changed the 'ESC' key jump to jump to my APT code.

Was:

```
1728-4C 18 18   JMP $1818
```

Now:

```
1728-4C 26 0A   JMP $0A26
```

My idea of two good APTs are:

Add more Rockfords whenever you like

Advance to the next cave

Well the joystick happens to have two buttons, what if we use those to select which APT to execute.

```
0A26:AD 61 C0   LDA $C061   Get Button 0
0A29:10 04      BPL $0A2F   If not pressed goto
                            A2F
0A2B:A6 26      LDX $26     What player number
0A2D:F6 28      INC $28,X   Add 1 to the
                            Rockfords
0A2F:AD 62 C0   LDA $C062   Get Button 1
0A32:10 03      BPL $0A37   If not pressed branch
                            to A37
0A34:4C 20 18   JMP $1820   Jump to advance a
                            cave routine
0A37:AD 61 C0   LDA $C061   Get Button 0
0A3A:10 03      BPL $0A3F   If not pressed branch
                            to A3F
0A3C:4C E7 16   JMP $16E7   Jump to beginning of
                            the keyboard routine
0A3F:4C 18 18   JMP $1818   Jump to normal ESC
                            routine
```

In order to activate this APT you must press and hold the correct joystick button then press the 'ESC' key. If button 0 is being pressed, it will add 1 to the Rockfords. If button 1 is being pressed, it will advance to the next cave. If no button is being pressed, it will jump to the normal ESC routine (kill Rockford and restart cave). The back side is just slightly different but the logic is still the same. I hope this helps someone out there.

A fix for "Putting... Super Boulder Dash on a hard disk"

I liked Marc Batchelor's article about changing Super Boulder Dash to file form in issue #53. It was nice but there were some problems with it. Besides the ones mentioned in the article, you never see the title screens again. Well I have fixed all those problems and made the file shorter. Huh?! Did you say you fixed the problems and made it shorter at the same time? Yes and no, the saved file is shorter but when unpacked into memory it takes up more room. I would like to thank Marc for laying the ground work for this article.

Step-by-step

1. Create a slave disk with no hello program.

INIT HELLO
DELETE HELLO

2. Type in the programs at the end of this article and save them on the slave disk.
3. Copy the deprotected Super Boulder Dash (issue #38) disk to a new disk.
4. Scan the disk for 4C DD 14 and change them to 4C 59 FF.
5. Boot the disk (Super Boulder Dash copy).

6 ctrl-P

6. When the program drops into the monitor, enter the modification and then move memory from $800-$8FF to $4000.

809:20 43 0A 4C DD 14
4000<800.8FFM

7. Boot the slave disk you created earlier.
8. Enter the monitor.

CALL -151

9. Move the memory back to $800 and save it.

800<4000.40FFM
BSAVE MEMORY.800.I,A$800,L$100

10. Load and execute CREATE.BOTTOM.PART

BLOAD CREATE.BOTTOM.PART
801G

11. Save the resulting memory.

BSAVE BOTTOM.PART,A$5970,L$690

12. Boot the copy of Super Boulder Dash.
13. Move memory from $8100-$996F to $2100.

2100<8100.996FM

14. Boot the slave disk.
15 Enter the monitor.
16. Load "MEMORY", BOTTOM .PART, APT files into memory.

BLOAD MEMORY.800.I
BLOAD MEMORY.2000.I
BLOAD MEMORY.A42.I
BLOAD BOTTOM.PART,A$3970
BLOAD APT.I

17. Put a jump at the beginning of the program to jump to our code.

7FD:4C 00 20

18. If you wish to have the APTs listed in issue #50 type in the following modification. If not, just skip this step.

1729:26 0A

19. Save the game.

BSAVE BOULDER DASH I,A$7FD,L$7903

You have Boulder Dash I in file form and it takes up 21 sectors less.

Boulder Dash II is very similar. Here are the steps that are different.

4. Scan the disk for 4C 53 16 and change them to 4C 59 FF.
6. When the program drops into the monitor, enter the modification and then move memory from $800-$8FF to $4000.

809:20 7D 0A 4C 53 16
4000<800.8FFM

11. Save the resulting memory.

BSAVE BOTTOM.PART.A,A$5970,L$490
BSAVE BOTTOM.PART.B,A$5E00,L$200

13. Move memory from $8600-$9DFF to $2600.

2600<8600.9DFFM

16. Load "MEMORY", BOTTOM .PART, APT files into memory.

BLOAD MEMORY.800.II
BLOAD MEMORY.2000.II
BLOAD MEMORY.A7C.II
BLOAD BOTTOM.PART.A,A$2170
BLOAD BOTTOM.PART.B,A$3E00
BLOAD APT.II

18. If you wish to have the APTs listed in issue #50 type in the following modification. If not, just skip this step.

18E8:60 0A

19. Save the game.

BSAVE BOULDER DASH II, A$7FD, L$7E03

You will need to type in the programs below and save them to disk before beginning to make a file copy of Super Boulder Dash. Also the programs with a ".I" and a ".II" at the end of their names are for Boulder Dash I and Boulder Dash II, respectively. This is because there are small differences between the two versions of the game (Boulder Dash I & II).

The file, MEMORY.A42.I, moves a copy of page 2 down from $A000-$BFFF to page 1 and page 2 and then restores the bottom part of page 1 from $9970-$9FFF.

```
0A42:00 A9 20 8D 5C 0A        $F20F
0A48:A9 40 8D 59 0A A9 A0 8D  $768B
0A50:56 0A A0 00 B9 00 00 99  $E067
0A58:00 00 99 00 00 C8 D0 F4  $D284
0A60:EE 56 0A EE 5C 0A EE 59  $82CD
0A68:0A AD 59 0A C9 60 D0 E4  $92B2
0A70:A9 99 8D 8F 0A A9 00 A2  $4EE8
0A78:70 8D 42 0A AC 42 0A B9  $D28C
0A80:2A 03 8D 92 0A B9 00 03  $371C
0A88:8D 91 0A A0 00 BD 00 00  $B079
0A90:99 00 00 E8 D0 03 EE 8F  $35F2
0A98:0A C8 C0 28 D0 EF EE 42  $E494
0AA0:0A AD 42 0A C9 2A D0 D4  $333D
0AA8:60                       $3B83
```

BSAVE MEMORY.A42.I,L$A42,L$67

The file, MEMORY.2000.I, puts the hi-res locations that MEMORY.A42.I needs at $300, moves $2100-$396F back to its original position at $8100-$996F, copies the created bottom part of page 1 to $9970-$9FFF and copies page 2 to $A000-$BFFF. It also turns on the hi-res graphics and page 1.

```
2000:A0 53 B9 2C 20 99 00 03  $E507
2008:88 10 F7 A0 00 B9 00 21  $3C12
2010:99 00 81 C8 D0 F7 EE 0F  $EBFE
2018:20 EE 12 20 AD 12 20 C9  $4567
2020:C0 D0 EA AD 57 C0 AD 52  $C8AA
2028:C0 4C 00 08 50 50 D0 D0  $F068
2030:D0 D0 D0 D0 D0 D0 50 50  $F068
2038:50 50 50 50 50 50 D0 D0  $F068
2040:D0 D0 D0 D0 D0 D0 50 50  $F068
2048:50 50 50 50 50 50 D0 D0  $F068
2050:D0 D0 D0 D0 D0 D0 39 3D  $7B37
```

```
2058:21 25 29 2D 31 35 39 3D $B3CF
2060:22 26 2A 2E 32 36 3A 3E $AB37
2068:22 26 2A 2E 32 36 3A 3E $334F
2070:23 27 2B 2F 33 37 3B 3F $C3DF
2078:23 27 2B 2F 33 37 3B 3F $334F
```

BSAVE MEMORY.2000.I, A$2000, L$80

The file, CREATE.BOTTOM.PART, takes the first byte at line 150 on hi-res page 1 and stores it at $5970. It then takes then second byte and stores it at $5971. It continues this process until it reaches the end of the line and then it goes down to the next line (line 151). It keeps doing this until it reaches the bottom of the screen. The whole reason for this is that there are no differences between page 1 and page 2 above line 150. So why not save just the part that is different?

```
0800:00 A9 59 8D 23 08 A9 00 $AD75
0808:A2 70 8D 00 08 AC 00 08 $D4AE
0810:B9 64 08 8D 20 08 B9 3A $F183
0818:08 8D 1F 08 A0 00 B9 00 $9FF1
0820:00 9D 00 00 E8 D0 03 EE $3331
0828:23 08 C8 C0 28 D0 EF EE $652C
0830:00 08 AD 00 08 C9 2A D0 $01C5
0838:D4 60 50 50 D0 D0 D0 D0 $730D
0840:D0 D0 D0 D0 50 50 50 50 $43DD
0848:50 50 50 50 D0 D0 D0 D0 $730D
0850:D0 D0 D0 D0 50 50 50 50 $43DD
0858:50 50 50 50 D0 D0 D0 D0 $730D
0860:D0 D0 D0 D0 39 3D 21 25 $F5DF
0868:29 2D 31 35 39 3D 22 26 $19A5
0870:2A 2E 32 36 3A 3E 22 26 $617D
0878:2A 2E 32 36 3A 3E 23 27 $9A24
0880:2B 2F 33 37 3B 3F 23 27 $7A44
0888:2B 2F 33 37 3B 3F       $4210
```

BSAVE CREATE.BOTTOM.PART, A$800, L$8E

The file, APT.I, clears both hi-res screens and performs the APTs.

```
09FE:A9 20                   $B2CC
0A00:8D 0E 0A A9 40 8D 11 0A $2944
0A08:A0 00 A9 00 99 00 40 99 $12D0
0A10:00 60 C8 D0 F7 EE 0E 0A $C09D
0A18:EE 11 0A AD 11 0A C9 60 $A446
0A20:D0 E8 AD 54 C0 60 AD 61 $81A4
0A28:C0 10 04 A6 26 F6 28 AD $8BF6
0A30:62 C0 10 03 4C 20 18 AD $EA55
0A38:61 C0 10 03 4C E7 16 4C $6DC4
0A40:18 18                   $F58A
```

BSAVE APT.I, A$9FE, L$44

The file, MEMORY.2000.II, puts the hi-res locations that MEMORY.A7C.II needs at $300, moves $2600-$3DFF back to its original position at $8600-$9DFF, copies the created bottom part of page 1 to $0370-$7FF and $9E00-$9FFF, and copies page 2 to $A000-$BFFF. It also turns on the hi-res graphics and page 1.

```
2000:A9 26 8D 3D 20 A9 86 8D $4E03
2008:40 20 A9 C0 8D 38 20 20 $DE57
2010:39 20 A9 21 8D 3D 20 A9 $5BE6
2018:03 8D 40 20 A9 08 8D 38 $A3E3
2020:20 20 39 20 A0 53 B9 53 $3DE1
2028:20 99 00 03 88 10 F7 AD $372B
2030:57 C0 AD 52 C0 4C 00 08 $E2B4
2038:00 A0 00 B9 00 00 99 00 $11D9
2040:00 C8 D0 F7 EE 3D 20 EE $3AD9
2048:40 20 AD 40 20 CD 38 20 $4794
2050:D0 E9 60 50 50 D0 D0 D0 $7E1A
2058:D0 D0 D0 D0 D0 50 50 50 $1EFA
2060:50 50 50 50 50 D0 D0 D0 $7E1A
2068:D0 D0 D0 D0 D0 50 50 50 $1EFA
2070:50 50 50 50 50 D0 D0 D0 $7E1A
2078:D0 D0 D0 D0 D0 39 3D 21 $2649
2080:25 29 2D 31 35 39 3D 22 $AD20
2088:26 2A 2E 32 36 3A 3E 22 $8598
2090:26 2A 2E 32 36 3A 3E 23 $2CA0
2098:27 2B 2F 33 37 3B 3F 23 $BCF0
20A0:27 2B 2F 33 37 3B 3F    $316D
```

BSAVE MEMORY.2000.II, A$2000, L$A8

The file, MEMORY.A7C.II, moves a copy of page 2 down from $A000-$BFFF to page 1 and page 2 and then restores the bottom part of page 1 from $370-$7FF and $9E00-$9FFF.

```
0A7C:00 A9 20 8D             $64A2
0A80:96 0A A9 40 8D 93 0A A9 $2A0E
0A88:A0 8D 90 0A A0 00 B9 00 $5702
0A90:00 99 00 00 99 00 00 C8 $CA8D
0A98:D0 F4 EE 90 0A EE 96 0A $41B1
0AA0:EE 93 0A AD 93 0A C9 60 $A033
0AA8:D0 E4 A9 03 8D C9 0A A9 $62AC
0AB0:00 A2 70 8D 7C 0A AC 7C $DF58
0AB8:0A B9 2A 03 8D CC 0A B9 $376C
0AC0:00 03 8D CB 0A A0 00 BD $D1C9
0AC8:00 00 99 00 00 E8 D0 0F $C8E7
0AD0:EE C9 0A AD C9 0A C9 08 $58E4
0AD8:D0 05 A9 9E 8D C9 0A C8 $E337
0AE0:C0 28 D0 E3 EE 7C 0A AD $5B0C
0AE8:7C 0A C9 2A D0 C8 60    $4163
```

BSAVE MEMORY.A7C.II, A$A7C, L$73

The file, APT.II, clears both hi-res screens and performs the APTs. For some reason the advance to next cave does not function on level 5.

```
0A38:A9 20 8D 48 0A A9 40 8D $A9EE
0A40:4B 0A A0 00 A9 00 99 00 $2E56
0A48:40 99 00 60 C8 D0 F7 EE $D080
0A50:48 0A EE 4B 0A AD 4B 0A $4118
0A58:C9 60 D0 E8 AD 54 C0 60 $838A
0A60:AD 61 C0 10 04 A6 26 F6 $7F53
0A68:28 AD 62 C0 10 03 4C DF $AF71
0A70:19 AD 61 C0 10 03 4C A6 $6DA9
0A78:18 4C D7 19             $5160
```

BSAVE APT.II, A$A38, L$4E

MEMORY.A42.I.SOURCE

```
        .OR $A42
        .TF MEMORY.A42.I
HI.RES.LOW   .EQ   $300
HI.RES.HIGH  .EQ   $32A
LINE.NUMBER
        .HS 00
MOVE.SCREEN.DOWN
        LDA #$20
        STA DESTINATION2
        LDA #$40
        STA DESTINATION
        LDA #$A0
        STA SOURCE
        LDY #$00
LOOP
        .HS B900
SOURCE
        .HS 00
        .HS 9900
DESTINATION
        .HS 00
        .HS 9900
DESTINATION2
        .HS 00
        INY
        BNE LOOP
        INC SOURCE
        INC DESTINATION2
        INC DESTINATION
        LDA DESTINATION
        CMP #$60
        BNE LOOP
CHANGE.BOTTOM.SCREEN1
        LDA #$99
        STA SOURCE.POINTER
        LDA #$00
        LDX #$70
        STA LINE.NUMBER
LOOP2
        LDY LINE.NUMBER
        LDA HI.RES.HIGH,Y
        STA SCREEN.POINTER+1
        LDA HI.RES.LOW,Y
        STA SCREEN.POINTER
        LDY #$00
LOOP3
        .HS BD00
SOURCE.POINTER
        .HS 00
        .HS 99
SCREEN.POINTER
        .HS 0000
        INX
        BNE .1
        INC SOURCE.POINTER
.1      INY
        CPY #$28
        BNE LOOP3
        INC LINE.NUMBER
        LDA LINE.NUMBER
        CMP #$2A
        BNE LOOP2
        RTS
```

MEMORY.2000.I.SOURCE

```
        .OR $2000
        .TF MEMORY.2000.I
HI.RES.POINTERS   .EQ   $300
START
        LDY #$53
.1      LDA HI.RES.STUFF,Y
        STA HI.RES.POINTERS,Y
        DEY
        BPL .1
        LDY #$00
LOOP    .HS B900
SOURCE.ADDRESS
        .HS 21
        .HS 9900
TARGET.ADDRESS
        .HS 81
        INY
        BNE LOOP
        INC SOURCE.ADDRESS
        INC TARGET.ADDRESS
        LDA TARGET.ADDRESS
        CMP #$C0
        BNE LOOP
        LDA $C057
        LDA $C052
        JMP $0800
HI.RES.STUFF
        .HS 5050
        .HS D0D0D0D0D0D0D0D0
        .HS 5050505050505050
        .HS D0D0D0D0D0D0D0D0
        .HS 5050505050505050
        .HS D0D0D0D0D0D0D0D0
        .HS 393D
        .HS 2125292D3135393D
        .HS 22262A2E32363A3E
        .HS 22262A2E32363A3E
        .HS 23272B2F33373B3F
        .HS 23272B2F33373B3F
```

CREATE.BOTTOM.PART.SOURCE

```
        .OR $800
        .TF CREATE.BOTTOM.PART
LINE.NUMBER
        .HS 00
CHANGE.BOTTOM.SCREEN1
        LDA #$59
        STA SOURCE.POINTER
        LDA #$00
        LDX #$70
        STA LINE.NUMBER
LOOP2
        LDY LINE.NUMBER
        LDA HI.RES.HIGH,Y
        STA SCREEN.POINTER+1
        LDA HI.RES.LOW,Y
        STA SCREEN.POINTER
        LDY #$00
LOOP3
        .HS B9
SCREEN.POINTER
        .HS 0000
        .HS 9D00
SOURCE.POINTER
        .HS 00
        INX
        BNE .1
        INC SOURCE.POINTER
.1      INY
        CPY #$28
        BNE LOOP3
        INC LINE.NUMBER
        LDA LINE.NUMBER
        CMP #$2A
        BNE LOOP2
        RTS
HI.RES.LOW
        .HS 5050
        .HS D0D0D0D0D0D0D0D0
        .HS 5050505050505050
        .HS D0D0D0D0D0D0D0D0
        .HS 5050505050505050
        .HS D0D0D0D0D0D0D0D0
HI.RES.HIGH
        .HS 393D
        .HS 2125292D3135393D
        .HS 22262A2E32363A3E
        .HS 22262A2E32363A3E
        .HS 23272B2F33373B3F
        .HS 23272B2F33373B3F
```

MEMORY.A7C.II.SOURCE

```
        .OR $A7C
        .TF MEMORY.A7C.II
HI.RES.LOW   .EQ   $300
HI.RES.HIGH  .EQ   $32A
LINE.NUMBER
        .HS 00
MOVE.SCREEN.DOWN
        LDA #$20
        STA DESTINATION2
        LDA #$40
        STA DESTINATION
        LDA #$A0
        STA SOURCE
        LDY #$00
LOOP
        .HS B900
SOURCE
        .HS 00
        .HS 9900
DESTINATION
        .HS 00
        .HS 9900
DESTINATION2
        .HS 00
        INY
        BNE LOOP
        INC SOURCE
        INC DESTINATION2
        INC DESTINATION
        LDA DESTINATION
        CMP #$60
        BNE LOOP
CHANGE.BOTTOM.SCREEN1
        LDA #$03
        STA SOURCE.POINTER
        LDA #$00
        LDX #$70
        STA LINE.NUMBER
LOOP2
        LDY LINE.NUMBER
        LDA HI.RES.HIGH,Y
        STA SCREEN.POINTER+1
        LDA HI.RES.LOW,Y
        STA SCREEN.POINTER
        LDY #$00
LOOP3
        .HS BD00
SOURCE.POINTER
        .HS 00
        .HS 99
SCREEN.POINTER
        .HS 0000
        INX
        BNE .1
        INC SOURCE.POINTER
        LDA SOURCE.POINTER
        CMP #$08
        BNE .1
        LDA #$9E
        STA SOURCE.POINTER
.1      INY
        CPY #$28
        BNE LOOP3
        INC LINE.NUMBER
        LDA LINE.NUMBER
        CMP #$2A
        BNE LOOP2
        RTS
```

MEMORY.2000.II.SOURCE

```
        .OR $2000
        .TF MEMORY.2000.II
HI.RES.POINTERS   .EQ   $300
START
        LDA #$26
        STA SOURCE.ADDRESS
        LDA #$86
        STA TARGET.ADDRESS
        LDA #$C0
        STA END.TARGET.ADDRESS
        JSR MEMORY.MOVER
        LDA #$21
        STA SOURCE.ADDRESS
        LDA #$03
        STA TARGET.ADDRESS
        LDA #$08
        STA END.TARGET.ADDRESS
        JSR MEMORY.MOVER
        LDY #$53
.1      LDA HI.RES.STUFF,Y
        STA HI.RES.POINTERS,Y
        DEY
        BPL .1
        LDA $C057
        LDA $C052
        JMP $0800
END.TARGET.ADDRESS
        .HS 00
MEMORY.MOVER
        LDY #$00
LOOP    .HS B900
SOURCE.ADDRESS
        .HS 00
        .HS 9900
TARGET.ADDRESS
        .HS 00
        INY
        BNE LOOP
```

```
        INC  SOURCE.ADDRESS
        INC  TARGET.ADDRESS
        LDA  TARGET.ADDRESS
        CMP  END.TARGET.ADDRESS
        BNE  LOOP
        RTS
HI.RES.STUFF
        .HS  5050
        .HS  D0D0D0D0D0D0D0D0
        .HS  5050505050505050
        .HS  D0D0D0D0D0D0D0D0
        .HS  5050505050505050
        .HS  D0D0D0D0D0D0D0D0
        .HS  393D
        .HS  2125292D3135393D
        .HS  22262A2E32363A3E
        .HS  22262A2E32363A3E
        .HS  23272B2F33373B3F
        .HS  23272B2F33373B3F
```

If anyone wishes to correspond (location doesn't matter to me even if you are in another country), my address is:

Michael A. Horton
2500 East 4th Plain #204
Vancouver, Wa 98661

If you do write please include a phone number if possible and a time to call (Example: 5 p.m. to 10 p.m. PST). Don't forget to include the area code.

Krakowicz — NY

The Basics of Kracking

Part 10

The Arcade Machine with notes on NMI and IDSI's Juggler

Softkey for...

The Arcade Machine

Broderbund

This Broderbund protection scheme is a challenge for copiers, since it uses the technique known as spiraling or quarter-tracking, as well as the standard Broderbund system of a new address marker for each track. An attempt to copy the disk with a conventional nibble copier quickly reveals that tracks 0 and $3-$11 are easily copied with an address marker of D5 AA 96, while the rest of the tracks are a mystery. Probing into the loader reveals the following information about track usage:

Track	Contents
T0/S0	Preloader —> $800-$8FF (as always)
T0/S1-5	loader —> $300-$7FF
T1-2	Hires split "Broderbund" logo and program
T12-20	Main program which loads into $800-$BFFF
T12-13.5	Four half tracks used for quarter-tracking
T3-4	#1 shape creator
T5-6	#2 path creator
T7-8	#3 game options
T9-A	#4 level options
TC-D	#5 bkgd/title creator
TE-F	#6 load/save game
T10-11	#7 create game disk (option #8 jumps to $0800 to run the game)

The approach to Kracking this type of program seems straightforward: load the program into memory, reset it, and save it out to disk as a binary file, with the appropriate memory moves. Hopefully, you'll locate the starting address and be able to run the binary file at will. If you wish to include all of the advertising for Broderbund at the beginning, this works. If you try to delete the dual banner, it crashes. The reason is that module switching is via the stack—they push the correct location onto the stack and do an RTS. So, unless you happen to know the value of the program counter (that is, exactly what the address was when you stopped), the stack pointer (S) and the processor status word (P), and restore them exactly as they were before the reset, the program probably won't run. Anyone who tried to break JUGGLER found this to be frustrating in the extreme, since sometimes the game would run all the way through the first level before crashing - the same technique was used there, but with even more protection.

There is a hard way and an easy way to do everything, and if you are completely restricted to software devices, it is still possible to break Arcade Machine. Referring to the nibble alteration techniques described in the previous episode, it is possible to locate and alter the game loader so that it halts with conditions well defined after the entire program is in memory. If it is your purpose in life to learn as much as you possibly can about disk protection schemes and the circumvention thereof (only a few really crazy people are so inclined), this is rewarding. If you are interested in preparing an unprotected version of the game with minimum advertising and minimum effort, however, there is an easier way.

Non-Maskable Interrupt

This solution is elegant, but requires a visit to that god of the underworld **Hardware**. By now everyone is (or should be) familiar with the term NMI, thanks to an oversold card which uses this technique to replay single-load games from disk. NMI stands for Non-Maskable Interrupt, one of four types of interrupt available on the 6502 (the others are reset, break, and the IRQ or interrupt request). As the name of this one implies, it is an interrupt which must be attended, regardless of whatever else the CPU had in mind to do next. This line comes directly from pin 6 of the CPU chip, is held at 5 volts (logic 1) by a 1K resistor, and run out to pin 29 of the peripheral connectors. Connecting this pin momentarily to ground (pin 26) begins a small microprogram within the 6502 which stores the program counter ('PC', two bytes) and then the processor status word ('P', one byte) on the stack, and jumps to the address stored in locations $FFFA and $FFFB in the F8 ROM.

The Stack ($100-1FF)

This business of pushing onto the stack is a little obscure, so let's spend a few moments describing the stack structure. We all know that the stack is in page one of memory ($100-$1FF), and that something called a stack pointer (S) points to an address within that range. If the following program were run, the stack would look like as shown below:

```
1000:   TSX
1001:   TXA
1002:   JSR $1010
1005:   ...
1010:   JSR $1020
1020:   JSR $1030
1030:   TSX
        BRK
```

	Stack
Final stack pointer location >	xx (any)
	22
	10
	12
	10
	04
First stack pointer location >	10

This "program" stores the first value of the stack pointer in the accumulator, JSR'S to three places, stores the final value of the stack pointer in the X-register, and then halts. (We have to neglect for the moment that Apple's monitor does some weird things to the stack after the "BRK"). If we examine the stack memory between the locations in the ACC, and X-reg, we will find the values listed above. Although we speak of the stack as a "push-down" (also "LIFO" for Last-In, First-Out) stack, what actually happens is that the value of the stack pointer is decremented, so that it points to a location one less than it was. The subroutine addresses to which the program would return (if it were given an "RTS") are stored in normal fashion of low byte, high byte, at a location one higher than the value of the stack pointer. The RTS instruction transfers these numbers into the program counter, increments the stack pointer by two, increments the low byte by one, and starts the program executing again at the location of the program counter. The stack pointer now points to (one below) the next subroutine return address, and the next "RTS" instruction encountered in the program will return to that address. Notice that the final location of the stack pointer can have anything in it, since it points to the location where the next byte will be stored, not where the last one was stored. The data pairs "22,10", "12,10", and "04, 10" correspond to the subroutine return addresses $1023, $1013, and $1005 for the program, each one being one less than the actual return point.

That digression was intended to clarify the stack structure that results from an NMI signal:

Stack pointer: (anything)
S+1 status word (P)
S+2 Program Ctr Low (PCL)
S+3 Program Ctr Hi (PCH)

This was set up to allow an external device to interrupt the Apple, and then to resume the interrupted program exactly where it was before the interrupt occurred. The instruction that makes it all happen is "RTI", which obligingly puts the processor status word back, restores the original value of the PC, and cranks up the program just as it was before the NMI line was yanked.

The practical implementation of this trick in Kracking requires a minimum of two things: an altered F8 ROM and a switch. A normal F8 ROM has $FB $03 at $FFFA-$FFFB, which means that an NMI signal will execute the instruction at $03FB. Prudent software publishers will put there either a jump to the beginning of the game or a reboot: 4C 00 C6. To get around the problem, the F8 ROM must be modified. Since most serious Krackists already have a KrakROM or Lockbuster, etc., which relocates $0-$7FF memory when reset is pressed, this is not a major problem. You should put the starting address of the memory move routine in locations $FFFA-$FFFB, and burn a new 2716 EPROM. After this PROM is installed in the F8 socket, activating the NMI line will save all of the volatile memory as well as the PC and P.

(*A word of caution* - if you don't have a solid-state switch on the NMI line, you'll store some additional garbage on the stack, but the system will still work).

Each time you use the NMI ROM, you'll have to examine the memory area where the stack is stored. Since the stack pointer is always one less than the last location stored into, you should have no trouble identifying the correct value of PC and P. After saving the game, with memory moves if required, set the stack pointer to the location of the status word-1 (use LDX #NN, TXS), and do an RTI instruction. The program will start right back up as if it had never been interrupted. Be sure that your memory relocate routine in ROM saves the value of the A, X, and Y registers, and restores the correct values before the RTI.

One final caution — some games (like JUGGLER) require that you have an unmodified ROM in the F8 socket - this requires a little more assistance from the god of hardware, and will be dealt with in a future episode describing other applications of the NMI technique.

Returning to the A.M. Krack, you now can boot the disk and get to the main menu. Do the NMI trick by closing a switch wired between pins 29 and 26 of any peripheral card, and move the excess memory to $2000-$3FFF (the Norwegian nurds were nice enough to leave us Hi-Res page one open — tak!), Including $0-$8FF and $B600-$BFFF. Add the appropriate memory move routines as well as the register restore, stack pointer adjust, and RTI, then boot a slave disk and BSAVE the memory from $900-$9600.

Copy tracks $3-$11 from the original A.M. with your favorite copier, and tell the VTOC that those tracks are occupied. Save the file onto any tracks above 11, and, using the boot modifier described in the KKK III on WAY OUT, load in the main program as part of the boot. You should now be off and running with your own freshly broken copy of Arcade Machine.

It's not really as hard as it sounds, and if you really like to program your own left-right shoot-em-ups without learning to program, the result is worth the effort.

The Guardian — FL

Run MECC On Hard Disk

Softkey for...

MECC 3.5" ProDOS disks

MECC

Softkey for...

Backyard Birds
Chemistry: Balancing Equations
Cleanwater Detectives
Chemistry: The Periodic Table
Communikeys
Conquering Decimals (+,-)
Conquering Decimals (X,/)
Conquering Fractions (+,-)
Conquering Fractions (X,/)
Conquering Math Worksheet Generator
Conquering Percents
Conquering Ratios & Proportions
Coordinate Math
Decimal Concepts
Equation Math
Estimation Quicksolve I
Estimation Quicksolve II
Estimation Strategies
Exploring Gas Laws
Five-Star-Forecast
Fossil Hunter

Hail and well met, Eamon adventurer.

COMPUTIST has news of great import for loyal supporters of Eamon and members in good standing at the Main Hall. There is a newsletter for Eamon Adventure Buffs.

Yes! I want to support Eamon adventure and encourage new adventures to be written. Sign me up for a one year subscrition.

☐ US & Canada: $7.00 ☐ Foreign: $12.00 (U.S. funds)

I also want to order some back issues at $1.75 each for 1-5 issues and $1.25 each for 6 or more issues.

☐ Jun'88 ☐ Sep'88 ☐ Dec'88 ☐ Mar'89 ☐ Jun'89 ☐ Sep'89
☐ Dec'89 ☐ Mar'90 ☐ Jun'90 ☐ Sep'90 ☐ Dec'90 ☐ Mar'91
☐ Jun'91 ☐ Sep'91 ☐ Dec'91

Total enclosed _____

Name _____

Address _____

_____

City _____ State _____ Zip _____

"The Eamon Adventurer's Guild"

Tom Zuchowski is the editor and driving force behind it. The newsletter is printed on 8.5" x 11" bond, double sided with dot matrix type. It is non-profit (it's his hobby) and Tom's avowed purpose is to maintain a single point collection and clearing house for Eamon Adventures. He also intends to fix all known bugs in these adventures.

This is a grand concept worthy of support. We highly recommend that you subscribe. This newsletter will help you keep in touch with other Eamon adventurers. But more important, it will improve the state of Eamon adventures and encourage new adventures to be written. Tom has accomplished a great deal along these lines already, not only fixing bugs but also improving the Eamon Main program and authoring the version 7.0 Dungeon Designer Disk.

The "Guild" is printed quarterly. A 1 year subscription is:

US & Canada: $7.00 Foreign: $12.00 (U.S. funds)

The Eamon Adventurer's Guild
7625 Hawkhaven Dr.
Clemmons, NC 27012
(919) 766-7490

Eamon Adventure for only $1 (or less) each

☐ 1 Main Hall & Beginners Cave
☐ 2 The Lair of the Minotaur
☐ 3 The Cave of the Mind
☐ 4 The Zyphur Riverventure
☐ 5 Castle of Doom
☐ 6 The Death Star
☐ 7 The Devil's Tomb
☐ 8 The Abductor's Quarters
☐ 9 Assault on the Clonemaster
☐ 10 The Magic Kingdom
☐ 11 The Tomb of Molinar
☐ 12 The Quest for Trezore
☐ 13 Caves of Treasure Island
☐ 14 Furioso
☐ 15 Heroes Castle
☐ 16 The Caves of Mondamen
☐ 17 Merlin's Castle
☐ 18 Hogarth Castle
☐ 19 Death Trap
☐ 20 The Black Death
☐ 21 The Quest for Marron
☐ 22 The Senator's Chambers
☐ 23 The Temple of Ngurct
☐ 24 Black Mountain
☐ 25 Nuclear Nightmare
☐ 26 Assault on the Mole Man
☐ 27 Revenge of the Mole Man
☐ 28 The Tower of London
☐ 29 The Lost Island of Apple
☐ 30 The Underground City
☐ 31 The Gauntlet
☐ 32 House of Ill Repute
☐ 33 The Orb of Polaris
☐ 34 Death's Gateway
☐ 35 The Lair of Mutants
☐ 36 The Citadel of Blood
☐ 37 Quest for the Holy Grail
☐ 38 City in the Clouds
☐ 39 Museum of Unnatural History
☐ 40 Daemon's Playground
☐ 41 Caverns of Lanst
☐ 42 Alternate Beginners Cave
☐ 43 Priests of Xim!
☐ 44 Escape from the Orc Lair
☐ 45 SwordQuest
☐ 46 Lifequest
☐ 47 FutureQuest
☐ 48 Picnic in Paradise
☐ 49 The Castle Kophinos
☐ 50 Behind the Sealed Door
☐ 51 The Caves of Silmon Bluff
☐ 52 The Devil's Dungeon
☐ 53 Feast of Carnak
☐ 54 Crystal Mountain
☐ 55 The Master's Dungeon
☐ 56 The Lost Adventure
☐ 57 The Manxome Foe
☐ 58 The Land of Death
☐ 59 Jungles of Vietnam
☐ 60 The Sewers of Chicago
☐ 61 The Harpy Cloud
☐ 62 The Caverns of Doom
☐ 63 Valkenburg Castle
☐ 64 Modern Problems
☐ 65 The School of Death
☐ 66 Dungeons of Xenon
☐ 67 Chaosium Caves
☐ 68 The Smith's Stronghold
☐ 69 The Black Castle of NaGog
☐ 70 The Tomb of Y'Golonac
☐ 71 Operation Cleanup
☐ 72 House on Ramon Ridge
☐ 73 The Deep Canyon
☐ 74 DharmaQuest
☐ 75 Temple of the Guild
☐ 76 The Search for Yourself
☐ 77 Temple of the Trolls
☐ 78 The Prince's Tavern
☐ 79 The Castle of Count Fuey
☐ 80 The Search for the Key
☐ 81 The Rescue Mission
☐ 82 Escape from Mansi Island
☐ 83 The Twin Castles
☐ 84 Castle of Riveneta
☐ 85 The Time Portal
☐ 86 Castle Mantru
☐ 87 Caves of Hollow Mountain
☐ 88 The Shopping Mall
☐ 89 Super Fortress of Lin Wang
☐ 90 The Doomsday Clock
☐ 91 FutureQuest II
☐ 92 The Fugitive
☐ 93 Flying Circus
☐ 94 Blood Feud
☐ 95 The Maze of Quasequeton
☐ 96 The Chamber of the Dragons
☐ 97 The House of Secrets
☐ 98 Slave Pits of Kzorland
☐ 99 In the Clutches of Torrik
☐ 100 Sorceror's Spire
☐ 101 Ground Zero
☐ 102 The Eamon Railroad
☐ 103 Top Secret
☐ 104 The Lost World
☐ 105 The Strange Resort
☐ 106 Camp Eamon
☐ 107 The Last Dragon
☐ 108 The Mines of Moria
☐ 109 The Forest of Fear
☐ 110 Fire Island
☐ 111 A Vacation in Europe
☐ 112 Hills of History
☐ 113 The Life-Orb of Mevtrelek
☐ 114 Thror's Ring
☐ 115 The Ring of Doom
☐ 116 The Iron Prison
☐ 117 Dungeon of Doom (40 col)
☐ 117 Dungeon of Doom (80 col)
☐ 118 Pittfall
☐ 119A Grunewalde
☐ 119B Grunewalde
☐ 120 Orb of My Life
☐ 121 Wrenhold's Secret Vigil
☐ 122 The Valley of Death
☐ 123 Wizard of the Spheres
☐ 124 Assault on Dolni Keep
☐ 125 The Mattimoe Palace
☐ 126 The Pyramid of Anharos
☐ 127 The Hunt for the Ring
☐ 128 Quest of Erebor
☐ 129A Return to Moria
☐ 129B Return to Moria
☐ 130 Haradwaith
☐ 131 Nucleus of the Ruby
☐ 132 Rhadshur Warrior
☐ 133 The Final Frontier
☐ 134 Pyramid of the Ancients
☐ 135 The Tomb of Evron
☐ 136 The Mountain Fortress
☐ 137 The Ruins of Ivory Castle
☐ 138 Starfire
☐ 139 Peg's Place
☐ 140 Beginner's Forest
☐ 141 The Infested Fortress
☐ 142 The Beermeister's Brewery
☐ 143 The Alternate Zone
☐ 144 Gartin Manor
☐ 145A Buccaneer!
☐ 145B Buccaneer!
☐ 146 The House of Horrors
☐ 147A The Dark Brotherhood
☐ 147B The Dark Brotherhood
☐ 148 Journey to Jotunheim
☐ 149A Elemental Apocalypse
☐ 149B Elemental Apocalypse
☐ 149C Elemental Apocalypse
☐ 149D Elemental Apocalypse
☐ 150 Walled City of Darkness
☐ 151 EamonS.A.R.-1(Deneb Raid)
☐ 152 The Computer Club of Fear
☐ 153 Lost!
☐ 154 A Trip to Fort Scott
☐ 155 Tomb of the Vampire
☐ 156 The Lake
☐ 157 Pathetic Hideout of Mr R.
☐ 158 The Lair of Mr Ed
☐ 159 The Bridge of Catzad-Dum
☐ 160 Monty Python & Holy Grail
☐ 161A Operation Endgame
☐ 161B Operation Endgame
☐ 161C Operation Endgame
☐ 162 Eamon 7.0 Demo Adventure
☐ 163 The Sands of Mars
☐ 164 A Real Cliffhanger
☐ 165A Animal Farm
☐ 165B Animal Farm
☐ 166A Storm Breaker
☐ 166B Storm Breaker
☐ 166C Storm Breaker
☐ 167 Expedition to the Darkwoods
☐ 168 The High School of Horrors
☐ 169 The Black Phoenix
☐ 170 Ragnarok Revisited
☐ 171 The Pyramid of Cheops
☐ 172 The Mountain of the Master
☐ 173 The House that Jack Built
☐ 174 Escape from Granite Hall
☐ 175 Anatomy of the Body
☐ 176 Dirtie Trix's Mad Maze
☐ 177 Shippe of Fooles
☐ 178 The Alien Intruder
☐ 179 The Wizard's Tower
☐ 180 Gamma 1
☐ 181 The Eamon Sewer System
☐ 182 Farmer Brown's Woods
☐ 183 The Boy and the Bard
☐ 184 Quest for Orion
☐ 185 The Body Revisited
☐ 186 Beginners Cave II
☐ 187 Batman!
☐ 188 Encounter: The Bookworm
☐ 189 The Ruins of Belfast
☐ 190 Shift Change at Grimmwax
☐ 191 Enhanced Beginners's Cave
☐ 192 Mean Streets
☐ 193 The Creature of Rhyl
☐ 194 Attack of the Kretons
☐ 195 The Training Grounds
☐ 196 The House of Horrors
☐ 197 Star Wars - Tempest One
☐ 198 Revenge of the Bookworm
☐ 199 Quest of the Crystal Wand
☐ 200 The Lost Isle
☐ 201 The Caverns of Vanavara
☐ 202 The Plain of Srevi
☐ 203 Lotto's Masterpiece
☐ 204A Sanctuary
☐ 204B Sanctuary
☐ 205 Utterly Outrageous
☐ 206 Curse of the Hellsblade
☐ 207 Eamon Renegade Club
☐ 208 Assualt on Helstar
☐ 209 Apocalypse 2021
☐ 210 Return of Ngurct
☐ 211 Lair of the Marauders
☐ 212 Haunted Keep
☐ 213 Demongate

☐ Dungeon Designer Diskette v7.0
☐ Multi-Disk Supplement (DDD7.0)
☐ Eamon Utilities Diskette
☐ Graphics Main Hall

☐ Send me the Complete set of Eamon for: $125.00

Total number of Adventure disks _____ x $1 each = _____

Add only if total # of disks ordered is less than 10: $4.00

Washington State residents only add 7.8% sales tax. _____

Name _____

Address _____

_____

City _____ State _____ Zip _____

Country _____ Phone _____

Visa
MC _____ - _____ - _____ - _____ Exp _____

Signature _____

COMPUTIST, 33821 Orville Rd. E, Eatonville WA 98328-9590

Adventure Gaming doesn't have to cost a lot. The Eamon Adventure Gaming system was created by Donald Brown and placed into the public domain. Since then it has been updated and improved by game players all over the world. Take a look at what $1 will buy. (Get free games too.)

Note: Some Adventures are multi-part and take more than one disk. Be sure you have selected all of the disks.

The Eamon Master disk (#1) is required to play most adventures.

Free Adventures

Use the total number of adventures ordered to determine how many free adventures you get.

Be sure and check the boxes of your free disks that you want but **do not** include free disks when figuring total number of disks ordered.

# of disks at $1	# of Free disks
1-9	0
10-19	2
20-29	5
30-39	9
40-49	14
50-59	20
60-69	27
70-79	35
80-89	44
90-99	54
100-109	65
110-119	77
120-129	90
130-139	104

Complete set of Eamon

All 232 disks (includes all adventures plus designer and utility disks.)**$125**

Use your VISA/MC (206) 832-3055

COMPUTIST
33821 Orville Rd. E
Eatonville WA 98328-9590

Fraction Concepts, Inc.
Fraction Practice Unlimited
Grammar Gazzette
Grammar Toy Shop
Instant Survey
Instant Survey Sampler
Invisible Bugs
LittleTown Zoo
The Living Cell
Lunar Greenhouse
Measureworks
MECC Outliner
Mystery Matter
Mystery Objects
Paper Plane Pilot
Patterns
Picture Chompers
Probability Lab
Problem-Solving With Nim
Space Station Freedom
Spellevator
Spelling Puzzles and Tests
Spelling Series ToolKit
Sun & Seasons
Teaching Scientific Inquiry
Time Navigator
Time Navigator Leaps Back
Time Navigator Around The World
To Preserve, Protect & Defend
Weeds To Trees
Wood Car Rally
Wooly Bounce

MECC

Teachers, how would you like to use half of the MECC library (54 titles) on your hard disk without having to use their crossloader? It is possible with the help of B. Brett "Computist #53", M.E.C.C. software (1987) and Momma "Computist #77", M.E.C.C. 3.5" Disks (1990).

I was using these softkeys to make backup copies of my 5.25" ProDOS disks and to transfer them to 3.5" disks. After I finished I wondered if they would work on my hard disk. No! The softkey in issue #77 would make bootable copies that worked fine on 3.5" disks but they would give error messages when launched from my hard disk. That is all except for To Preserve, Protect and Defend.

I began to search for the difference in the disks. I had used issue #77 instead of issue #53 to softkey To Preserve, Protect and Defend. I remembered that Preserve, Protect and Defend had the same pattern as the disks in issue #53 but they were three bytes off. So instead of changing the 38 to 18 I NOPed out all of the bytes starting with 90 03 4C XX XX and the disk booted and worked. I did not think anything else about it until I started trying to launch the programs from hard disk.

I then NOPed out the same pattern on the other MECC disks as I had done with To Preserve, Protect and Defend and they worked perfectly. As a matter of fact I no longer had to search for the pattern of:

```
20 00 BF
80
11 22
B0 98
18
60
2C E5 21
30 91
38
60
```

This turns out not to be the critical pattern as described in issue #77. Instead the critical pattern appears to be:

```
20 XX XX
90 03 4C XX XX 60
```

Note: The 90 03 4C pattern appears several times on the disk. The only time it is important is when it is preceded by 20 XX XX.

The 20 XX XX seems to set up a JSR for the protection scheme and the 90 03 4C XX XX 60 carries it out. By changing the 90 03 4C XX XX to a CLC and EA's the protection scheme is skipped. The XX's are different on some of the disks but the pattern remains the same.

Change the 90 03 4C XX XX to 18 EA EA EA EA. Now you can run the disks from your hard disks with ease and no more damaged disks from students. Again, change from 90 03 4C XX XX 60 to 18 EA EA EA EA 60.

Here are the patterns and locations for some of the disks:

Title	Block	Bytes
Estimation Quicksolve I	002B	20 29 21 90 03 4C XX XX 60
Invisible Bugs	0032	20 29 21 90 03 4C XX XX 60
Lunar Greenhouse	002C	20 02 21 90 03 4C XX XX 60
Measureworks	0058	20 2A 21 90 03 4C XX XX 60
The Living Cell	000C	20 XX XX 90 03 4C 02 82 60
Grammar Gazette	002C	20 XX XX 90 03 4C 77 91 60
Estimation Strategies	002C	20 XX XX 90 03 4C E1 88 60
Little Town Zoo	002C	20 XX XX 90 03 4C 71 8A 60
Paper Plane Pilot	002C	20 XX XX 90 03 4C D2 86 60
MECC Outline	0039	20 XX XX 90 03 4C 72 0E 60
Estimation Quicksolve II	002C	20 29 21 90 03 4C XX XX 60
Exploring Gas Laws	0011	20 03 21 90 03 4C XX XX 60
BackYard Birds	xxxx	20 F1 0F 90 03 4C XX XX 60
Spelling Press	0093	20 ED 08 90 03 4C XX XX 60
Weeds To Trees	002C	20 4E A9 90 03 4C XX XX 60
Time Navigator Leaps Back	002C	20 29 21 90 03 4C XX XX 60
Time Navigator	000C	20 4E 21 90 03 4C XX XX 60
Spelling Puzzles & Tests	005A	20 ED 08 90 03 4C XX XX 60
Spellevator	0033	20 29 21 90 03 4C XX XX 60
Probability Lab	002C	20 29 21 90 03 4C XX XX 60
Picture Chompers	00A3	20 02 21 90 03 4C XX XX 60
Patterns	0012	20 EF 20 90 03 4C XX XX 60
Moneyworks	0049	20 EE 20 90 03 4C XX XX 60
Instant Survey	002C	20 F0 40 90 03 4C XX XX 60
Grammar Toy Shop	002C	20 29 21 90 03 4C XX XX 60
Five Star Forecast	000C	20 29 21 90 03 4C XX XX 60
Fossil Hunter	00DA	20 29 21 90 03 4C XX XX 60
Wooly Bounce	002C	20 02 21 90 03 4C A2 91 60
Time Navigator Around The World		20 29 21 90 03 4C 4B 75 60
Problem Solving With Nim		20 29 21 90 03 4C 17 80 60
Space Station Freedom		20 42 18 90 03 4C A1 87 60
Cleanwater Detectives		20 02 21 90 03 4C AE 89 60
Teacher Option Organizer		20 XX XX 90 03 4C 85 19 60

This method will work for any 3.5" ProDOS disk from MECC. Strangely, it will not work on the 5.25" ProDOS disks. Use the method in issue #77 or #53 for 5.25 disks.

You can copy a 5.25" MECC ProDOS disk by using COPYA.

POKE 47397,24
POKE 47398,96
RUN COPYA

Then copy the files over to a ProDOS formatted 3.5" disk and make the above sector edits and it work work on the 3.5" disk. It will not work if you copy the files back to the 5.25" disk.

It will not work with many of the 3.5 disks that have recently been transferred over from 5.25" Dos disks. MECC seems to have used something similar to Unidos on these disks which means even if they are softkeyed they will not work on a hardisk under ProDOS.

I can not stress how important it is to give as much information as possible when submitting an article. None of the above softkeys would be possible if it were not for the information from the other two articles. Please explain as much as possible and include as much sector information as possible when sending an article in.

Partial Bitkey for...

MECC COPY SYSTEM /LABEL UTILITY

MECC

I recently was able to obtain a copy of the 1991-1992 MECC Copy System/ Label Utility 3.5 format. I was unable to copy it using disk copy. I used the manual bit copy on Copy II Plus 9.0 and got no errors. The copy would partially boot and give a boot error message. I decided to examine the disk for the bytes 90 03 4C and found the following pattern:

```
20 48 3E    JSR 3E48
18          CLC
60          RTS
C9 80       CMP #80
90 03       BCC 11A4 {+03}
4C 43 22    JMP 2243
C9          1B  #1B
D0 02       BNE 11AA {+02}
38          SEC
60          RTS
```

They looked suspicious. I wondered what would happen if I changed the 38 to 18 or CLC. I tried it and the disk booted and worked perfectly.

The disk still has a Nibble count routine that prevents it from being copied normally but a manual bit copy and two sector edits makes a working copy. If someone else can help remove the Nibble count routine, we will have a softkeyed MECC Copy System.

To make a backup copy of the 3.5" MECC 1991-1992 COPY SYSTEM/ LABEL UTILITY follow these steps:

Step-by-step

1. Make a copy using a good manual bit copy system that will ignore errors.
2. Sector Edit:

Blk	Byte	From	To
00F2	09F	D0 02 38 60	DO 02 18 60
016A	09F	D0 02 38 60	D0 02 18 60

Softkey for...

Miner's Cave

MECC

This title (listed in the MECC catalog as not networkable) fits the pattern in issue #77. Search block C for 60 2C 78 11 30 91 38 60 and change the 38 to 18. This will make the disk boot and work from harddisk but the quit routine causes your computer to restart.

Step-by-step

1. Use COPYA to make a copy of the disk.

POKE 47397,24
POKE47398,96
RUN COPYA

2. Make the edit as above.

Michael S. Pollock — CA

Ⓠ I purchased a used IIe system with an early Duodisk drive (serial #676-102). This drive has trashed the 0 sector of disks under certain conditions. The fix is to cut off two capacitors, I am told. The question is which two? It also seems to be unable to read disks after repeated disk access. It would seem to be a thermal problem but where?

Ⓠ I came across an Apple ROM Card without instructions. Am I correct in assuming it was used to switch between integer and Apple BASIC?

Eric W Taylor — CA

Softkey for...

Dungeon Master's Assistant vol2

SSI

Requirements:
Apple IIgs
1 Blank Disk
Any Fast Disk Copier
Any Sector/Block Editor

This article will show how to softkey the D&D-Master Assistant vol 2 that is on the Apple Most Wanted list. I have had this softkey done for quite some time but did not have the energy to write in. Now that the editor put out a call for articles I decided once again to contribute to the only Apple magazine I subscribe to.

In issue #79 I read the softkey for this same program by Terry Waskowich. Although his softkey is quite close to what I did and both softkeys should work perfectly well, I went about my softkey in a much different way than he did. This article will show that there are many ways to complete a common task. Terry's approach was one of top down while mine is one of bottom up. I think the way I softkeyed this disk is general enough that it can be applied in many situations.

Dungeon Master's Assistant is indeed word protected. I hate word protection because I am lazy and do not want to be constantly searching for the manual to look up a word or page number. This type of protection is especially annoying after you are familiar with the program and no longer need the manual by your side when you run the program. This softkey removes all traces of the word protection from the disk and allows for easy archive copies to be made. For those of you who do not want to know what I did go ahead and jump straight to the cookbook instructions.

I started by making a copy of the original disk to perform the softkey on.

This can easily be accomplished by using any fast disk copier. I used Copy Two Plus for this step. Because the disk copies with no errors this indicates no strange RWTS is being used and that the format of the data headers/trailers and sector headers/trailers must be normal.

Next I booted my IIgs to the Finder and launched BASIC. At the BASIC prompt I entered the monitor by typing CALL -151. Once in the monitor I activated the Visit Monitor CDA by typing *#. I pressed ctrl-C to get back to BASIC and typed BYE to get back to the Finder. Next I entered the control panel and changed my boot drive to slot six. Finally I shutdown the Finder and then booted the copy of the program.

Once the program had loaded to the point where it was prompting for the password I entered the CDA menu (Open Apple-Control-Escape) and selected the Visit Monitor CDA. At this point I cleared memory (800:0 N 801<800 .BFFFM). I re-entered the CDA menu by pressing control-Y and selected quit to exit back to the running program. The computer crashed into the monitor at *AE9E. This implies that the input routine for the word protection is located somewhere around address $AE9E. I then rebooted the program and when it was again prompting for a password I entered the monitor via the Visit Monitor CDA and examined the memory around where it had crashed (*AE90L). The following is the code I saw around this location.

AE90L

```
l=ml=x l=LCbank(0/1)
00/AE90:E4 0A     CPX 0A
00/AE92:0A        ASL
00/AE93:05 E4     ORA E4
00/AE95:85 E4     STA E4
00/AE97:60        RTS End of routine
00/AE98:2C 10 C0  BIT  C010   Clear keypress
                              strobe
00/AE9B:EE 36 03  INC  0336   LSB seed for
                              random numbers?
00/AE9E:D0 03     BNE AEA3 {+03} Take branch if
                              LSB has not
                              overflowed
00/AEA0:EE 37 03  INC  0337   MSB seed for
                              random numbers?
00/AEA3:AD 00 C0  LDA  C000   Check for key
                              press
00/AEA6:10 F3     BPL  AE9B {-0D} No press yet,
                              loop
00/AEA8:C9 E0     CMP #E0     Is it the ' key?
00/AEAA:90 06     BCC AEB2 {+06} Yes, exit
                              routine
00/AEAC:C9 FF     CMP #FF     Is it the delete
                              key?
00/AEAE:F0 02     BEQ AEB2 {+02} Yes, exit
                              routine
00/AEB0:29 DF     AND #DF     Set sixth bit to
                              zero
00/AEB2:60        RTS         End of routine
00/AEB3:A9 3F     LDA #3F
00/AEB5:2C A9 FF  BIT  FFA9
00/AEB8:8D 34 03  STA  0334
```

By starting at the crash location of $AE9E and searching backwards through the above listing I determined that location $AE97 was the end of the previous routine (that is where the first RTS (60) is). The beginning of the routine were the crash occurred was AE98. Since this routine ended with an RTS I decided to check for all calls to this routine using a JSR. The code for a JSR AE98 is 20 98 AE. I used the built in GS monitor search routine to search the first bank of memory for this string (*\20 98 AE\<0.BFFFP). This search found the string in several locations.

```
\20 98 AE\<0.BFFFP
00/0201:
00/A913:
00/AAE3:
00/ABE7:
00/AD24:
```

(*Note:* The code at $A913 is located on disk at block $14, byte $13 (track $2, sector $1, byte $13). The code at $AAE3 is located on disk at block $14, byte $1E3 (Track $2, sector $3, byte $E3). The code at $ABE7 is located on disk at block $15, byte $E7 (track $2, sector $5, byte $E7). The code at $AD24 is located on disk at block $16, byte $24 (track $2, sector $9, byte $24). This info was found by searching the disk for $20 98 AE.)

The location $201 occurrence is where the string was stored when I input it. This left four possible locations where the routine that the crash occurred in could have been called from. I rebooted the computer and before I got to the word protection code I entered the monitor through the CDA Visit Monitor. I replaced the 20 98 AE at location $A913 with a 4C 69 FF (jump to monitor). I returned to the program and allowed it to continue. I got to the word protection input without entering the monitor so the call at location $A913 is not involved in the protection. I rebooted again and replaced the 20 98 AE at location $AAE3 with a 4C 69 FF. When I allowed the program to continue from this point I ended up in the monitor before I had to input the word protection. This implies that the call located at $AAE3 has something to do with the protection routine. I examined the code around $AAE3 and this is what I saw.

AAE0LL

```
l=ml=xl=LCbank(0/1)
00/AAE0:CE 32 03  DEC 0332
00/AAE3:20 98 AE  JSR  AE98   Get a key
00/AAE6:AE 9F AF  LDX  AF9F   Load current
                              password offset
00/AAE9:C9 9B     CMP #9B     Is it an escape
                              key?
00/AAEB:F0 45     BEQ AB32 {+45} If so take
                              branch
00/AAED:C9 8D     CMP #8D     Is it a carriage
                              return?
00/AAEF:F0 32     BEQ AB23 {+32} If so take
                              branch
00/AAFI:C9 88     CMP #88     Is it a backspace
                              key?
00/AAF3:D0 14     BNE AB09 {+14} If not take
                              branch
00/AAF5:8ATXA                 Transfer
                              password offset
00/AAF6:F0 EBBEQ AAE3 {-15}   If offset is zero
                              branch to get key
00/AAF8:20 B6 AE  JSR  AEB6
00/AAFB:20 3F AB  JSR  AB3F
00/AAFE:20 3C AB  JSR  AB3C
00/AB0I:20 B3 AE  JSR  AEB3
00/AB04:CE 9F AF  DEC  AF9F   Decrement
                              password offset
00/AB07:10 D2     BPL  AADB {-2E}
00/AB09:EC 9E AF  CPX  AF9E
00/AB0C:B0 D5     BCS  AAE3 {-2B}
00/AB0E:C9 A0     CMP #A0     Is it a space key?
00/AB10:90 DI     BCC AAE3 {-2F} If less than
                              space key branch
                              to get a key
00/AB12:C9 A2     CMP #A2     Is key a "
00/AB14:F0 CD     BEQ AAE3 {-33} If so branch to
                              get a key
00/ABI6:C9 FF     CMP #FF     Is key the delete
                              key?
00/AB18:F0 C9     BEQ AAE3 {-37} If so branch to
                              get a key
00/ABIA:9D 00 A0  STA A000,X  Store key at
                              current password
                              offset
00/ABID:20 90 A9  JSR  A990   Print keystroke?
00/AB20:E8        INX         Increment password offset
00/AB21:10 B5     BPL  AAD8 {-4B} Branch for
                              next keystroke
00/AB23:A9 00     LDA #00     End of password
                              designator
00/AB25:9D 00 A0  STA A000,X  Store at end of
                              password
00/AB28:20 B6 AE  JSR  AEB6
00/AB2B:A9 A0     LDA #A0     Load space
                              character
00/AB2D:20 90 A9  JSR  A990   Print keystroke?
00/AB30:18        CLC         Clear the carry bit
00/AB31:60        RTS         End of get string
                              routine
00/AB32:A9 00     LDA #00
00/AB34:9D 00 A0  STA A000,X
00/AB37:20 B6 AE  JSR  AEB6
00/AB3A:38        SEC
```

The important part of the above code is that if a carriage return is entered the code jumps to $AB23 and then continues on its merry way. Following the code from $AB23 shows that $AB31 is the end of the get string routine. I booted the disk and used the Visit Monitor CDA to place a 00 (break instruction) at $AB31. Allowing the code to continue caused a crash into the monitor after the word protection word had been entered and the carriage return had been pressed. This confirmed my guess that $AB31 was the end of the get string routine. Now I had to find out where the get string routine was being called from.

Once again I booted the disk and entered the monitor. I replaced the 60 A9 00 9D starting at location $AB31 with the code 68 AA 68 00 (PLA TAX PLA BRK). This code pulls the low byte of the return address off of the stack and places it into the X register. Then the high byte of the return address is pulled off of the stack into the A register. Finally a break instruction is executed. This segment of code will tell a programmer what address the code intends to return to.

I allowed the program to continue running by returning from the Visit Monitor CDA and exiting the CDA menu. After the word protection word had been entered and the carriage return had been entered the code crashed into the monitor as expected. The A register contained a 19 and the X register contained a D7. Adding one to the low byte of the address (the X register) tells the programmer the exact address of the next instruction to be fetched. Adding one shows that the next instruction will be executed at location $19D8. Knowing that a JSR instruction is three bytes long implies the call to the get string routine originated from address $19D5. I listed the code around $19D5 and this is what I saw.

***19D1LL**

```
l=ml=xl=LCbank(0/1)
00/19DI:A0 0D     LDY #0D
00/19D3:A9 0F     LDA #0F
00/19D5:20 15 A8  JSR  A815   Call get string
                              routine
00/19D8:A2 00     LDX #00     Offset into real
                              password
00/19DA:8E 67 18  STX  1867   Reset password
                              number
00/19DD:A0 00     LDY #00     Offset into
                              entered password
00/19DF:BD 3C IA  LDA  IA3C,X Read character of
                              real password
00/19E2:10 09     BPL  19ED {+09} Branch if last
                              character of real
                              password
00/19E4:D9 00 A0  CMP A000,Y  Compare to
                              entered password
00/19E7:D0 10     BNE 19F9 {+I0} Branch if
                              characters are not
                              same
00/19E9:E8        INX         Increment offset
                              into real password
00/19EA:C8        INY         Increment offset
                              into entered
                              password
00/19EB:D0 F2     BNE 19DF {-0E} Branch to
                              compare next
                              character
00/19ED:09 80     ORA #80     Set high bit of real
                              password
00/19EF:D9 00 A0  CMP A000,Y  Compare to
                              entered password
00/19F2:D0 0D     BNE IA0I {+0D} Branch if
                              characters do not
                              match
00/19F4:B9 01 A0  LDA  A00I,Y Load next
                              character of
                              entered password
00/19F7:F0 13     BEQ IA0C {+13} Branch out if
                              end of password
                              designator
00/19F9:BD 3C IA  LDA  IA3C,X Read character
                              from real
                              password
00/19FC:10 03     BPL  IA0I {+03} Branch if end
                              of real password
l=ml=x l=LCbank(0/1)
00/19FE:E8        INX         Increment offset
                              into real password
00/19FF:D0 F8     BNE 19F9 {-08} Branch to get
                              next password
                              character
00/IA0I:E8        INX         Increment offset
                              into real password
00/IA02:BD 3C IA  LDA  IA3C,X Load character
                              from real
                              password
00/IA05:F0 17     BEQ IAIE {+17} Branch if end
                              of password table
00/IA07:EE 67 18  INC  1867   Increment
                              password number
00/IA0A:D0 DI     BNE 19DD {-2F} Branch to
                              check next
                              password
00/IA0C:A2 FF     LDX #FF     Load password
                              number asked for
00/IA0E:EC 67 18  CPX  1867   Compare to
                              password number
                              entered
00/IAII:D0 0B     BNE IAIE {+0B} Branch if not
                              correct password
00/IA13:A9 00     LDA #00     Password is OK
00/IA15:8D 9A A8  STA  A89A   Store needed
                              value into
                              memory
00/IA18:A9 10     LDA #10     Password is OK
00/IAIA:8D 9B A8  STA  A89B   Store needed
                              value into
                              memory
00/IAID:60        RTS         End of password
                              check routine
00/IAIE:20 00 A8  JSR  A800
00/IA21:A9 09     LDA #09
00/IA23:8D 33 03  STA  0333
00/IA26:20 6C A8  JSR  A86C
00/IA29:D7 D2     CMP [D2],Y
```

The password table in memory is stored such that a character without its high bit set indicates the end of a password. The password number the code wants the user to enter is stored at location $1A0D. Location $1867 holds the password number entered by the user. Going through the above code shows that $1A1D is the end of the word protection check routine. The important thing to notice about the above code is

that if the correct word is entered values are stored at locations $A89A and $A89B. If these values are not inserted into the code bad things will happen when it is run.

The question still remained as to where the password check routine was called from. Using the same technique I explained earlier I rebooted and inserted a 68 AA 68 00 at location $1A1D. This replaced the code 60 20 00 A8. I allowed the code to continue running and typed in the password when I was prompted for it. After entering the carriage return the code crashed as expected and the A register contained a 18 and the X register contained a 53. Adding one implies the next instruction will be executed at location $1854. This implies the call to the word protection check routine occurred at location $1851. Listing the code (*184EL) around this location revealed the following.

***184EL**

```
l=ml=xl=LCbank(0/1)
```

00/184E:20 78 18	JSR 1878		Puts up the text credit page
00/1851:20 74 l9	JSR 1974		Runs the word protection check
00/1854:AD 0D 1A	LDA 1A0D		Load password number required
00/1857:CD 67 18	CMP 1867		Compare to password number entered
00/185A:D0 08	BNE 1864	{+08}	Branch if password number not correct
00/185C:A9 C4	LDA #C4		Password correct
00/185E:85 50	STA 50		Store vital code into memory
00/1860:A9 D7	LDA #D7		Password correct
00/1862:85 51	STA 5l		Store vital code into memory
00/1864:4C 99 A8	JMP A899		Jump to continue program
00/1867:00 11	BRK 11		
00/1869:11 09	ORA (09),Y		
00/186B:0A	ASL		
00/186C:0F 0F 07 0F	ORA	0F070F	
00/1870:00 03	BRK 03		
00/1872:04 07	TSB 07		
00/1874:08	PHP		
00/1875:0B	PHD		
00/1876:0C 0F A2	TSB A20F		
00/1879:00 8E	BRK 8E		

Examining the above code shows that once the password has been entered and its number has been determined the actual password the code wants to see is loaded and it is compared against the password number the user entered. As in the word protection check routine if the passwords do not match then vital code is not inserted into memory (in this case the code that allows the initialize disk option to operate).

That is all the analysis that must be done for this softkey. The softkey is to make the code think it passed the word protection check. The code that is inserted at location $A89A and $A89B must still be inserted. Also the code that is inserted at $50 and $51 must be inserted. The easiest way to accomplish this is to replace the call that runs the word protection check with a call to insert the code at locations $A89A and $A89B. Then after that call is done jump over the code that compares the password number to where the code is inserted at locations $50 and $51.

Step-by-step

1. Make a copy of the original disk using any fast copier.
2. Search for 20 74 19 AD 0D 1A CD 67 18. (I found mine starting at block $1B, byte $51 (track $3, sector $C, byte $51).
3. Replace the 20 74 19 AD 0D 1A with 20 13 1A 4C 5C 18.

This replaces the call to the word protection check and the loading of the password number with a call to insert the necessary code when the password is correct in the word protection check and a jump over the password number check to the code where the other vital code is inserted into memory).

4. Write the edits back to the copy and hide your original disk.

Michael A. Horton WA

EZ APT's with Compare Disk program

Computist readers I have a few questions for you. Do you like games that can save the game to disk to be continued later? Have you been unable to find the saved game data? Would you like a program that can point it out for you on almost any game?

Sounds too good to be true, right?! Wrong!!! Now you will be able to find that saved data easily and with very little effort.

There is actually a very simple way to find this saved data on a disk. All you do is make a back-up copy of the save disk, play the game until "something" changes such as losing a few hit points, save the game and then compare the two disks. Saving the game again after just 1 thing changes makes finding it easier. This technique works because once something in the game changes the data that is saved is different. We can then compare it to the starting data and the differences will be the information that has changed in the game. One nice thing about this is that it can be done over and over until you discover all the locations that you need. Using this technique makes APTing a game a breeze. I did this in order to get the locations I needed to create the Alternate Reality Character Editor in issue #55. Apparently some errors crept into the listing of the character editor. On lines 760, 1010, 1050, 1140, 1150 the number sign should be a raised to a power sign (change # to ^).

Now the comparison program comes in two parts. One is the machine language file (Compare Disks) that does the actual comparing and the other is a BASIC file (Compare Two Disks) that allows you to modify the parameters of the comparison program and save those modifications if you wish or to print out the differences. Compare Disks is a "stand alone" program. You can just BRUN it and it will compare the two disks according to the currently saved parameters. This program runs under DOS 3.3 only. If you modify DOS to read "protected disks" then this program can compare protected disks too. Compare Two Disks was made to allow the user to easily change the compare parameters or print out the differences. The limits allowed are :

parameter	values
Original Slot 1 to 6	6
Original Drive 1 or 2	1
Comparison Slot 1 to 6	6
Comparison Drive 1 or 2	1
First Track 0 to 35	0
Last Track 0 to 35	34
Track Increment 1 to 35	1

Now lets suppose you have 2 disk drives in slot 6 and wish to make the program remember that you want it to use both disk drives, answer the questions and when the computer asks if this is correct, type control-S and now these are saved on the disk. So all you have to do is BRUN the compare disks file and it will use the new parameters without you having to enter them in each time. If you modify DOS, the BASIC program might not be able to save the new parameters to disk.

If you need to modify DOS in order to compare the protected disk, first you must set the parameters and save them if they are different. Exit the program (ctrl-C). Bload compare disks. Make your modifications. Start Compare Disks (D00G).

One final note: to get a print out by BRUNing Compare Disks type in PR# (the slot number that the printer is in 1-6) and then BRUN the file. Don't forget to turn the printer off by typing PR#0 after you are done. Good luck to all of you out there and I hope you find this program useful, I know I have.

COMPARE TWO DISKS

```
10 D$ = CHR$ (4)
20 PRINT D$ "BLOAD◊COMPARE
  ◊DISKS"
30 HOME : VTAB 2
40 INPUT "ORIGINAL◊SLOT◊◊:"
  ;OS
50 IF OS < 1 OR OS > 6 OR OS
  < > INT (OS) THEN 40
60 INPUT "ORIGINAL◊DRIVE◊◊:"
  ;OD
70 IF OD < > 1 AND OD < > 2
  THEN 60
80 PRINT : INPUT "COMPARISON
  ◊SLOT◊:" ;CS
90 IF CS < 1 OR CS > 6 OR CS
  < > INT (CS) THEN 80
100 INPUT "COMPARISON◊DRIVE
  :" ;CD
110 IF CD < > 1 AND CD < > 2
  THEN 100
120 PRINT : PRINT
130 INPUT "FIRST◊TRACK◊◊◊◊:"
  ;FT
140 IF FT < 0 OR FT > 35 OR
  FT < > INT (FT) THEN 130
150 INPUT "LAST◊TRACK◊◊◊◊◊:"
  ;LT
160 IF LT < 0 OR LT > 35 OR
  LT < > INT (LT) THEN 150
170 IF LT < FT THEN 130
180 INPUT "TRACK◊INCREMENT
  ◊:" ;TI
190 IF TI < 0 OR TI > 35 OR
  TI < > INT (TI) THEN 180
200 PRINT : PRINT "ARE◊THESE
  ◊CORRECT(Y/N)"
210 AD = PEEK (43634) + PEEK
  (43635) * 256
220 LE = PEEK (43616) + PEEK
  (43617) * 256
230 GOSUB 420
240 IF A = 206 THEN POKE -
  16368,0: GOTO 30
250 IF A < > 217 AND A < >
  147 THEN 230
260 POKE AD + 3,OS: POKE AD
  + 4,OD
270 POKE AD + 5,CS: POKE AD
  + 6,CD
280 POKE AD + 7,FT: POKE AD
  + 8,LT
290 POKE AD + 9,TI
300 IF A = 147 THEN PRINT D$
  "BSAVE◊COMPARE◊DISKS,A"
  ;AD; ",L" ;LE: GOTO 230
310 PRINT : PRINT "DO◊YOU◊
  WANT◊A◊PRINTOUT(Y/N)"
320 GOSUB 420
330 IF A = 206 THEN 370
340 INPUT "SLOT◊#◊FOR◊PRINT
  ER:" ;SL
350 IF SL < 1 OR SL > 6 OR
  SL < > INT (SL) THEN 340
360 PRINT D$ "PR#" ;SL
370 POKE AD + 10,255
380 CALL AD
390 PRINT D$ "PR#0"
400 PRINT "COMPARISON◊IS◊
  DONE."
410 END
420 POKE - 16368,0
430 A = PEEK ( - 16384)
440 IF A < 128 THEN 430
450 RETURN
```

Checksums

```
10-$9E93   160-$5EE6   310-$D3DA
20-$C615   170-$8BB2   320-$C704
30-$832C   180-$1FB5   330-$E685
40-$B4DD   190-$BB0E   340-$BF8B
50-$B674   200-$0192   350-$1503
60-$BD31   210-$C8B0   360-$87E6
70-$5BA9   220-$3D6F   370-$979E
80-$9A18   230-$35E7   380-$AA55
90-$1BF4   240-$76EC   390-$784D
100-$5122  250-$14B1   400-$261A
110-$C04D  260-$D308   410-$A8B2
120-$EAEA  270-$A3C5   420-$8183
130-$A14D  280-$FACD   430-$8395
140-$1C1A  290-$44D3   440-$5ECA
150-$2087  300-$DFA1   450-$A90E
```

Compare Disks

```
0D00:4C 23 0D 60 01 60 01 00 $B923
0D08:22 01 00 00 00 AD 03 0D $46F8
0D10:CD 05 0D D0 0B AD 04 0D $1E1B
0D18:CD 06 0D D0 03 A9 00 60 $B463
0D20:A9 FF 60 A9 00 85 01 85 $AE88
0D28:03 20 0D 0D F0 06 20 58 $0AA5
0D30:FC 20 68 0E AD 07 0D 85 $CCD1
0D38:00 A9 07 8D 0C 0D 20 0D $43F8
0D40:0D D0 06 20 58 FC 20 4E $B496
0D48:0E A5 00 8D 79 0F A9 10 $644D
0D50:8D 43 0F 20 44 0F AD 79 $24EC
0D58:0F 18 6D 09 0D 8D 79 0F $B8C3
0D60:C9 24 B0 15 CD 08 0D F0 $DFBE
0D68:02 B0 0E AD 43 0F 18 69 $A5B6
0D70:10 8D 43 0F CE 0C 0D D0 $B305
0D78:DA A9 10 85 02 A9 07 8D $061E
0D80:0C 0D 20 0D 0D D0 06 20 $8107
0D88:58 FC 20 5B 0E A5 00 8D $312E
0D90:79 0F A9 80 8D 43 0F 85 $DDA8
0D98:04 20 44 0F A0 00 B1 01 $BDEE
0DA0:D1 03 F0 03 20 07 0E C8 $C526
0DA8:D0 F4 AD 0B 0D F0 0B A9 $1ADE
0DB0:8D 20 ED FD 20 ED FD 20 $C069
0DB8:75 0E A9 00 8D 0B 0D E6 $8D3A
0DC0:02 E6 04 A5 04 C9 90 D0 $28F1
0DC8:D5 AD 79 0F 18 6D 09 0D $C469
0DD0:8D 79 0F C9 24 B0 27 CD $9198
0DD8:08 0D F0 02 B0 20 CE 0C $C1E1
0DE0:0D D0 AF A5 00 18 A0 07 $347E
0DE8:6D 09 0D 88 D0 FA 85 00 $57A8
0DF0:C9 24 B0 0A CD 08 0D F0 $6BEE
0DF8:02 B0 03 4C 39 0D AD 0A $22EE
0E00:0D F0 01 60 4C D0 03 84 $BB7B
0E08:07 AD 0B 0D D0 33 A9 FF $FBEE
0E10:8D 0B 0D A9 27 85 05 A9 $5679
0E18:0F 85 06 20 8E 0E AD 79 $E53C
0E20:0F 20 DA FD A9 2F 85 05 $AAC2
0E28:A9 0F 85 06 20 8E 0E A5 $BFCF
0E30:02 29 0F 20 DA FD A9 39 $797B
0E38:85 05 A9 0F 85 06 20 8E $7379
0E40:0E A5 07 20 DA FD A9 A0 $4E4F
0E48:20 ED FD A4 07 60 A0 00 $9095
0E50:B9 9B 0E F0 2D 99 A8 05 $D8BC
0E58:C8 D0 F5 A0 00 B9 C1 0E $ADCD
0E60:F0 20 99 A8 05 C8 D0 F5 $FCCF
0E68:A0 00 B9 E9 0E F0 13 99 $32C1
0E70:A8 05 C8 D0 F5 A0 00 B9 $A088
0E78:0C 0F F0 06 99 D0 07 C8 $389C
0E80:D0 F5 8D 10 C0 AD 00 C0 $00B9
0E88:10 FB 20 58 FC 60 A0 00 $E77A
0E90:B1 05 F0 06 20 ED FD C8 $6971
0E98:D0 F6 60 C9 CE D3 C5 D2 $C02D
0EA0:D4 A0 CF D2 C9 C7 C9 CE $94ED
0EA8:C1 CC A0 C4 C9 D3 CB A0 $C2B1
0EB0:C1 CE C4 A0 D0 D2 C5 D3 $DBD2
```

```
0EB8:D3 A0 C1 A0 CB C5 D9 AE  $BCFD
0EC0:00 C9 CE D3 C5 D2 D4 A0  $754C
0EC8:C3 CF CD D0 C1 D2 C9 D3  $F411
0ED0:CF CE A0 C4 C9 D3 CB A0  $955C
0ED8:C1 CE C4 A0 D0 D2 C5 D3  $BC2F
0EE0:D3 A0 C1 A0 CB C5 D9 AE  $EB10
0EE8:00 C9 CE D3 C5 D2 D4 A0  $12B1
0EF0:C2 CF D4 C8 A0 C4 C9 D3  $F57E
0EF8:CB D3 A0 C1 CE C4 A0 D0  $1F86
0F00:D2 C5 D3 D3 A0 C1 A0 CB  $7593
0F08:C5 D9 AE 00 D0 D2 C5 D3  $2A53
0F10:D3 A0 C1 CE D9 A0 CB C5  $7DA0
0F18:D9 A0 D4 CF A0 C3 CF CE  $9AF9
0F20:D4 C9 CE D5 C5 AE 00 D4  $D91F
0F28:D2 C1 C3 CB A0 A4 00 A0  $2DF9
0F30:D3 C5 C3 D4 CF D2 A0 A4  $AD0F
0F38:00 A0 C2 D9 D4 C5 A8 D3  $6160
0F40:A9 BA 00 10 A9 0F 8D 7A  $62D8
0F48:0F 18 6D 43 0F 8D 7E 0F  $EC59
0F50:20 65 0F AC 7A 0F AD 82  $BEB1
0F58:0F 99 8A 0F CE 7E 0F CE  $3244
0F60:7A 0F 10 EC 60 A9 0F A0  $A616
0F68:75 18 20 D9 03 B0 05 A9  $FD3B
0F70:00 8D 82 0F 60 01 60 01  $526C
0F78:00 00 00 86 0F 00 20 00  $27CE
0F80:00 01 00 00 60 01 00 01  $328E
0F88:D8 EF 00 00 00 00 00 00  $EF0B
0F90:00 00 00 00 00 00 00 00  $5FFB
0F98:00 00                    $55C5
```

COMPARE DISKS.SOURCE

```
         .OR  $0D00
         .TF  COMPARE DISKS
DOS.EXIT .EQ  $03D0
RWTS     .EQ  $03D9
KEYBOARD      .EQ    $C000
CLR.KEYBOARD  .EQ    $C010
CLEAR.SCREEN  .EQ    $FC58
HEX.OUT  .EQ  $FDDA
CHAR.OUT      .EQ    $FDED
BASE.TRACK    .EQ    $00
POINTER  .EQ  $01,02
POINTER2      .EQ    $03,04
MESSAGE       .EQ    $05,06
BYTE     .EQ  $07
START    JMP START1
SOURCE.SLOT.HS     60
SOURCE.DRIVE       .HS    01
TARGET.SLOT .HS    60
TARGET.DRIVE       .HS    01
STARTING.TRACK     .HS    00
LAST.TRACK    .HS    22
TRACK.INCREMENT    .HS    01
AM.I.BEING.CALLED  .HS    00
DIFFERENT.FLAG     .HS    00
TRACKS.TO.READ     .HS    00
IS.DRIVE.SAME
         LDA SOURCE.SLOT
         CMP TARGET.SLOT
         BNE .1
         LDA SOURCE.DRIVE
         CMP TARGET.DRIVE
         BNE .1
         LDA #$00
         RTS
.1       LDA #$FF
         RTS
START1
         LDA #$00
         STA POINTER
         STA POINTER2
         JSR IS.DRIVE.SAME
         BEQ .1
         JSR CLEAR.SCREEN
         JSR INSERT.DISKS
.1       LDA STARTING.TRACK
         STA BASE.TRACK
LOOP
         LDA #$07
         STA TRACKS.TO.READ
         JSR IS.DRIVE.SAME
         BNE .1
         JSR CLEAR.SCREEN
         JSR PRINT.ORIG
.1       LDA BASE.TRACK
         STA TRACK
         LDA #$10
         STA PAGE
.2       JSR READ.TRACK
         LDA TRACK
         CLC
         ADC TRACK.INCREMENT
         STA TRACK
         CMP #$24
         BGE READ.2ND.DISK
         CMP LAST.TRACK
         BEQ .3
         BGE READ.2ND.DISK
.3       LDA PAGE
         CLC
         ADC #$10
         STA PAGE
         DEC TRACKS.TO.READ
         BNE .2
READ.2ND.DISK
         LDA #$10
         STA POINTER+1
         LDA #$07
         STA TRACKS.TO.READ
         JSR IS.DRIVE.SAME
         BNE .1
         JSR CLEAR.SCREEN
         JSR PRINT.COMP
.1       LDA BASE.TRACK
         STA TRACK
.2       LDA #$80
         STA PAGE
         STA POINTER2+1
         JSR READ.TRACK
         LDY #$00
.3       LDA (POINTER),Y
         CMP (POINTER2),Y
         BEQ .4
         JSR DIFFERENT
.4       INY
         BNE .3
         LDA DIFFERENT.FLAG
         BEQ .5
         LDA #$8D
         JSR CHAR.OUT
         JSR CHAR.OUT
         JSR PRESS.KEY
.5       LDA #$00
         STA DIFFERENT.FLAG
         INC POINTER+1
         INC POINTER2+1
         LDA POINTER2+1
         CMP #$90
         BNE .3
         LDA TRACK
         CLC
         ADC TRACK.INCREMENT
         STA TRACK
         CMP #$24
         BGE DOS
         CMP LAST.TRACK
         BEQ .8
         BGE DOS
.8       DEC TRACKS.TO.READ
         BNE .2
         LDA BASE.TRACK
         CLC
         LDY #$07
.6       ADC TRACK.INCREMENT
         DEY
         BNE .6
         STA BASE.TRACK
         CMP #$24
         BGE DOS
         CMP LAST.TRACK
         BEQ .7
         BGE DOS
.7       JMP LOOP
DOS      LDA AM.I.BEING.CALLED
         BEQ .1
         RTS
.1       JMP DOS.EXIT
DIFFERENT
         STY BYTE
         LDA DIFFERENT.FLAG
         BNE .1
         LDA #$FF
         STA DIFFERENT.FLAG
         LDA #TRACK.MESS
         STA MESSAGE
         LDA /TRACK.MESS
         STA MESSAGE+1
         JSR PRINT.MESSAGE
         LDA TRACK
         JSR HEX.OUT
         LDA #SECTOR.MESS
         STA MESSAGE
         LDA /SECTOR.MESS
         STA MESSAGE+1
         JSR PRINT.MESSAGE
         LDA POINTER+1
         AND #$0F
         JSR HEX.OUT
         LDA #BYTE.MESS
         STA MESSAGE
         LDA /BYTE.MESS
         STA MESSAGE+1
         JSR PRINT.MESSAGE
         LDA BYTE
.1       JSR HEX.OUT
         LDA #$A0
         JSR CHAR.OUT
         LDY BYTE
         RTS
PRINT.ORIG
         LDY #$00
.1       LDA ORIG,Y
         BEQ WAIT.FOR.KEY
         STA $05A8,Y
         INY
         BNE .1
PRINT.COMP
         LDY #$00
.1       LDA COMP,Y
         BEQ WAIT.FOR.KEY
         STA $05A8,Y
         INY
         BNE .1
INSERT.DISKS
         LDY #$00
.1       LDA BOTH,Y
         BEQ WAIT.FOR.KEY
         STA $05A8,Y
         INY
         BNE .1
PRESS.KEY
         LDY #$00
.1       LDA PRESS,Y
         BEQ WAIT.FOR.KEY
         STA $07D0,Y
         INY
         BNE .1
WAIT.FOR.KEY
         STA CLR.KEYBOARD
.1       LDA KEYBOARD
         BPL .1
         JSR CLEAR.SCREEN
         RTS
PRINT.MESSAGE
         LDY #$00
.1       LDA (MESSAGE),Y
         BEQ .2
         JSR CHAR.OUT
         INY
         BNE .1
.2       RTS
ORIG     .AS  -"INSERT ORIGINAL DISK AND
                PRESS A KEY."
         .HS  00
COMP     .AS  -"INSERT COMPARISON DISK
                AND PRESS A KEY."
         .HS  00
BOTH     .AS  -"INSERT BOTH DISKS AND
                PRESS A KEY."
         .HS  00
PRESS    .AS  -"PRESS ANY KEY TO CON-
                TINUE."
         .HS  00
TRACK.MESS
         .AS  -"TRACK $"
         .HS  00
SECTOR.MESS
         .AS  -" SECTOR $"
         .HS  00
BYTE.MESS
         .AS  -" BYTE(S):"
         .HS  00
PAGE     .HS  10
READ.TRACK
         LDA #$0F
         STA SECTOR
         CLC
         ADC PAGE
         STA BUFFER
READ     JSR READ.SECTOR
         LDY SECTOR
         LDA ERROR
         STA ERROR.TABLE,Y
         DEC BUFFER
         DEC SECTOR
         BPL READ
         RTS
READ.SECTOR
         LDA /I.O.BLOCK
         LDY #I.O.BLOCK
         CLC
         JSR RWTS
         BCS .1
         LDA #00
         STA ERROR
.1       RTS
I.O.BLOCK
         .HS  01     TABLE TYPE
         .HS  60     SLOT #*16
         .HS  01     DRIVE #
         .HS  00     VOLUME #
TRACK    .HS  00     TRACK #
SECTOR   .HS  00     SECTOR #
         .DA  DCT    ADDRESS OF DCT
         .HS  00     LOW BYTE OF BUFFER
BUFFER   .HS  20     HIGH BYTE OF BUFFER
         .HS  00
         .HS  00
         .HS  01     COMMAND (01 = READ,
                     02 = WRITE)
ERROR    .HS  00     WHAT ERROR OC-
                     CURRED
         .HS  00
         .HS  60     LAST SLOT # USED
         .HS  01     LAST DRIVE # USED
DCT      .HS  00
         .HS  01     PHASES PER TRACK
         .HS  D8EF   TIME ON COUNT
ERROR.TABLE
         .HS  00000  000
         .HS  00000  000
         .HS  00000  000
         .HS  00000  000
```

Rich Etarip — WI

Bug in Gorgon Softkey

In issue #82, page 10, column 4, near the end of paragraph 4, it says "If you have the right sector you'll see $00 $20 $FB $27.". It should read "$04 $20 $FB $27".

Bugs in Captain Goodnight Softkey

I have discovered two bugs in the softkey procedure:

Page 15, column 4, the 6 line hex listing for 9400: the 5th line should start with **1B** not 18.

Page 16, column 1, the 14 line hex listing for 8000: the 12th line should start with 60 **8D** FF BF not 60 BD FF BF.

The softkey refers to listing 1 & 2, neither of which were printed. They are not necessary but just there for verifying the assembly listing after keying in the hexdump.

Boot code tracing Star Maze

Softkey for...

Star Maze

Sir-Tech

Here is another Most Wanted List Softkey. Star Maze is well protected as far as the disk format. The data is encoded in 4+4, contains a six byte data header and it loads similar to the Sierra On-Line spiradisc protection. However, the boot code is very simple to trace and the program can be captured as a single file.

Begin the boot code trace by entering the monitor.

CALL-151

As always, the DOS boot routine must me moved to RAM to modify.

9600<C600.C6FFM

Change the JMP $801 to JMP $9801 and at $9801 put a JMP $FF59. Then execute $9600 to load in boot stage 1.

96FA:98 N 9801:4C 59 FF

9600G

Unlike most disks, the boot 1 on this disk is 4 pages long. In most cases, I would move boot 1 from $800 to $9800 to be altered but since this boot 1 is 4 pages long, it's too much trouble to try to relocate it. Instead, we can leave right where it is at $800 and change $9600 to load the normal boot 1 into $9800 so ours isn't overwritten.

9659:98

Listing through $800, you'll see a memory move followed by what appears to be incomplete assembly code. What it really is is the infamous self-modifying code. There is nothing important within the self-modifying code but it appears to be used to hide the JMP $5FD0 at $860. This is the jump to the entry point of the game program. It can

easily be changed to jump to the monitor.

861:59 FF

Boot 0 at $9600 is still exiting into $9801 so change it to go to $801 where our modified boot 1 is at. At this point, the boot code is ready to load in the game.

96FA:08
9600G

The game is now in memory and it's time to reboot DOS so it can be saved. Insert a slave disk with NO HELLO program for saving but first, move page $8 to safe memory. After rebooting, enter the monitor and move page $8 back.

8000<800.8FFM
C600G
CALL-151
800<8000.80FFM

The game is almost ready to be saved but a few changes must be made first. The game accesses the disk for high scores so these routines must be disabled. Put an RTS at the start of each routine.

3DD3:60
3E5D:60

The high score buffer is at $AEC0 and it should be cleared because the high scores will no longer be read from the disk. Otherwise, the high scores will come up as 'garbage'. There is already a routine in the program at $3E4F to clear the buffer so at $7FA we'll call this routine and then jump to $5FD0 to start the game.

7FA:20 4F 3E 4C D0 5F
BSAVE STAR MAZE,A$7FA,L$7406

All done!

Softkey for...

Rings of Saturn

?

Requirements:

Locksmith Fast Disk Backup (or a copy program that bypasses errors)
Any copy program that copies select tracks
Sector editor

Even though Rings of Saturn was a 1981 release (just about the time I was learning about deprotection) the protection is still enough to keep the average person stumped. In my first several attempts to crack this disk I came up empty. Tracks 3 through 22 are normal with the exception of sectors $E and $F on track $13 and $14. Tracks $1 and $2 contain typical 6+2 data however, only sector 0 has an address header. This would cause some difficulty in reading these tracks. However, by further examination, I concluded that tracks $1 and $2 only contain the DOS of the disk which will not be needed for the finished product. There would be quite a bit more involved if these tracks were needed but fortunately, they're not. Sector $E of Track $14 contains the 'HELLO' program of the disk and by altering DOS, I was able to read it in. All it does is read Track $16 Sector $0 into $200 and JMP to it. It turns out that everything that is needed for this disk to run is out in the open on the normal tracks.

Step 1 is copying the normal tracks of the disk. Locksmith FDB (mentioned above) works best for this. After copying the disk, run a copier that allows you to select tracks. Copy track 0 from any normal DOS disk. The Apple Master disk WILL NOT work for this because it contains a different boot code. Once track 0 is copied, run your sector editor. The following edits will cause the boot code to read Track $16 Sector $0 into $200 and JMP to it.

Trk	Sct	Byte	From	To
$00	$01	EC ED	?? ??	16 00
		F0 F1	?? ??	00 02
		F4	??	01

At byte 0 of this same sector, enter the following:

Trk	Sct	Byte	From	To
$00	$01	$00	??	A9 B7 A0 E8 20
			??	B5 B7 4C 00 02

Then rewrite the sector. The entry point for the Rings of Saturn RWTS is $B6D3. Instead of searching the disk and changing every JSR $B6D3 to JSR $B7B5, there is a much simpler method. Since Track $0 Sector $0 is loaded into $B600 at boot, and there is nothing important at $B6D3 in this sector, simply write a JMP $B7B5 at $B6D3 and it will work just fine. Read Track $0 Sector $0 and at byte $D3 enter '4C B5 B7' and write the sector back to the disk.

Trk	Sct	Byte	From	To
$00	$00	D3	??	4C B5 B7

That's all there is to it. Now you can scratch Rings of Saturn from the Most Wanted List.

Softkey for...

Axis Assassin

Electronic Arts

I first expected Axis Assassin to be protected much like Archon, Skyfox, and many other Electronic Arts (EOA) games, but it is not. The disk format is rather basic and somewhat consistent with the other EOA games but the secondary protection is extensive. That might be why it's on the Most Wanted List. The only two normal tracks on the disk are tracks 0 and 21 and track 21 is blank so it need not be copied. Tracks $01 to $1F have the typical EOA altered data marks of D5 BB CF. Note that there may be other versions of Axis Assassin with different protection.

The first step in cracking this disk is copying the disk onto a normal format. Super IOB will work for this purpose. Install the provided Axis Assassin controller into Super IOB and proceed to copy the disk. When the copy is finished, reboot DOS and run your sector editor. Super IOB could have been written to perform the sector edits but I will be explaining each sector edit as we go along so it will be more easily understandable.

CONTROLLER

```
1000 REM "AXIS ASSASSIN
1010 TK = 0:ST = 0:LT =
  32:CD = WR
1020 T1 = TK: GOSUB 490
1025 IF TK= 1 THEN GOSUB 210
1030 GOSUB 430: GOSUB 100:ST
  =ST+1:IF ST< DOS THEN 1030
1040 IF BF THEN 1060
1050 ST = 0:TK = TK + 1: IF
  TK < LT THEN 1025
1060 GOSUB 490:TK= T1:ST = 0
1070 GOSUB 430: GOSUB 100:ST
  =ST+1:IF ST< DOS THEN 1070
1080 ST= 0:TK= TK + 1: IF BF
  = 0 AND TK < LT THEN 1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "DONE◊WITH
  ◊COPY" : END
5000 DATA 213,187,207
```

Checksums

```
1000-$356B  1040-$557B  1090-$BD55
1010-$E3B7  1050-$70E5  1100-$A24D
1020-$DC17  1060-$7823  5000-$E663
1025-$E591  1070-$7026
1030-$F390  1080-$8F3C
```

The first changes to be made to the disk are in the EOA loader so it reads the normal format. Don't forget to re-write the sector.

Trk	Sct	Byte	From	To
$00	$09	$39	BB	AA
$00	$09	$43	CF	AD

At this point you could boot the disk and it would work up to a point. Then you would get a -beep- and the wonderful message 'ERR'. This is EOA's secondary protection hard at work. It was time to find the disk check and disable it. I started by patching a JMP $FF59 into the boot code so it would give me control once the loader was in memory. By listing through the code and examining it closely, I found some- thing suspicious in the $BB00 area. It was reading from the disk, comparing memory locations, and branching conditionally...all of the ingredients for a disk check recipe. This appeared to be what I was looking for so I began making modifications. At the tail end of the routine ($BBE3), two values are pulled off the stack followed by an RTS. This is where it appears to go upon a successful disk check. Otherwise, at $BBAD, it returns without pulling the two values off the stack and because the stack determines the RTS address, it returns somewhere else. To bypass this problem, always pull the two values from the stack before returning and it will work. The changes must be made to $BB00 and it can be found on Track 0, Sector 2.

Trk	Sct	Byte	From	To
$00	$02	$AD	60	68
		$AE	A5	68
		$AF	00	60

Now if you were to boot the disk it would boot completely and the game would start...BUT...There is yet another disk check between each level of the game so this must be disabled. I searched the disk for accesses to the disk read address ($C0EC or $C08C,X depending on how they use it) and found several on Track $0E. After some examining, the main check routine is on Track $0E, Sector $05, byte $14. This is just a disk check and does not return with a checksum to the call routine. This means that it can be disabled with an RTS.

Trk	Sct	Byte	From	To
$0E	$05	$14	20	60

After this modification, I booted the disk and crossed my fingers. When the game loaded in and the title page came up, it rebooted. This meant one thing...secondary-secondary protection. In other words, checking the disk check routine to make sure it hasn't been tampered with. This is common with EOA software. The check routine shouldn't be too hard to find because it happens right at the start of the program. I started by jumping to $FF59 instead of the start of the game (Track 0-Sector 0-byte $44). Then I traced through the start of the game until I found the check routine. I later found out that there were several checks to different parts of the game and an unsuccessful check would cause a JMP to $5B2E which reboots. The routines I found would either RTS (if no error) or JMP to the reboot routine. This means that instead of changing them all, you can just put an RTS at $5B2E. I attempted to do this but could not find the reboot routine on the disk. There was yet another memory check in the $4300 area and I could not find it either. It didn't surprise me a bit that EOA encoded a large portion of the program. There are two ways to deal with this problem. The easiest way would be to patch a little routine at the start of the program to make the two modifications before starting the game, but, being as stubborn as I am, I decided to hunt down the encoded memory and change it right on the disk. It would take pages to explain the entire process of finding and decoding the memory but here is a quick run down of what I did. I interrupted the boot code at a point where the memory was still encoded and marked down the values in the locations I needed to change. Then I booted the disk, entering the monitor when the game was loaded in and checked the values in the decoded form. By having the value in its encoded as well as decoded form, I was able to arrive at the Exclusive-OR value used to encode each byte. I then took the values I needed for disabling the check routine and used the encoding value to get the proper result. Here are the final sector edits:

Trk	Sct	Byte	From	To
$08	$04	$2E	13	D3
$07	$0C	$E9	AF	47

That's it! Your working copy of Axis Assassin. All total, it took about six hours of work to arrive at ten minutes worth of Softkey procedure. Until next time...Keep Cracking!

Softkey for...

Keyboarding Klass
Math Facts Tracker

Mastery Development

This Softkey will instruct you on how to deprotect two disks released by Mastery Development. Both disks contain the same format and the protection is similar. Tracks 0-2 are normal and tracks 3-22 have the address/data prologue bytes flipped. Where D5 AA AD is normal for data, they changed it to D5 AA 96 which is normal for address and vice-versa.

This Softkey was done on the demo copies of these games and I can't guarantee that it will work on the actual manufacturer disk. It's very likely, though, that they are protected the same.

Both disks are formatted the same way so you can copy them both with the Super IOB controller listed at the end of the article. Once the copy is made, a few sector edits must be done.

Even though the disk (original) has two formats, they use the same RWTS for the whole disk and they self-modify it depending on what part of the disk it is reading. As mentioned above, the only address/data mark changed was the $96 to $AD and vice versa.

MATH FACTS TRACKER

Trk	Sct	Byte	From	To
00	0E	5F	96	AD
00	0E	61	AD	96
1F	0D	F1	AD	96

KEYBOARDING KLASS

Trk	Sct	Byte	From	To
00	00	49	AD	96
00	00	50	96	AD
21	0A	6C	03	23

To the best of my knowledge, the backups work but I really did not test them in-depth. There may be secondary protection along the way but there does not appear to be.

CONTROLLER

```
1000 REM "MASTERY
  DEVELOPMENT
```

```
1010 TK=0:ST=0:LT= 35:CD= WR
1020 T1 = TK: GOSUB 490
1025 IF TK > 2 THEN POKE
  47356,150: POKE 47466,173
1030 GOSUB 430: GOSUB 100:ST
  =ST+1:IF ST< DOS THEN 1030
1040 IF BF THEN 1060
1050 ST = 0:TK = TK + 1: IF
  TK < LT THEN 1025
1060 GOSUB 230: GOSUB 490:TK
  = T1:ST = 0
1070 GOSUB 430: GOSUB 100:ST
  =ST+1:IF ST< DOS THEN 1070
1080 ST= 0:TK= TK +1: IF BF=
  0 AND TK < LT THEN 1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "DONE◊WITH
  ◊COPY" : END
```

Checksums

```
1000-$356B  1030-$4978  1070-$BF01
1010-$3266  1040-$7727  1080-$A8BD
1020-$C11A  1050-$C27D  1090-$76E6
1025-$5F79  1060-$B704  1100-$61F0
```

Softkey for...

Bandits

Sirius

"They said it couldn't be done..." quoting the anonymous crackist, supposedly the first to crack BANDITS by Sirius Software. This disk had been on the Most Wanted List for quite some time back in the 80's but, for some reason, no longer is. Bandits is about the toughest crack I've done to date, but with a good amount of work, it CAN be done. That's what this long awaited Softkey article will show you how to do. Some of the procedure is quite involved and may be hard to understand, but if you have the ability to follow instructions carefully, this will lead you to a cracked copy of Bandits. I will try my best to explain the Softkey procedure but to truly understand parts of it, it takes a considerable knowledge of DOS. I always recommend the book 'Beneath Apple DOS'.

To explain things a bit, the Bandits disk is encoded in 4+4 (what Sirius disk isn't?). The data is not split into sectors but could be considered as one large sector that is $C00 bytes long after decoding. To make things worse, the data is not directly encoded in 4+4. What I mean by this is that the data is byte encoded by means of Exclusive- OR before it is nibble encoded on the disk. Plus, there are several checksum bytes stuck in the raw disk data along the way. Even though Bandits is a 1981 release, I've yet to find a bit copier that can copy it. A long look at the Bandits main loader proved it to be quite a complex loading process. I never came to fully understand exactly how the loader works but with a lot of snooping around, I was able to figure out how to use it to read any desired range of tracks. This made downloading the disk data to a normal disk quite possible. It's almost ironic that Sirius made the disk format so complex but made the loader so easy to use. We can copy the disk but first we have to get our hands on the loader.

This is accomplished by doing a partial boot code trace. Then using their loader to read, and the RWTS to write, we will do a manual copy of the entire disk onto a normal format. From there, we will write a compact loader for the disk using some of the RWTS routines. In most cases, you can use the normal RWTS right where it is but in this case, the RWTS area is occupied by data. We have just the text page area ($400-$7FF) to work with for the loader. The entire process may be a bit time consuming but may be worth it to you. Be sure to follow the procedure VERY CAREFULLY. One minor mistake could lead to a non-working copy and a lot of work gone down the drain. The first step (as mentioned above) is getting our hands on the game loader.

CALL-151
9600<C600.C6FFM
96FA:98 N 9801:AD E8 C0 4C 59 FF
9600G
9800<800.8FFM
982F:64
9859:68
986D:59 FF
9600G

By following the above steps, the loader should be in memory at $6400. It normally loads into $400 but you can't work with it in the text page. Reboot a slave disk and save the loader for safe keeping. (the disk you boot MUST contain a normal Apple RWTS)

C600G
BSAVE LOADER,A$6400,L$400

You will need a blank initialized disk for the copy so initialize a disk if you haven't already. Then insert the Bandits disk and enter the monitor.

CALL-151

We will be using memory from $1000 to $9FFF for reading the disk so move the loader to $800, disconnect DOS, and clear memory.

800<6400.67FFM
FF59G
1000:00 N 1001<1000.9FFFM

A few modifications must be made to the loader so it returns to the monitor and doesn't check for errors.

965:2C
B22:2C
B29:59 FF

At $BC6 ($7C6) is a $22 byte table that specifies where in memory to load each track. We will be reading 8 tracks at a time so the table should tell it to read into $1000 through $8FFF. Track 1 uses the value in $BC7, track 2 uses the value in $BC8, and etc.

BC7:10 20 30 40 50 60 70 80 10 20 30
:40 50 60 70 80 10 20 30 40 50 60
:70 80 10 20 30 40 50 60 70 80 90

Enter this data continuously without pressing return until you've typed the last byte. You'll notice at the end there is a 90. That's because in the final copy pass we will be reading 9 tracks instead of 8. If we were to read 4 passes of 8 tracks, one stray track would be left to copy. Now enter a small routine that moves the loader to its proper place, specifies the track, and calls the loader.

B000:A2 00 BD 00 08 9D 00 04
:BD 00 09 9D 00 05 BD 00
:0A 9D 00 06 BD 00 0B 9D
:00 07 E8 D0 E5 A9 03 85
:57 A9 12 8D 37 04 A9 60
:85 34 20 00 B6 20 00 04
:4C 59 FF
B600<C600.C6FFM
B654:60

We will be using part of the $C600 boot program to recalibrate the drive arm to track 0 so the Bandits loader can seek the correct track. It will also be called before writing to the normal disk. The routine was moved to $B600 and an RTS placed in the routine so it returns after the disk arm is moved.

If you list the routine at $B000, it is storing in locations $57, $437 and $34. Location $57 holds the start track, times 2. When using the track seek routine to move the disk arm, you have to multiply the track by two before calling it. The Bandits data is written on half-tracks so track 1.5 times 2 equals 3. Track 2.5 times 2 equals 5, and so on and so forth...

Location $437 is for the end track. When the loader steps to the next track, it INCrements $57 twice and then compares it to $437. If $437 contains $FF, it reads to the end of the disk. Location $34 is simply the slot number times $10 and should be $60.

We're ready for the first read pass so execute $B000. The text screen will fill with clutter and lo-res graphics will turn on but don't be alarmed because it's supposed to do that. The disk drive will also recalibrate as if it were booting but it is just seeking track 0 so the loader knows where it is.

B000G

Tracks $1.5 to $8.5 are now in at $1000-8FFF. The RWTS should still be intact and we're going to use it to write out the disk data. The multi-sector read/write routine is at $B793 but we'll add a few loads and stores to it so we don't have to keep entering the same IOB information every time we call it.

B77C:20 00 B6 A9 00 8D 47 04
:A9 0F 8D ED B7 A9 80 8D
:E1 B7 A9 8F 8D F1 B7

Cause the RWTS to exit to the monitor reset routine after writing.

B79A:00 BD
B7B4:4C 59 FF

Now enter the remaining IOB data. The tracks, sectors, and pages will all be written backward.

B7EB:00 08
B7F0:00
B7F4:02

Insert the copy disk and call $B77C to write it. Once again the disk drive will sound like it is rebooting but just to seek track 0.

B77CG

Insert the Bandits disk again and enter the next range of tracks.

B01E:13
B022:22
B000G

Insert the copy disk...

B7EC:10
B77CG

Insert Bandits...

B01E:23
B022:32
B000G

Insert the copy disk...

B7EC:18
B77CG

Insert Bandits...

B01E:33
B022:FF
B000G

Insert the copy disk...

B7EC:21
B78A:90
B78F:9F
B77CG

If you have gotten this far, the copy is complete! Now that the game is on a normal disk format, their loader must be reconstructed to read this format.

Writing the DOS 3.3 BANDITS loader

Unfortunately, as mentioned earlier, the RWTS area is occupied by program data. This is quite an inconvenience. In such a situation, one must either relocate the RWTS when space permits, or write a compacted DOS. This is what we are going to have to do. Start by rebooting a slave disk and loading the program 'LOADER' which we saved earlier. The program should load into $6400.

Four sections must be taken from DOS. They are:

Read address routine	$B944
Read data routine	$B8DC
Post-nibble routine	$B8C2
Read translate table	$BA96

Enter the monitor and move the DOS routines into the loader area.

CALL-151
64B8<B8C2.B99FM
6596<BA96.BAFFM

The DOS 6+2 encoding requires 2 buffers for the decoding. We will use $600 for one and the destination buffer (address is stored in $3E and $3F) for the other. Several modifications must be made to these relocated DOS routines for them to work with Bandits. Type VERY carefully...

64BF:B1 3E EA
64C4:06
64C8:06
64CD:EA EA
64F9:41
6501:05
6503:41
6506:06
650A:41
6512:05
6514:41 91 3E EA
6538:18
653D:41
6542:41
6565:EA EA
656E:41
6575:41

Write a routine at $6700 to call the DOS routines to read in a track, then enter the sector skew at $67E9. Compare $6700 with listing 1.

6700:20 3A 05 A4 2D B9 E9 07
:C5 42 D0 F4 A9 00 85 3E
:A5 30 85 3F 20 D2 04 20
:B8 04 C6 30 A2 60 C6 42
:10 DE 60 EA EA
67E9:00 07 0E 06 0D 05 0C 04
:0B 03 0A 02 09 01 08 0F

The main section of the loader must be altered to call our track read routine. How this part works is rather simple. It compares the current track (location $57) to the end track ($437). If greater than or equal, it turns off the drive and exits. Otherwise, it calls the track seek routine ($72B) and then the track load routine ($700). Finally, it increments the track and jumps back to the beginning.

643C:C6 57
643E<648C.64A6M
644A:EA EA
6455:25
6459:18 69 0B 85 30 A9 0B 85
:42 A5 57 A6 34 20 2B 07
:20 00 07 E6 57 E6 57 4C
:3E 04

The loader is complete so save it to a slave disk (not the Bandits copy) to be safe.

BSAVE LOADER2,A$6400,L$400

The next step is to write the loader to track 0 of the Bandits copy. This is technically the 2nd stage boot loader. Insert the copy disk. We will use the multi-sector routine again to write to the disk. First, enter the correct IOB information:

B7EB:00 00 04
B7F0:00 67 00 00 02
B7E1:04

This tells the loader to write to track 0 sector 4 and write 4 pages of memory. It's ready to write so call the routine.

B793G

Now run your sector editor for the final step of the Softkey. Track 0 sector 0 (boot 1) has to be modified to read in the loader. For some reason, if you read the loader directly into $400 from the disk, it takes about 10 times as long to read in, so instead, read it into $2400 and move it down to $400 before jumping to it.

Trk	Sct	Byte	From	To
$00	$00	$4A	??	4C B0
$00	$00	$FE	??	23 04

The boot stage is now jumping to $8B0 which is where we will enter the following routine to turn on hi-res page 2, memory move the loader from $2400 down to $400, and JMP to $41F to load the game. Begin entering at byte $B0.

AD 50 C0 AD 57 C0 AD 52
C0 AD 55 C0 A0 00 B9 00
24 99 00 04 B9 00 25 99
00 05 B9 00 27 99 00 07
C8 D0 EB 4C 1F 04

Finally, rewrite the sector....and there you have it! The cracked copyable version of BANDITS! Hopefully everything works correctly for you. If not, you may have possibly made en error along the way which is easy to do in a long procedure such as this. The boot process will take a bit longer than on the original disk because normal sectorized loading is not quite as fast as the 4+4 Bandits format. Have fun!!!

Listing 1

```
6700-  JSR $053A     Read addr field from track
6703-  LDY $2D       The sector # from address
                     field is stored in $2D
6705-  LDA $07E9,Y   Load from sector skew
6708-  CMP $42       Is it sector we're looking
                     for
670A-  BNE $6700     Not correct sector, try
                     again
670C-  LDA #$00      Correct sector found,
                     store a $00 in low order
670E-  STA $3E       byte of destination buffer
6710-  LDA $30       Take high order byte of
                     destination buffer from
6712-  STA $3F       $30 and store in $3F for
                     the read routines
6714-  JSR $04D2     Read data field from disk
6717-  JSR $04B8     Decode the nibblized data
                     into buffer
671A-  DEC $30       Decrement buffer (data is
                     read backward)
671C-  LDX #$60      Re-load X with disk slot
                     times $10
671E-  DEC $42       Decrement sector until it is
                     negative number
6720-  BPL $6700     If number is not negative,
                     read another sector
6722-  RTS          Finished reading track.
                     Return to caller
```

Softkey for...

Bill Budge's Space Album

California Pacific

Although this California Pacific release is 11 years old and the games are quite simple, breaking through protection is always a bit of a challenge. For those of you who may still have this antique disk, this article will explain how to convert it to normal DOS. Being from 1980, the disk is in a DOS 3.2 format but will also boot on the 3.3 system. In most cases, this would involve a simple swap-DOS copy and installing a 3.3 RWTS. However, the RWTS on this disk could actually be called an RTS (if it hadn't already been used) because it is a scrunched version of the RWTS with no write capability. This makes things a little more difficult because you can't just easily replace their DOS with a normal size RWTS. Fortunately, their RWTS (or RTS) lives at $7600 and the normal RWTS area is not used by the program. What we are going to do is read in the disk data manually using their loader, and write it back out using normal DOS. Even though the loader at $7600 is an obstacle in this process, all of the disk data can still be read in at one time because only Tracks $0-$B are used and there are only 13 ($C) sectors per track.

Begin by booting a normal DOS 3.3 disk and freshly initializing the target disk for the copy.

INIT HELLO

Now, without turning off the computer, insert the Space Album disk and boot it with a PR#6 command. When the title screen appears, press RESET several times until the prompt (}) appears. Then, enter the monitor.

CALL -151

The first thing we'll do is call their loader to read 60 sectors into $1000 through $7000. Their IOB is at $76E8 and to be tricky, they 'flip-flopped' the track and sector locations. Where $76EC is normally track, $76EC is sector and $76ED is track. Also, location $F6 specifies the number of pages to be read or written.

76EB:00 00 01
76F1:10
F6:60

Modify the loader so it ignores any errors it may encounter.

7D62:EA EA EA EA EA

We're ready to do the first read pass. The multi-sector read/write routine is at $7D59.

7D59G

There are only $2F sectors remaining to be read from the disk and they can be read into $8000 through $AEFF. The IOB already points to the next track and sector to be read.

76F1:80
F6:2F
7D59G

At this point, the entire disk contents are in memory (yes, they are small programs) and the original disk is no longer needed. Insert the initialized copy disk.

If you have followed everything correctly, the normal RWTS should still be in memory at $B800. We will be using it to write the data to the normal disk but using the same IOB at $76E8. Modify the RWTS to read the interchanged track and sector locations in the IOB.

BD91:05
BE27:04

The sectors on the 3.2 disk are read in ascending order for faster loading speed and there is no sector skew. However, when reading this way in DOS 3.3, the loading is quite slow. That's what sector skewing is for...to maximize speed. (see Beneath Apple DOS) Instead of going through the trouble of changing the loader to read the sectors in reverse order, a Pascal skew will solve the problem. Changing the skew is not absolutely necessary but the disk will load in at least five times faster.

BFB8:00 02 04 06 08 0A 0C 0E 01 03 05 07 09 0B 0D 0F

Continuing on, change their loader to go to the RWTS entry point at $BD00. Also, tell the IOB to write.

7D5F:BD
76F4:02

Now enter the IOB information to write the data the same way it was read in.

76EB:00 00 01
76F1:10
F6:60
7D59G *(write pass 1)*
76F1:80
F6:2F
7D59G *(write pass 2)*

Even though we won't be using their DOS to read from the copy, there is quite a bit of program code in the loader area. Plus, their IOB will still be used with the normal RWTS so the loader must be written to disk. On the original, it is on track 0 but our track 0 contains the normal RWTS. The next available track is $0C. Before writing it, several modifications must be made so it works with the normal RWTS.

765B:B5 B7
763E:2C
7CE4:49 A1
7CEB:49 A1
7CF2:20 00 79
7900:A9 A1 85 48 4C BC FE

The IOB already points to track $0C sector $00 but it still must be set to write $8 pages from $7600.

76F1:76
F6:08
7D59G

The copying is complete but a few sector edits must be performed in order for the disk to boot. Reboot DOS and run your sector editor. The first five edits are to track 0, sector 1 (boot stage 1) which will cause it to read the Space Album loader from track $0C into $7600. The final sector edit to track 0 sector 9 installs the Pascal sector skew into the RWTS. Remember, the data must be read the same way it was written.

Trk	Sct	Byte	From	To
$00	$01	$00		20 93 B7 A9 05 8D
				91 BD A9 04 8D 27
				BE 4C 00 76
$00	$01	$9F		C8
$00	$01	$AC		EE
$00	$01	$E1		08
$00	$01	$EB		00 0C 00
$00	$01	$F1		76 00 00 01
$00	$09	$B9		02 04 06 08 0A 0C 0E
				01 03 05 07 09 0B 0D

Always remember write the sector back to the disk after editing. ...and Bill Budge's Space Album is deprotected. That's all for now.

Softkey for...

Mabel's Mansion

Datamost

The first thing I do when attempting to copy a protected disk is to run the Locksmith Fast Disk Backup over the disk to see if there are any normal tracks. To my surprise, all tracks except 1 and 20 are normal. What this usually means is dealing with a nibble count routine or some other sort of secondary protection. The first thing I did was copy the disk and boot it. At first it appears to work but when you attempt to play the game, it reboots. Obviously there is a disk check somewhere. The problem is finding it in the program code.

The best way to locate a disk access routine is to use a disk search utility and look for accesses to the disk drive read address ($C08C). The only ones I found were in the loader and boot routine which were loading with a normal format. I was hoping to find something a little more suspicious looking but to no avail. If there was an actual nibble count routine, it was most likely encoded somewhere on the disk. In an attempt to find the routine in memory or a decoding routine, I began to look at the boot code. On track 0, sector 1 I found a JMP $7D00 so I patched in a JMP $FF59 and booted the disk. Then looking at $7D00, I immediately found something suspicious. A small routine at $7D06 decoded a large portion of page $7D. I called this routine so I could then look at the memory in the decoded form. At $7D5F there was yet another decode routine for pages $7E-$A6.

The encoding was done by Exclusive-ORring each byte with the byte before it. After calling this routine I was able to search through all of the decoded memory and right at the end in page $A6 I found the disk access I was looking for. If it doesn't find what it's looking for, it goes to $A69A which clears memory and reboots. I originally patched an RTS at the beginning of the disk routine to skip it but found out that it didn't work. My next thought was to put an RTS at the beginning of the reboot routine so it would return from the routine instead of reboot and this worked. When dealing with encoded memory, you want to make as few byte changes as possible because for every byte, you have to find its encoded equivalent. I wanted to change the $A9 at $A69A to a $60. The encoded form of $A9 was $F8 which is an Exclusive-OR of $51. By Exclusive-ORring $60 with $51 we come up with $31. The next step is finding where page $A6 is stored on the disk. Using the disk search utility, I found it on Track $1A, Sector $6.

The Deprotection Process

Begin by copying the entire disk with Locksmith Fast Disk Backup or some other copier that bypasses errors. Then get out a sector editor and read Track $1A, Sector $6. At byte $9A change from $F8 to $31 and write the sector back to the disk. The final process is that simple. Mabel's Mansion is now easily copyable.

Trk	Sct	Byte	From	To
$1A	$06	$9A	F8	31

Softkey for...

Mr Robot and His Robot Factory

Datamost

Requirements:

Locksmith Fast Disk Backup or any normal DOS copier that will bypass the read errors on Track 1

Sector Editor

Mr. Robot contains a rather common type of protection. The entire disk is normal except for Track 1 which even a bit copier has a hard time copying. This usually means that somewhere during the loading process, the protected track is checked. Once you find the routine that checks the disk, disabling it is not too much of a problem.

Of course, on Mr. Robot, this routine is 'hidden' on the disk in an encoded form. It can be found on Track 9, Sector 6 and it loads into $1900. At the beginning of the sector is a small routine that decodes the rest of the page so it can be read. After decoding it, I was able to modify the routine to bypass the disk check.

There is also a checksum routine on Track 0, Sector 1 that causes a reboot if the disk check routine is altered.

Step-by-step

1. Use any normal DOS copier that will bypass the read errors on Track 1 and make a copy of Mr. Robot.
2. Run a sector editor and make the following changes to the copy:

Trk	Sct	Byte	From	To
$00	$01	$A5	4C 20 08	EA EA EA
$09	$06	$1A	04	21
$09	$06	$41	A5 D3	C9 A0

And Mr. Robot is no longer copy protected!

Softkey for...

Flip Out

Sirius

Yes, another Sirius game, another boot code trace. Unlike some Sirius games, Flip Out does not access the disk after loading and there is nothing tricky about the boot code (is this really a Sirius game??) Let's start by entering the monitor and moving boot 0 down to RAM so it can be modified to return to the monitor after reading in boot stage 1 at $800.

CALL-151
9600<C600.C6FFM
96FA:98 N 9801:AD E8 C0 4C 59 FF

Boot 0 is now exiting to $9801 where we will be moving the next stage of boot. Execute $9600 to load it in and move it to $9800.

9600G
9800<800.8FFM

At $982F there is an LDA #$04 followed by a STA $F8 and STA $FA. The $04 is to specify the page to load into as well as how many pages to load in. We want to change only the value in $F8 to $64 so it reads into $6400 instead of the text page. Some of the code toward the beginning of the boot stage is not vital so we can fit an LDA and a STA there. Also, NOP over the other STA $F8.

9806:A9 64 85 F8
9831:EA EA

The JMP to the next boot stage (JMP $429) is at $988A. Change this to JMP $FF59 and call up $9600 to read in the next stage.

988B:59 FF
9600G

Boot stage 2 is now in memory from $6400 to $67FF and if you list all the way to the end, you will find a JMP $7800 at $67E2. Because there are only two bytes to change, restore the original boot 1 so it loads boot 2 in its proper place and before jumping to boot 2, have boot 1 change the two bytes.

9800<800.8FFM
988A:A9 59 8D E3 07 A9 FF 8D E4 07 4C 29 04

At this point, the game is ready to be loaded in.

9600G

The game uses memory from $C00 to $8FFF. Nothing will be overwritten by rebooting DOS so insert a slave disk with no HELLO program and reboot DOS. Then enter the monitor.

C600G
CALL-151

Write a short routine to turn on the hires screen and wait for a key press, patch DOS for a long file, and save the game.

BE0:AD 50 C0 AD 57 C0 AD 52 C0 AD 10 C0
:AD 00 C0 10 FB AD 10 C0 4C 00 78
A964:FF
BSAVE FLIP OUT, A$BE0, L$8420

Softkey for...

Borg

Sirius

Yes, it's time to crack yet another Sirius product and at the same time, wipe another disk off the Computist Most Wanted List! This is not an easy crack if you don't have a decent knowledge of disk formatting and encoding. Borg is encoded in a straight 4+4 and can be read into memory without too much trouble. We will be making a few alterations to the RWTS so it will read the 4+4 format but still write out in the normal 6+2. Then we will use Super IOB with a special controller to copy the Borg disk with this DOS. Before doing that though, freshly INITialize a blank disk. Make sure you format it with normal Apple DOS.

INIT HELLO

Custom Tailoring the RWTS

Start by entering the monitor and moving an image of the RWTS down to $1900 which is where Super IOB can move it from.

CALL-151
1900<B800.BFFFM

We could use a swap controller for this but it is not necessary in this case because a 4+4 read routine takes very little space and we can simply modify one of the read routines. DOS uses the read address routine to locate the correct sector when writing but the read data routine isn't needed for writing so this is where we'll put the 4+4 reader. Do the following modifications (type carefully).

19DC:AD 13 03 85 D1
19E7:DD D0 F7
19F1:AD
19FC:DA
19FF:A0 00 84 D0 A9 0C 85 D2
1A07<B971.B97FM
1A16:91 D0 C8 D0 EC E6 D1 C6
:D2 D0 E6 BD 88 C0 68 68
:28 18 60

The above modifications will cause the read data routine to read an entire track from Borg. The tracks are not in a sector format and there are only $C00 bytes per track as opposed to the normal $1000. We want the RWTS to write the sectors with a Pascal skew so they can be read in numerical order quickly. Patch in the skew.

20B8:00 02 04 06 08 0A 0C 0E 01 03 05 07 09 0B 0D 0F

The Borg code is written on half-tracks so we want to be able to read half-tracks but write whole tracks. Make a few changes so the RWTS has that capability.

1DE0:18 69 01 4C A0 B9
1F8C:E0 BC

The RWTS is complete so save it to disk for safe keeping.

BSAVE BORG RWTS, A$1900, L$800

It's time to copy the disk so get out Super IOB and install the Borg controller. Make sure the Borg RWTS is in memory before running Super IOB. When ready, copy the disk.

CONTROLLER

```
1000 REM "BORG CONTROLLER
1010 TK = 1:ST = 0:LT =
  33:CD = WR
1015 GOSUB 360
1020 T1 = TK: GOSUB 490:
  POKE 48581,220: POKE
  48582,184: POKE 48354,1
1030 GOSUB 430: GOSUB 100
1040 POKE BU, PEEK (BU) +
  15: IF PEEK (BU) = > MB
  THEN 1060
1050 TK = TK + 1: IF TK = 12
  THEN 1050
1055 IF TK < LT THEN 1030
1060 GOSUB 490:TK = T1:ST =
  0: POKE 48581,68: POKE
  48582,185: POKE 48354,0
1070 GOSUB 430: GOSUB 100:ST
  = ST + 1: IF ST < DOS THEN
  1070
1080 ST = 0:TK = TK + 1: IF
  TK = 12 THEN 1080
1085 IF BF = 0 AND TK < LT
  THEN 1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT
  "COPY◊COMPLETE!" : END
```

Checksums

```
1000-$356B 1040-$A908 1080-$BD75
1010-$B094 1050-$D2ED 1085-$B118
1015-$34EB 1055-$C96B 1090-$75B3
1020-$915A 1060-$F1C7 1100-$7545
1030-$4462 1070-$F9C2
```

When the copy is complete, run your sector editor. The first edits will be to Track 0, Sector 1. This is the stage 2 boot loader. Enter a routine to load the game program into memory. Start entering at byte $00.

A0 00 84 00 B9 35 B7 D0
07 A9 0A 85 31 4C B0 0E
8D F1 BC A9 00 8D ED BC
A9 BC A0 E8 20 00 BD EE
F1 BC EE ED BC AD ED BC
C9 0C D0 EC EE EC BC A4
00 C8 4C 02 B7 08 14 20
2C 34 60 6C 74 80 8C 94
00

Write this sector back to the disk. Then, read Track 0, Sector 6. Part of the game code will load over page $B7 where the normal IOB is located so the IOB has to be written elsewhere. There is enough free space at $BCE8-$BCFF for the IOB. Enter the IOB at byte $E8.

01 60 01 00 01 00 FB BC
00 00 00 00 01 00 FE 60
01 00 00 00 01 EF D8 00

Again, rewrite the sector. The next edit is on Track 0, Sector 9. We wrote to the disk with a Pascal sector skew so it must be read with the same skew. Enter at byte $B8.

00 02 04 06 08 0A 0C 0E
01 03 05 07 09 0B 0D 0F

Write the sector. During the Borg boot code, it stores $4C D0 8C at location $20 before jumping to the game. There is only one JMP $0020 and the locations do not change. You can find this JMP on Track 4, Sector $B. At byte $AF enter D0 8C so it jumps directly to $8CD0. (Write it back)

Read Track $A, Sector 9. Change byte $58 from $16 to $17. This is because the copy of the disk is no longer on half-tracks.

The final edits are on Track $A, Sector $A. You'll find their track seek routine at byte 0. Since we are using the RWTS which contains its own seek routine, their routine is not needed. Instead, store the desired track in the IOB track location. Enter at byte $00.

38 E9 01 4A 8D EC BC 60

Finally, at byte $90, is their track read routine. We will modify it to read the same desired track but using the RWTS. Enter the read routine at byte $90.

8C F1 BC A9 00 8D ED BC
A9 BC A0 E8 20 00 BD EE
F1 BC EE ED BC AD ED BC
C9 0C D0 EC 60

Write this sector back to the disk... and there you have it! Another protection scheme foiled and another copyable disk.

Softkey for...

Type Attack

Sirius

What makes Type Attack more difficult to crack than most Sirius disks is that it goes back to the disk for each level and allows you write your own lessons to the disk. With a little time and ambition, the game and its levels can be copied down to a normal disk. Begin by freshly INITializing a slave disk and label it 'TYPE ATTACK COPY'. Then, enter the monitor.

CALL-151

Insert the Type Attack disk. We need to get their DOS into memory from Track 0 by boot code tracing.

9600<C600.C6FFM
96FA:98 N 9801:4C 59 FF
9600G
9800<800.8FFM
9805:98
9894:59 FF
9600G
C0E8

The Type Attack DOS runs from $800 to $DFF. Move it to $1800 so it doesn't get overwritten, insert your copy disk, and reboot.

1800<800.DFFM
C600G

A few changes must be made to this DOS so it uses the RWTS instead of its own read/write routines.

CALL-151
19A8:60
19C3:60
1A0D:A5
1A45:EA EA EA EA EA EA
1A51:EA
1A5B:4C 80 B7
1B01:EE F4 B7 4C 15 0B
1B1E:20 80 B7 CE F4 B7 60

The Type Attack DOS is going to be written to the disk, not as a file, but directly to Track 0. We'll use the RWTS and the IOB at $B7E8.

B7E1:06
B7EC:00 0F FB B7 00 1D 00 00 02
B793G

The RWTS on the disk must be altered to read in the Type Attack loader at boot and also to work with the loader. Run your sector editor and read Track $0, Sector $1.

Trk	Sct	Byte	From	To
$00	$01	$00	??	20 93 B7 4C CD 09
		$80	??	18 69 0B 8D F1 B7 8C EC B7 A9 0B 8D ED B7 A9 0C 8D E1 B7
		$E1	??	06
		$EB	??	00 00 0F FB B7 00 0D 00 00 01

The Type Attack loader is ready to work with normal DOS but there is just one problem... there is nothing on the disk to load. That's where Super IOB comes in. Like I did with BORG, I patched a 4+4 loader into the DOS read routines so it will read Type Attack instead of a normal format. First, enter the monitor and move a copy of the RWTS to $1900 where Super IOB will use it.

CALL-151
1900<B800.BFFFM

The following modifications will cause the 'read data' routine to read an entire track from Type Attack.

19DC:AD 13 03 85 D1
19E7:AD D0 F7
19F1:DA

19FC:DD
19FF:A0 00 84 D0 A9 0C 85 D2
1A07<B971.B97FM
1A16:91 D0 C8 D0 EC E6 D1 C6
:D2 D0 E6 BD 88 C0 68 68
:28 18 60

In case of an error, you may want to save this to a disk. DON'T use the Type Attack copy disk.

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

Next, load Super IOB and install the Type Attack controller. Then run the program (make sure TA.RWTS is intact) and copy the disk. It will only copy tracks $01-$0C.

CONTROLLER

```
1000 REM "TYPE ATTACK
1010 TK = 1:ST = 0:LT =
  13:CD = WR
1015 GOSUB 360
1020 T1 = TK: GOSUB 490:
  POKE 48581,220: POKE
  48582,184
1030 GOSUB 430: GOSUB 100
1040 POKE BU, PEEK (BU) +
  15: IF PEEK (BU) = > MB
  THEN 1060
1050 TK = TK + 1: IF TK < LT
  THEN 1030
1060 GOSUB 490:TK = T1:ST =
  0: POKE 48581,68: POKE
  48582,185
1070 GOSUB 430: GOSUB 100:ST
  = ST + 1: IF ST < DOS THEN
  1070
1080 ST = 0:TK = TK + 1
1085 IF BF = 0 AND TK < LT
  THEN 1070
1090 IF TK < LT THEN 1020
1100 HOME : PRINT "DONE◊WITH
  ◊COPY" : END
```

Checksums

```
1000-$356B  1040-$CAEA  1085-$B035
1010-$E114  1050-$230F  1090-$EEDE
1015-$35E9  1060-$5DC6  1100-$D998
1020-$534D  1070-$55C3
1030-$6F46  1080-$E6EE
```

When the copy is complete, Type Attack should be ready to boot and play. Have fun!!!

Softkey for...

Minotaur

Sirius

Minotaur has almost the same identical protection and format as Bandits, but fortunately, the lengthy Bandits Softkey procedure need not be used to deprotect this game. Once the game program is in memory, nothing more is done with the disk except for a disk check which can be disabled easily. Most single load programs can be downloaded into a single BRUNable file but Minotaur uses memory from $800 to $BFFF with only Hires page 1 open. The game could be saved as two files where the first file reads in the second but that is almost more trouble than it's worth. It's much easier to just write the game directly to the disk using the RWTS and read it into memory when the disk boots.

1. To begin, initialize a disk for the deprotected copy.

INIT HELLO
DELETE HELLO

2. The Minotaur disk must be boot code traced twice to capture all of the memory. Enter the monitor and type the following.

CALL-151
9600<C600.C6FFM
96FA:98 N 9801:AD E8 C0 4C 59 FF
9600G
9800<800.8FFM
986C:A9 2C 8D 64 04 A9 59 8D 34 07 A9 FF
8D 35 07 4C 1F 04
9600G

3. When you hear the 'beep', the game will be completely in memory. Hi-res page 1 ($2000-3FFF) is blank. We need to save the zero page so move it into $2000.

2000<0.FFM

4. We still have $1F pages of memory left in the Hi-res page so save $A100-BFFF.

2100<A100.BFFFM

5. Before rebooting, move page $8 to safety, then insert your Minotaur backup disk, reboot, enter the monitor, and move page $8 back.

8000<800.8FFM
C600G
CALL-151
800<8000.80FFM

6. We're going to write $800-$7FFF to tracks 3-A. Enter the appropriate IOB information. The routine at $B793 uses the information in the IOB to read or write a specified number of sectors. $B7E1 is the number of pages to read or write. Execute this routine at $B793.

B7EC:0A 0F FB B7 00 7F 00 00 02
B7E1:78
B793G

7. Once again, the disk must be boot code traced so memory from $8000 to A0FF can be saved. You should still be in the monitor so repeat steps 1-6 from above.

8. The memory we're saving must be moved so it is not overwritten by DOS when we reboot. Insert the Minotaur copy disk.

2000<8000.A0FFM
C600G

9. At $8294 (which is now at $2294) is a JSR $0400. This is where it goes to the disk for verification of the original. Overwrite this command with 3 NOPs.

CALL-151
2294:EA EA EA

10. This time we will be writing this memory from track $B, sector 0 to Track $D, sector 0.

B7EC:0D 00 FB B7 00 40 00 00 02
B7E1:21
B793G

11. The entire game is now written to the disk. All that is left is to change boot stage 2 to load in the game. Run your sector editor and read Track 0, Sector 1. Beginning at byte $00, enter the following routine:

20 93 B7 A2 00 BD 00 B7
9D 00 03 E8 D0 F7 4C 11
03 BD 00 21 9D 00 A1 E8
D0 F7 EE 13 03 EE 16 03
AD 16 03 C9 C0 D0 EA BD
00 20 9D 00 00 E8 D0 F7
4C F8 0F

The above routine loads into $B700, moves itself to $300 and jumps to $311. From there it moves $2000 to the zero page and $2100-3FFF to $A100-BFFF and finally jumps to the start of the program at $FF8.

12. At byte $EB, enter the correct IOB information to load in the game.

00 0D 00 FB B7 00 A0 00 00 01

13. And finally, enter a $99 at byte $E1 to specify the number of pages to read in. Re-write this sector and you're all finished!!

Alan Chaney — MD

The softkey that I am about to give, were given before in issue #61 by Mr. Jim Bancroft and issue #68 by Mr. Joseph P. Karwoski. Their keys involved turning off the error checking in DOS (B942:38 to 18). Now don't hold me to this, but I read in one of my Computist issues that turning off the error checking in DOS should only be done as a last resort. Reason being that no errors that might occur will get caught by DOS after this patch is made. Well, this is why I rekeyed this program.

Softkey for...

Early Skills (2 diskettes)
Clock
Money
What's First? What's Next?

Hartley Courseware, Inc.

Requirements:
COPYA or any program that can ignore epilog errors
Sector Editor

Copy disk or diskettes depending on the program, with the method you choose to ignore epilog errors. Scan disk with a sector editor for C9 DA and change to C9 DE , Both changes should be in sector 03 on Track 00. The two changes represent the corrections to the address and data epilog bytes and the end of the protection. Both of the Early Skills diskettes must be deprotected due to the fact that both disk have their own DOS to Boot up or start up with. These programs are for kids in grades K-3.

Softkey for...

Kinder Koncepts

Queue

Requirements:
COPYA or any program that can ignore epilog errors
Sector Editor

Copy disks with the method you choose to ignore epilog errors. Scan disk with a sector editor for C9 DF and change to C9 DE. Changes must be made on all 3 diskettes. The change is to the data epilog byte only. I hope someone understands this article.

Softkey for...

Alge - Blaster Plus

Davidson & Associates

Requirements:
2 Blank 5.25" disks
Copy program that can ignore errors
Sector Editor

The bit copy of Alge-Blaster also works on Alge-Blaster Plus (See Mr. Gerald E. Myers article in issue # 64). I think I understand the meaning of the saying "a little knowledge can be dangerous". I would be glad when someone mails me my little knowledge, so I can be dangerous. Well, so much for my wishes. Lets get cracking.

Because the program quits to the ProDOS screen shortly after booting the copy, I searched for the ProDOS quit code 20 00 BF 65 and looked for any branch to the code. I changed the branch from D9 to 00 so it would continue on its way to the game.

Step-by-step

1. Copy program side with program to ignore errors.
2. Scan disk for 20 C8 0A F0 D9 AD 98 BF.
3. Change D9 to 00 and write sector back to copy.
4. Copy the other 3 sides of program with any copier.

You may think that I get to the point a little to fast? Well all I see is the point. Maybe after I get my mail, I will beat my gums a little longer.

Softkey for...

Mastertype's Writer

Scarborough System

Requirements:
1 Blank disk
Copy program (23 Tracks)
Sector Editor

The Bitcopy of Mastertype's Writing Wizard by Mr. Kearney J. Gravis, issue #67 page 29, works also on this program. But the program can't be copied by any copy program. Mr. Robert Phillis softkey for this program wore me out, because I couldn't find the place at which to install the patch (I guess I am not a rocket scientist after all). But his article in issue #70 page 14 gave me enough info to softkey the program another way. Thank you sir.

Step-by-step

1. Copy all 23 tracks of the original.
2. Replace the ProDOS on the disk (optional).
3. Scan for 20 AF 10 D0 01 60 00 AD
4. Change 00 to 60 and write sector back to copy.

Note: I was going to change the branch 01 to 00 that way you can return also. But after looking further back (50 bytes), I found at least 5 branches to address AA4A, which is the 00 byte. Thanks again everyone for allowing me to work in this organization.

Softkey for...

Spell It

Davidson & Associates

Requirements:
1 Blank 3.5 disk
Sector Editor
Any copy program

Spell it shows an error on block 308 when you copy with Copy II Plus fast copy. But after reading Mr. Jim Ross's article in issue #74 page 11, Somehow I knew I could defeat this error and get the program up and running. The edit Mr. Ross found for his program were on block 1DD on my program. His softkey involved noping out 12 bytes to get to the right code, I just changed the code that the program branches to if not correct, to the code that the program is looking for (A9 00 to A9 FF). Without Mr. Ross's article this could not have happened. My style is scan for bytes, so here we go.

Step-by-step

1. Copy disk with any fast copier.
2. Scan the copy for D0 04 A9 00 85 FF 68.
3. Change 00 to FF and write sector back to copy.

That's it!

Aaron Culliney — HI

WindwalkerGS Editor

Windwalker is a neat game, but like most games it can be frustrating to play

at times. Most of my frustration was over unclear documentation on how to cast spells. Specifically, the game asks you to "Speak the mantra" when you attempt to invoke magic. I found out much later that for mantras to work, they must be typed using all lower-case letters. This was in contrast to the part of the game listing all the mantras with upper-case first letters. I went through almost the whole game without using magic because the mantras that I typed in using upper-case first letters were not accepted. So, on the verge of calling ORIGIN and inquiring about the problem, I whipped out a copy of the ProDOS block editor from Computist #55 and did some snooping around. I discovered the mystery of the mantras, and was finally able to cast spells. Unfortunately in-game magic alone would not move my character away from the margin of utter death; going through a whole game without use of spells had depleted the KARMA. From these beginnings the idea and reality of a simple character editor for Windwalker GS developed.

The reading and writing machine code routine for the Windwalker GS character editor is taken directly from the ProDOS Block Editor by Bob Bergstrom and modified by Rene Gaudet, Computist #55. In fact the WIND.OBJ program is just a modified OBJ.PROEDIT program from the same article in Computist #55. WIND.OBJ is modified to read and write block $038 only, where the character data resides.

This is a simple editor but it allows editing of character statistics, (BODY/SPIRIT/HONOR/KARMA) and items, (health elixirs, magic talismans,the antidote, etc). The program loads the character data onto memory page $9000.

Entering the code:

1. Enter the BASIC program and save it.
SAVE WINDWALKGS.EDIT

2. Enter the machine code and save it.
BSAVE WIND.OBJ, A$0300, L$A5

WINDWALKGS.EDIT

```
0 BUF = 36864: GOTO 780
10 REM WINDWALKER GS
  CHARACTER EDITOR
20 REM BY AARON CULLINEY
30 REM WIND.OBJ PROGRAM
  MODIFIED FROM
40 REM OBJ.PROEDIT PROGRAM
  COMPUTIST #55
50 REM THANKS TO RENE
  GAUDET/BOB BERGSTROM
60 REM NO FRILLS, BUT IT
  WORKS...
70 REM ***
80 REM IF BUFFER EMPTY, READ
  IN BLK $38
90 IF PEEK (36864) = 0 THEN
  POKE 783,128: CALL 774
100 HOME : REM DISPLAY NAME/
  EDIT STATS
110 FOR X = 0 TO 15:FV =
  PEEK (BUF + X): VTAB 1:
  HTAB X + 1: PRINT CHR$
  (FV): IF PEEK (BUF + X) =
  0 THEN GOTO 130
120 NEXT
130 FOR X = 1 TO 4: READ A$:
  READ B: PRINT A$, PEEK
  (BUF + B): NEXT
140 PRINT :ED = 0: GOSUB 320
150 HOME : REM MULTIPLE
  ITEMS OF A KIND
160 PRINT "ITEMS(LIST1):" ,
  "VALUE:"
170 FOR X = 1 TO 20: READ
  C$: READ D: PRINT C$, PEEK
  (BUF + D): NEXT
180 ED = 1: GOSUB 320
190 HOME : REM CERTAIN ITEMS
  LIST1
200 PRINT "ITEMS(LIST2):" ,
  "CONDITION:"
210 FOR X = 1 TO 20: READ
  E$: READ F: IF PEEK (BUF +
  F) > 0 THEN PRINT E$,
  "HAVE◊IT"
220 IF PEEK (BUF + F) = 0
  THEN PRINT E$, "DONT◊HAVE
  ◊IT"
230 NEXT
240 ED = 2: GOSUB 320
250 HOME : REM CERTAIN ITEMS
  LIST2
260 PRINT "ITEMS(LIST3):" ,
  "CONDITION:"
270 FOR X = 1 TO 17: READ
  G$: READ H: IF PEEK (BUF +
  H) > 0 THEN PRINT G$,
  "HAVE◊IT"
280 IF PEEK (BUF + H) = 0
  THEN PRINT G$, "DONT◊HAVE
  ◊IT"
290 NEXT
300 ED = 3: GOSUB 320
310 RESTORE : GOTO 780
320 PRINT "U/D◊ARROW-MOVE|
  SPACE-EDIT|RETURN-NEXT"
  :PTR$ = ">" :YY = 2: REM
  GET INPUT
330 VTAB YY: HTAB 16: PRINT
  PTR$
340 VTAB 22: GET IP$
350 IF IP$ = CHR$ (10) THEN
  VTAB YY: HTAB 16: PRINT
  "◊" :YY = YY + 1: GOTO 410
360 IF IP$ = CHR$ (11) THEN
  VTAB YY: HTAB 16: PRINT
  "◊" :YY = YY - 1: GOTO 410
370 IF IP$ = CHR$ (13) THEN
  RETURN
380 IF IP$ = CHR$ (27) THEN
  RESTORE : GOTO 780
390 IF IP$ = "◊" THEN GOTO
  460
400 GOTO 330
410 IF YY < 2 THEN YY = 2:
  GOTO 330: REM DETERMINE
  POINTER OUT-OF-BOUND
420 IF ED = 0 AND YY > 5
  THEN YY = 5: GOTO 330
430 IF ED = 3 AND YY > 18
  THEN YY = 18: GOTO 330
440 IF YY > 21 THEN YY = 21:
  GOTO 330
450 GOTO 330
460 IF ED = 0 THEN GOTO 500:
  REM DIFFERENT EDITS FOR
  DIFF CHARACTER DATA
470 IF ED = 1 THEN GOTO 530
480 IF ED = 2 THEN GOTO 670
490 IF ED = 3 THEN GOTO 750
500 VTAB YY: HTAB 16: INPUT
  NN: REM POKE CHARACTER
  STATS
510 IF NN < 0 OR NN > 10
  THEN GOTO 500
520 POKE (BUF + 40 + (YY -
  1)),NN: GOTO 330
530 VTAB YY: HTAB 16: INPUT
  NN: REM POKE MULTIPLE
  ITEMS OF A KIND
540 IF NN < 0 OR NN > 99
  THEN GOTO 530
550 IF YY = 2 THEN POKE (BUF
  + 51),NN
560 IF YY = 3 THEN POKE (BUF
  + 52),NN
570 IF YY = 4 THEN POKE (BUF
  + 54),NN
580 IF YY = 5 THEN POKE (BUF
  + 55),NN
590 IF YY > 5 AND YY < 13
  THEN POKE (BUF + 60 + (YY
  - 6)),NN
600 IF YY > 12 AND YY < 17
  THEN POKE (BUF + (79 - (17
  - YY) * 2)),NN
610 IF YY = 17 THEN POKE
  (BUF + 80),NN
620 IF YY = 18 THEN POKE
  (BUF + 81),NN
630 IF YY = 19 THEN POKE
  (BUF + 88),NN
640 IF YY = 20 THEN POKE
  (BUF + 89),NN
650 IF YY = 21 THEN POKE
  (BUF + 90),NN
660 GOTO 330
670 IF YY = 2 THEN NP = (BUF
  + 53): REM POKE CERTAIN
  ITEMS1
680 IF YY > 2 AND YY < 7
  THEN NP = (BUF + 50 + (YY
  + 3))
690 IF YY > 6 AND YY < 11
  THEN NP = (BUF + 60 + YY)
700 IF YY > 10 AND YY < 15
  THEN NP = (BUF + (84 - (17
  - YY) * 2))
710 IF YY = 15 THEN NP =
  (BUF + 79)
720 IF YY > 15 AND YY < 22
  THEN NP = (BUF + 70 + (YY
  - 4))
730 IF PEEK (NP) = 0 THEN
  POKE NP,1: VTAB YY: HTAB
  17: PRINT "HAVE◊IT◊◊◊◊" :
  GOTO 330
740 IF PEEK (NP) > 0 THEN
  POKE NP,0: VTAB YY: HTAB
  17: PRINT "DONT◊HAVE◊IT" :
  GOTO 330
750 NP = (BUF + 90 + (YY -
  1)): REM POKE CERTAIN
  ITEMS2
760 IF PEEK (NP) = 0 THEN
  POKE (NP),1: VTAB YY: HTAB
  17: PRINT "HAVE◊IT◊◊◊◊" :
  GOTO 330
770 IF PEEK (NP) > 0 THEN
  POKE (NP),0: VTAB YY: HTAB
  17: PRINT "DONT◊HAVE◊IT" :
  GOTO 330
780 TEXT : HOME : PRINT
  "WINDWALKER-GS◊CHARACTER
  ◊EDITOR"
790 IF PEEK (774) < > 32
  THEN PRINT : PRINT CHR$
  (4) "BLOADWIND.OBJ,A$0300"
800 PRINT : PRINT "1)◊EDIT◊
  CHARACTER" : PRINT "2)◊SA
  VE◊CHANGES" : PRINT
  "?)◊QUIT"
810 PRINT : PRINT "(ESC)◊WI
  LL◊SHOW◊THIS◊MENU" : PRINT
  "WINDWALKER◊B/C◊MUST◊BE◊IN
  ◊SLOT5◊DRIVE1"
820 GET Q
830 IF Q = 1 THEN GOTO 90
840 IF Q = 2 THEN POKE
  783,129: CALL 774: GOTO
  780
850 END
900 DATA BODY,41,SPIRIT,42,
  HONOR,43,KARMA,44
910 DATA FOOD,51,MONEY,52,
  COMMON◊INCENSE,54,BLESSED
  ◊INCENSE,55,HEALTH◊ELIXIR
  ,60
920 DATA SPIRIT◊ELIXIR,61,
  XAL◊XE◊PASTE,62,OBECALP◊FO
  ◊YU,63,RAT◊BLADDERS,64
930 DATA STONEHEAD◊SYRUP,65,
  EYES◊OF◊FIRE,66
940 DATA HERON◊FEATHERS,71,
  BLIND◊MANS◊SHOE,73, BEETLE
  ◊JAW,75,DRAGON◊SCALE,77
950 DATA GREEN◊TURTLE,80,
  STRIPED◊TURTLE,81,PARCH
  MENTS,88,QUILL◊PEN,89,INK
  ◊HORN,90
960 DATA STRAW◊MAT,53,PROVIN
  CE◊MAP,56,EXPLORERS◊MAP,57
  ,SEXTANT,58,ROBE/STAFF,59
970 DATA PRISON◊KEY,67,WARL
  ORDS◊KEY,68,FENG◊SHUS◊KEY
  ,69,BRASS◊KEY,70
980 DATA LEVITATION,72,INVIS
  IBILITY,74,WALK◊ON◊WATER,
  76,INVULNERABLE,78
990 DATA IDOL◊OF◊STONE,79,
  RHINO◊HORN,82
1000 DATA JASMINE◊FLOWER,83,
  JADE,84,GOLD◊DUST,85,PEACH
  ◊SEED,86,ANTIDOTE,87
1010 DATA SHRINE◊SCROLL,91,
  RELIGION◊SCROL,92,PROPHE
  CY◊SCROL,93,MOEBIUS◊SCROLL
  ,94
1020 DATA NUBIA◊SCROLL,95,
  LANGUAGE◊SCROL,96,ASTRONMY
  ◊SCROL,97,SURVIVAL◊SCROL
  ,98
1030 DATA IDOLATRY◊SCROL,99,
  WARLORD◊SCROLL,100,ALCHE
  MST◊SCROL,101,SHAMAN◊
  SCROLL,102
1040 DATA FENG◊SHU◊SCROL,103
  ,LU◊SHANG◊SCROL,104,DIAHN◊
  JON◊SCRL,105,APTHECRY
  ◊SCROL,106
1050 DATA JAILER◊SCROLL,107
```

Checksums

```
0-$2366     340-$AA14   680-$350A
10-$7CAB    350-$A2D9   690-$DD00
20-$0521    360-$2281   700-$A149
30-$47B9    370-$1044   710-$062B
40-$77B4    380-$6160   720-$1A1C
50-$A667    390-$D7F9   730-$B283
60-$894F    400-$B3A2   740-$30CB
70-$9145    410-$E6BC   750-$D761
80-$2B0E    420-$3ADD   760-$3811
90-$9F6A    430-$0300   770-$7F6B
100-$1E1F   440-$293B   780-$BE5C
110-$E23B   450-$1006   790-$3537
120-$A591   460-$5B47   800-$ECB0
130-$1194   470-$BB47   810-$FBF1
140-$D63E   480-$D8AB   820-$02AC
150-$3A4F   490-$6412   830-$81EE
160-$5DC5   500-$E52A   840-$DBED
170-$030D   510-$7820   850-$4EA3
180-$EE74   520-$7E62   900-$914B
190-$2043   530-$96A4   910-$50C4
200-$323C   540-$0C71   920-$41BA
210-$4271   550-$E456   930-$040D
220-$1243   560-$B6ED   940-$A4A3
230-$C522   570-$723E   950-$FA8C
240-$DEC9   580-$5A74   960-$0948
250-$DDD1   590-$58C2   970-$B1D8
260-$A386   600-$64D4   980-$8B42
270-$F219   610-$1F3C   990-$BEE1
280-$439F   620-$BF33   1000-$8AB2
290-$F364   630-$19DB   1010-$29E9
300-$2079   640-$D9E4   1020-$3F42
310-$3F04   650-$7DD1   1030-$42BD
320-$AA6D   660-$FCE7   1040-$B4DF
330-$9DB4   670-$D449   1050-$A942
```

WIND.OBJ

```
0300:01 00 00 00 2F 00 20 0C $6F39
0308:03 D0 08 60 20 00 BF 80 $DB57
0310:1A 03 60 8D 04 03 20 3A $43E8
0318:FF 60 03 50 00 90 38 00 $7366
0320:AD 03 03 4C 30 03 AD 03 $61A6
0328:03 AE 00 03 E0 02 F0 04 $739A
0330:20 DA FD 60 09 80 20 ED $AE2D
0338:FD 60 A9 01 85 25 20 22 $7695
0340:FC A9 00 85 24 8D 02 03 $C11E
0348:A9 0D 8D 01 03 A9 A0 20 $4D81
0350:ED FD AE 02 03 BD 00 90 $A706
0358:AE 00 03 E0 02 F0 0B 20 $85EA
0360:DA FD A9 A0 20 ED FD 4C $822C
0368:8B 03 09 80 C9 A0 10 0E $DAB1
0370:AE 05 03 E0 00 F0 05 E9 $7E8F
0378:80 4C 7E 03 A9 AE 20 ED $AF02
0380:FD A9 A0 20 ED FD A9 A0 $B57D
0388:20 ED FD EE 02 03 F0 08 $DED0
0390:CE 01 03 D0 BD 4C 48 03 $D20F
0398:20 9C FC E6 25 20 22 FC $213B
03A0:A9 16 85 22 60 1E 00 00 $AFC6
03A8:00 1F 00 00 20 21 00 22 $B978
03B0:23 24 25 26 27 28 00 00 $B9DC
03B8:00 00 00 29 2A 2B 00 2C $446D
03C0:2D 2E 2F 30 31 32 FF FF $8110
03C8:33 34 35 36 37 38 FF 39 $418F
03D0:4C 00 BE 4C 00 BE 8D 08 $1815
03D8:C0 B5 42 8D 09 C0 95 42 $7F78
03E0:CA 10 F3 A9 28 38 8D 08 $3CBE
03E8:C0 60 25 FD 00 00 00 00 $25F1
03F0:59 FA 00 BE 1B 4C 03 BE $6FC1
03F8:4C 00 BE 4C 59 FF EB BF  $4FC
```

IBM IBM IBM IBM IBM IBM

Don Westcott — CO

In COMPUTIST #76 on page 22 Mike from Canada submitted a method for bypassing password protection. Most of the IBM games I have use this protection so I tried this method on all of them but it only worked on one:

IBM Softkey for...

Battle Chess II

Interplay

The copy protection for this game is having to look up a chess move from the manual. The first move listed in my manual is: C2H5 So I used the PC SHELL of PC TOOLS 6.0 to do a Text Search for this string. I found it in Relative sector 48 of the file SETUP.EXE. All the other chess moves follow the first one in the same order that they appear in the manual.

The C2H5 in hex is: 43 32 48 35 The moves are separated by 0D 0A so I changed the moves to 00's. It ended up looking like this: 0D 0A 00 00 00 00 0D 0A 00 00 00 00 0D 0A... through all of the moves. Write them back to the file. Then when asked to enter a move just hit the enter key.

Perhaps the other games have somehow encoded their passwords because I couldn't find any of them. Most of Accolade's games use a code wheel with numbers instead of words. Microprose's M1 TANK PLATOON uses password AND key disk protection.

Unknown

IBM Softkey for...

Carrier Command

?

Well, another doc check. At least they were explicit about it. It can be removed like most by a small change.

For Norton users search the file CARRIER.EXE for the byte pattern C2 00 74 AB and change the 74 AB to 90 90.

DEBUG method. DEBUG is assumed to be in the current path or dir.

REN CARRIER.EXE CARRIER.ZAP
DEBUG CARRIER.ZAP
E FBB9 90 90
W
Q
REN CARRIER.ZAP CARRIER.EXE

IBM Softkey for...

Where in the U.S.A. is Carmen Sandiego?

Broderbund

This file will tell you how to remove the copy protection from CARMEN.EXE in "Where in the U.S.A. is Carmen Sandiego?" by Broderbund.

1. Copy all the files to a new subdirectory.
2. Copy DEBUG.COM to the new subdirectory.
3. Patch CARMEN.EXE using DEBUG.

REN CARMEN.EXE CARMEN.ZAP
DEBUG CARMEN.ZAP
E 3C7C 90 90
E 3C7F EB 05
E 3C99 90 90
E 3C9C EB 05
E 3CA5 04
E 3CC4 90 90
E 3CC7 90 90 90 90 90 EB 07
E 3CD7 04 90 90 90
E 3CEC 90 90
E 3EAA EB 05
W
Q
REN CARMEN.ZAP CARMEN.EXE

You should be able to run CARMEN from hard disk, or any other disk without the master disk in drive A. Now you can become the detective you've always wanted to be.

IBM Softkey for...

Colonel's Bequest

Sierra

This softkey will cause the fingerprint to be Celie's all the time, so when it light's up just hit enter! Use PCtools or other program and edit SCIV.EXE. Go to sector 68, offset 223, and change 75 to EB. That's it!

IBM Softkey for...

Continuum

Data East

To softkey Continuum, you need a hex string search utility program, such as the Norton Utilities. The code that needs to be changed is in the file PROGS.CC1 (filesize and datestamp are 163539 11-29-90 12:00p). There are three hex strings you will need to find and change.

Search for:	75 11 BF AB 24 2E 8B
Replace with:	90 90 BF AB 24 2E 8B
Search for:	75 11 BF D5 24 2E 8B
Replace with:	90 90 BF D5 24 2E 8B
Search for:	75 11 BF AB 24 2E 1B
Replace with:	90 90 BF AB 24 2E 1B

That's it ! Any four symbols entered during the ID sequence will start the game.

IBM Softkey for...

Crime Wave

Access

To remove questions use PCtools or other edit program to edit CW.EXE. Go to sector 7, offset 307, and change CD 21 to 90 90. Then to sector 7, offset 314, and change CD 21 to 90 90. Then to sector 7, offset 416, and change 75 0D to 90 90. That's all there is to it.

IBM Softkey for...

Crimewave v1.1

Access

Search (a copy of) CW.EXE for 75 0D and change it to 90 90. That's all there is to it. Now when it asks you for a password, just hit return.

IBM Softkey for...

Curse of the Azure Bonds

?

Requirements:
Norton Utilities (or similar program)
A copy of the file START.EXE from your Azure Bonds disk A

First load START.EXE into Norton. Then search for the string 80 3E CC. This should take you to file offset 9BA hex. Go back to 9B5 hex this should be 9A (the first machine language code for a far call). Change the values of the bytes from 9B5-9B9 hex to 90's. Save the changes.

Now the program will skip the part where it asks for code letter, you now can put away that annoying code disk until needed for decoding messages in the game.

IBM Softkey for...

Dragon's Lair II

?

Here's a sure fire solution that worked for me. Hopefully you have a TEXT/HEX editor (I used PCTOOLS.)

Search DL2DISK2.DAT (on disk #2) for 75 01 CB 8C D3 and replace the 75 01 with 90 90. The screen will still be there, just enter any 5 digit number and you're on your way

IBM Softkey for...

Dragon's Lair

?

Use Norton utils, PCtools etc and search for the following byte patterns and replace them as shown.

Search for	Replace with
32 04 74 07 B8	32 04 EB 07 B8
7E 00 73 07	7E 00 EB 07
3B C3 74 14	3B C3 EB 14

That's it! Enjoy!

IBM Softkey for...

Dragon's Lair

?

Use PCTOOLS or other program and edit GAME.EXE. Go to Sector 29, offset 3 and change CD 21 to 90 90. Go to sector 29, offset 10 and change CD 21 to 90 90. Go to sector 29, offset 18 and change 74 to EB. Go to sector 29, offset 33 and change 73 to EB. Go to sector 29, offset 45 and change 74 to EB. That's it!

IBM Softkey for...

Earthrise

?

Well it looked like another simple doc check, but these guys are a little sneaky. The game program actually begins in the file SOL.EXE, but it is set up to exit to DOS if you try to run it. You must run EARTHRIS.EXE which then runs SOL.EXE. EARHTRIS.EXE was designed to make you think this is the program to tamper with. It overrides INT 3 and give you a "Mind your own business. It's a wild goose chase anyway" message. There is a decisive jump in EARTHRIS.EXE for the DOS exit routine, but altering the program at this point makes a "Security Violation" message appear upon playing. Also the program uses a JMP to decide your answer, not a JZ or JNZ or anything like that as shown below. It calls a routine which then uses a JMP to exit instead of a RET. But by eliminating the "you are wrong jump" in SOL.EXE this game is at your feet.

For Norton users, search SOL.EXE for the byte pattern E9 28 FD and change these numbers to 90 90 90.

DEBUG users follow the steps below. DEBUG is assumed to be in the current path or dir.

REN SOL.EXE SOL.ZAP *DEBUG cannot save .EXE*
DEBUG SOL.ZAP
E 33AC 90 90 90
W *to save it*
Q *to quit DEBUG*
REN SOL.ZAP SOL.EXE

Okay, you're all set. Just hit return when the doc check appears.

IBM Softkey for...

Escape From Hell

?

Better grab a microscope if you're haven't got a cracked version. This doc check asks about some monsters whose tiny pictures appear in the manual.

Since the portion to be altered is not in the first segment of the file you will have to use Norton, or another good editor. DEBUG won't work, unless someone knows how to find where DEBUG loads additional segments.

Below is a list of offsets of the byte to change in the file ESCAPE.EXE. Go to the following offsets one by one and change the bytes 75 05 at each offset to 90 90

offsets	From	To
14DFC	75 05	90 90
14E3A	75 05	90 90
14E78	75 05	90 90
14EB6	75 05	90 90
14EF3	75 05	90 90
14F1E	75 05	90 90

There are six possible types of questions the game can ask about a character and each has it's own routine. The above will fix all of the routines.

IBM Softkey for...

Earl Weaver's Baseball v1.5

?

Be sure to backup your the program disk before starting and use the back up for the softkey. Modify only the backup copy!

REN WEAVER.EXE WEAVER
DEBUG WEAVER *Load program into DEBUG*
S 0000 FFFF 74 E3 *Search for 1st protection pattern*
`xxxx:yyyy`

The search will return one address. If more than one is returned this softkey may not work.

E yyyy 90 90 *Edit the contents of the returned address*
S 0000 FFFF 75 0D 3B *Search for 2nd protection pattern*
`xxxx:yyyy`

Again, the search will return one address. If more than one is returned this softkey may not work.

E yyyy EB 04 *Edit the contents of the returned address*
W *Writing XXXX bytes*
Q
REN WEAVER WEAVER.EXE

Now try to run the new (Hopefully) unprotected version of Earl Weaver's Baseball. Just push ENTER when asked for secret codes.

IBM Softkey for...

F-15

?

Requirements:
DEBUG.COM (found on your DOS disk)

1. Start up DEBUG.

DEBUG

2. At the DEBUG prompt (-), insert your copy of F-15 into drive A: and enter the following command lines:

L 0 0 2A 1
F 99 L 10 20
W 0 0 2A 1
Q

When asked for your code just hit ENTER! To check your copy, after hitting ENTER for the code prompt, try to switch between weapons (try pressing 'M').

unClassifieds

How to place an UnClassified Ad

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

Special Graphics Instructions: The first three words of the first line are printed in bold for free. If you want other words bolded, use 5 characters less per line. Use 10 characters less per line if you have a lot of uppercase bold letters. Bold letters are wider than normal. If the typed copy does not show bold, circle the words you want bolded and, on the side, write BOLD. If you want a line centered, write **CENTER** next to that line. There is no charge for centering any line.

You must check your ad for errors, the first time it runs. Errors on our part will be corrected, then, for free. Errors or changes on your part will be charged a $5 processing fee.

★★★★ **New Rates** (per line) ★★★★

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

The minimum order is $5.

- Our liability for errors or omissions is limited to the cost of the ad.
- We reserve the right to refuse any ad.
- Washington state residents add 7.8% sales tax.
- Send a check or money order (funds drawn on US bank only) for the entire amount to:

COMPUTIST unCLASSIFIEDS
33821 East Orville Road
Eatonville, WA 98328

WANTED

"Most Wanted List" Software

Need help to deprotect a disk

Softkey hobbist is interested in acquiring copy protected software to deprotect. Good track record, many successful attempts. Original disk will be returned along with softkey for COMPUTIST. Especially interested in older software (pre-1988) but will give any disk a shot. I'm especially interested in:

Drol --Broderbund
Serpentine --Broderbund
Spare Change --Broderbund
Wings of Fury --Broderbund
Star Cruiser --Sirius
Space Eggs --Sirius
Falcons --Picadilly
Microwave --Cavalier

System: Apple IIe, 128K. Send disk to:

Rich Etarip
824 William Charles, Apt #2
Green Bay, WI 54304

Buy/Rent IIgs Software

Most Rentals $5
Average Sale Price $15
Over 200 titles (none public domain)
Send $5 for catalog and membership

GSoft
425 Loch Devon Drive
Lutz, FL 33549

Wanted

Photocopies of instruction manuals for these Beagle Bros programs, now in the public domain: Beagle BASIC, DOS Boss, Fatcat, Flex Type, Font Mechanic, Frame Up, Shape Mechanic and Utility City.

Jeffrey K. Wagner
Firelands College
901 Rye Beach Road
Huron, OH 44839

99¢ IBM Shareware

100's of titles to choose from! Send $1 for catalog, double refunded with first order!

UnKnown Shareware
9944-A Belle Fourche Ave
Ellsworth AFB SD 57706

Why Pay More ? ? ?

Apple Stuff

Lots of Apple Software and books. Send SASE for list to:

Brent Michalski
9944-A Belle Fourche Ave
Ellsworth AFB SD 57706

RDEX Contributors

Aaron	Culliney	20
Alan	Chaney	20
Rich	Etarip	15
The	Guardian	10
Michael A.	Horton	6, 14
Jeff	Hurlburt	4
	Krakowicz	10
Michael S.	Pollock	12
Eric W	Taylor	12
	Unknown	22
Don	Westcott	22

Apple Most Wanted

65 Airheart ..... Broderbund
63 Alcon ..... Taito
74 Algebra Shop ..... Scholastic
63 Alien Mind ..... PBI Software
73 American History Explorer Series ..... Mindscape
75 Anchorman ..... Virginia Reel
74 Animals of the Past ..... Focus Media
72 Ankh ..... Datamost
73 Ant Farm ..... Sunburst
67 Aquatron ..... Sierra
63 Bad Street Brawler ..... Mindscape
73 Bank Street Beginner's Filer ..... Sunburst
73 Bank Street School Filer ..... Sunburst
63 Beyond Zork ..... Infocom
65 Bilestoad ..... Datamost
69 Blue Powder - Grey Smoke ..... Grade
74 Birds - Trees & Flowers ..... Focus Media
63 Border Zone ..... Infocom
67 Bouncing Kamungas ..... Penguin
66 Boxing ..... ?
65 Bureaucracy ..... Infocom
67 C'est La Vie ..... Adventure International
69 Caverns of Callisto ..... Origin
69 Checker ..... Odesta
69 Chess 7.0 ..... Odesta
81 Chessmaster 2100 IIe .. Software Toolworks
75 Clue Master Detective ..... Leisure Genius
63 Cosmic Relief ..... Datasoft
65 Crime & Punishment ..... Imagic
81 Crosscountry USA School Edition . Didatech
69 Crossword Magic v4.0 ..... ?
69 Cybernation ..... Nexa Corp.
74 Decimal Dungeon ..... Unicorn
74 Decisions Decisions: Colonization v1.0 ..... Tom Snyder Productions
69 Delta Squadron ..... Nexa Corp.
67 Desecration ..... Mind Games
66 Disk Optimizer System ..... Nibble Notch
65 Dondra ..... Spectrum Holobyte
69 Dragon Eye ..... Epyx
69 Dueling Digits ..... Broderbund
68 D&D-Master Assistant vol2 ..... SSI
62 DROL ..... Broderbund
67 Epoch ..... Sirius
74 Exploring Tables & Graphs Level 2 (SU) ..... Weekly Reader
67 Evolution ..... Sydney
67 Falcons ..... Piccadilly
68 Factastics Trivia ..... Daystar
75 Final Frontier ..... Softsmith
73 Fisher's Cove ..... Tom Snyder Productions
69 Flt Wars ..... Sirius
74 Fraction Action ..... Unicorn
69 Gemstone Healer ..... SSI
73 Geometric Supposer (the) ..... Sunburst
66 GEOS ..... Berkley Softworks
72 Galactic Gladiators ..... SSI
63 Gladiator ..... Taito
73 Goodell Diamond Caper ..... Tom Snyder Productions
66 GradeBuster 1 2 3 ..... Grade Buster
61 Gutenberg Sr. ..... Micromation LTD.
65 Halls of Montezuma ..... Electronic Arts
67 High Orbit ..... Softsmith
67 Horizon V ..... Softsmith
75 Hunt for Red October GS ..... Datasoft
69 Impossible Mission ..... Epyx
62 Indoor Sports ..... Mindscape
68 Infocomics ..... Infocom
66 Jane ..... ?
63 Joker Poker ..... Mindscape
72 Kabul Spy ..... Sirius
68 Kingdom of Facts Santa Barbara/Thunder Mountain
75 Kobayashi Alternative (The) ..... Simon & Schuster
72 Lane Mastodon ..... Infocom
67 Lancaster ..... SVS
72 Laser Force (IIgs) ..... Britannica
81 The Last Ninja (IIe) ..... Activision
75 L.A. Land Monopoly ..... Softsmith
66 Legacy of the Ancients ..... Electronic Arts
65 Lost Tomb ..... Datasoft
81 M-ss-ing L-nks: Classics old & new ..... Sunburst
74 Mammals - Reptiles & Amphibians ..... Focus Media
65 Manhunter New York IIgs ..... Sierra On Line
65 Mavis Beacon Teaches Typing (gs) ..... Software Toolworks
73 McGraw-Hill Problem-Solving Lvl 5&6 ..... Tom Snyder Productions
67 Microwave ..... Cavalier
73 Mind Castle I ..... MCE Inc.
63 Modem MGR ..... MGR Software
68 Mr. Pixel's Cartoon Kit ..... Mindscape
73 Mystery of Hotel Victoria ..... Tom Snyder Productions
63 National Inspirer ... Tom Snyder Productions
75 Neptune ..... Softsmith
66 Observatory (The) ..... Mindscape
74 Ocean Life ..... Focus Media
66 Odin ..... Odessta
63 Operation Wolf ..... Taito
68 Pensate ..... Datasoft/Softdisk
69 Phantasie II ..... SSI
67 Phantoms 5 ..... Sirius
67 Pig Pen ..... Datamost
74 Plants & Animals of the Desert Focus Media
75 Prince of Persia (5.25") ..... Broderbund
67 Project: Space Station ..... Avantage
75 Promethean Prophecy (The) ..... Simon & Schuster
67 Pulsar II ..... Sirius
68 Pure Stat Basketball ..... ?
62 Quadratic Equations II ... Olympus Educational Software
81 Quarter Mile IIe ..... ?
63 Questron II ..... Electronic Arts
68 Rails West ..... SSI
63 Renegade ..... Taito
67 Rescue Raiders ..... Sir Tech
63 Rocket Ranger (IIgs) ..... Cinemaware
69 Roundabout ..... Datamost
75 Russki Duck ..... Softsmith
63 S.D.I. (IIgs) ..... Cinemaware
62 Sea Stalker ..... Broderbund
67 Serpentine ..... Broderbund
74 Seven Cities of Gold ..... Electronic Arts
68 Skeletal System ..... Brainbank
63 Sky Shark ..... Taito
63 Sound Song & Vision ... Advanced Software
67 Space Ark ..... Datamost
62 Spare Change ..... Broderbund
67 Spectre ..... Datamost
62 Speedy Spides ..... Readers Digest
67 Star Cruiser ..... Sirius
63 StickyBear Math: Add & Subtract ..... Optimum Resources
68 Stickybear GS Versions 3.5 ..... Xerox
67 Succession ..... Piccadilly
65 Superstar Ice Hockey ..... Mindscape
61 Superstar Indoor Sports ..... Mindscape
74 Surveys Unlimited ..... Mindscape
68 Talking Text Writer GS ..... Scholastic
68 Tangled Tales ..... Origin Systems
81 Test Drive IIe ..... Accolade
69 Tetris (IIe) ..... Spectrum Holobyte
72 Theatre Europe ..... PBI
74 The Other Side v2.0 ..... Tom Snyder Productions
81 Think Quick! v1.2 ..... Learning Company
65 Thunder Chopper ..... ?
63 Ticket to Washington D.C. ..... Blue Lion Software
74 Time Explorers ..... Gameco
74 Time Liner v1.1 .... Tom Snyder Productions
68 Tomahawk (IIgs) ..... Datasoft
69 Track Attack ..... Broderbund
68 Triad ..... Thunder Mountain
72 Triango (IIgs) ..... California Dreams
68 Trinity ..... Infocom
73 Unicorn 5.25" software ..... Unicorn
73 Vincent's Museum Tom Snyder Productions
68 Volcanoes v1.8 . Earthware Comp. Services
66 War in the Middle Earth ..... Melbourne
67 Wayout ..... Sirius
63 Wings of Fury ..... Broderbund
63 Wizardry:Return of Werda ..... Sir-Tech.
68 Word Attack Plus (IIgs) ..... Davidson
65 Works (the) ..... First Star Software
67 Zenith ..... Softsmith

IBM Most Wanted

84 Ace of Aces ..... Accolade
84 Bar Games ..... Accolade
84 Colony ..... Mindscape
84 Don't Go Alone ..... Accolade
75 Empire ..... Intersil
84 Final Orbit ..... Innerprise
72 GBA Championship Football ..... Electronic Arts
68 Graphitti ..... George Best Phillips Academy
63 Heros of the Lance ..... SSI
84 Hardball II ..... Accolade
84 Harmony ..... Accolade
84 Hat Trick ..... Capcom
84 Heatwave ..... Accolade
84 Ishido ..... Accolade
84 Jetfighter ..... Velocity
84 John Elway's Quarterback ..... Melbourne House
72 Kings Quest III ..... Sierra
84 M1 Tank Platoon ..... Microprose
84 Monty Python's Flying Circus ..... Mastertronic
72 Operation Wolf ..... Taito
84 Outrun ..... Sega
84 Phantasm ..... Exocet
84 Powerdrome ..... Electronic Arts
72 Radio Baseball ..... Electronic Arts
84 Sim City ..... Maxis
84 Space Harrier ..... Sega
84 Stormovik ..... Electronic Arts
84 Test Drive III: The Passion ..... Accolade
84 Third Courier ..... Accolade
84 Troika ..... Paragon
84 Wayne Gretzky Hockey 2 ..... Bethesda
84 World's Greatest Baseball Game ..... Epyx/Keypunch

#79• The Product Monitor• *Bitkeys:* Kabul Spy• *Softkeys:* ABM• Algebra 1-6• Cause and Effect• Chemistry: Series I• Computer Generated Mathematics Vol. 2• Cribbage• Designer Puzzles• Dungeon Master Assistant Vol. 2• Economics• Genesis• Gin King• Go• Graphmaster• Hard Hat Mack• Hi Res Computer Golf• Integer Arcade• Laser Bounce• Mammals Reptiles and Insects• Master Grades• Mickey's Crossword Puzzle Maker• Mind Benders• Missing Links• Non-Western Cultures• RoboCOP• Safari Search• SAT Score Improvement Series• Special Product and Algebraic Factors• Stickybear GS Talking series Talking Alphabet• Talking Opposites• Talking Shapes• Task Force• Teacher's Toolkit version 3.1• The Great Knowledge Race• The History of Europe• The Solar System• The Time Tunnel• Thief• TrianGO• US History• Wasteland• Water and Weather• Who Am I?• Word Problems for Algebra• Worksheet Generator• Writing Chemical Formulas• Your Body• Your Body: Series II• *Playing Tips:* Baneful Tales• Elite• *Mac Features:* Mac Hard Disk Ejection Fix• *Mac Softkeys and other Patches:* ABCBase• Animation Toolkit1• Aztec C 1.0• Aztec C version 1.00c• Championship Boxing• Chart• Checkminder• Cutthroats• Cutthroats alternate• Deja Vu• Desk Toppers• Dollars & Sense• Dollars & Sense alternate• Electric Checkbook• Excel• Excel alternate fix• Fact Finder 1.0• Factfinder• Farenheit 451• Feathers & Space• File• FileMaker• Filevision• Filevision alternate• Forecast• Frogger• FunPak• Gato• Grid Wars• Griffin Terminal• Haba-Comm• Haba-Comm alternate• HabaCheckMinder• Habadex 1.1• Harrier Strike Mission• Hayden Speller• Hayden Speller alternate• Hippo^C Level 1• Hitchhiker's alternate• Hitchhiker's Guide to the Galaxy• Home Accountant• Legacy• Lode Runner• Mac Fortran• Macattack• MacChkrs/Rvrsi• MacCommand• MacDraft 1.0• MacDraft 1.1• MacGammon/Cribbage• MacJack/Poker II• MacLabeller• MacMatch• MacPascal (version 1.0)• MacPoker• MacType• Master Type• Master Type alternate• Mouse Stampede• Multiplan alternate• Multiplan version 1.02• OverVue• PageMaker• PageMaker 1.0• Pensate• PFS• PFS File/Report• PFS version A.03• Real Poker• Rogue• Sargon III• SkyFox• Smooth Talker• The Quest• Think Tank• Think-Tank 1.1• ThinkTank 128• ThinkTank 512• Transylvania• Triple Play 1.0• Trivia Arcade• Trivia Fever• Typing Intrigue• Ultima][• Ultima III• VideoWorks 1.0• WellTris• Winter Games• Xyphus• *Features, Notes & such:* COPYA-able Questron II• How to make Thief into a BRUNable file• How to run Task Force on your hard drive• Making Genesis into a single BRUNable file• Making Hard Hat Mack into a single BRUNable file• Making PLATO software run on the Enhanced //e• Multi-Column Print Utility (MCP)• Notes on Battle Chess• Notes on Silent Service GS• Notes on Wildcard II card• Object Module Format (OMF)• ORCA/Disassembler Scripts• ORCA/Disassembler utilities• Other Notes• Running Teacher's Toolkit v3.1 (3.5") on a Laser 128• Task Force on a hard drive and Wings by Vitesse• The Basics of Kracking (part 5): Deprotection of Modified DOS disks• The Basics of Kracking Part 6: Mating Zone & Nibblizing Mysteries• Update on theSilent Service GS v925.01 crack• Xternal Commands for BASIC: CWD (Change Working Directory)• ONLINE•

#80• The Product Monitor• *Features, Notes & such:* Add Copy II Plus file handling to your BASIC program• Comments on the Beginner's Book• Formatting 720K disks as 1.44M HD• How to SAVE hexdumps as CDA's• Logging ProDOS Drives• The Basics of Kracking (part 7)• The Basics of Kracking (part 8)• *Bitkeys:* Black Magic• Guild of Thieves• Gunslinger• King's Quest Series• Leisure Suit Larry• Man Hunter: New York• Police Quest• Realms of Darkness• Saracen• Sierra Boot Disks• Silicon Dreams• Space Quest Series• Ultima V• Wizardry Series• Xyphus• *Softkeys:* Ancient Art of War• Battle Chess• Bridge 6.0• Captain Blood GS• Dinosaur Days v1.0• Empire• Fahrenheit 451• Fay's Word Rally• GATO v1.3• Greeting Card Maker• Hostage• Keef The Thief• Magic Spells v2.0• Magic Spells v2.1• Mickey's Crossword Puzzle Maker• Monsters and Make Believe v1.1• Pipe Dream• Pipe Dreams• Rear Guard• Rendezvous with Rama• Same or Different• Teacher's Tool Kit• Teacher's Tool Kit (IIc)• War of the Lance• Where in the USA is Carmen Sandiego?• WindwalkerGS• Windwalker IIe• *APTs:* Space Rogue• Wizardry III• *Playing Tips:* Countdown• Space Rogue• *IBM Softkeys:* Serve and Volley• Welltris

#81• The Product Monitor• *Bitkeys:* Micro Typewriter• *Softkeys:* Backyard Birds• Balance of Power• Chemistry: Balancing Equations• Chemistry: The Periodic Table• Chuck Yeager's AFT• Equation Math• Estimation: Quick Solve I• Estimation: Quick Solve II• Five-Star Forecast• Fossil Hunter• Grammar Toy Shop• Instant Survey• Micro Typewriter v4.0• Murphy's Minerals• Patterns• Picture Chompers• Probability Lab• Professor Al's Sequencing Lab• Stickybear Shapes (ProDOS 1.5)• Studymate (the grade booster)• Sun and Seasons• The Duel: Test Drive II• Time Navigator• Tomahawk• Windwalker• *APTs:* Where in Europe is Carmen Sandiego?• Where in the USA is Carmen Sandiego?• Where in the World is Carmen Sandiego?• Where in Time is Carmen Sandiego?• *Playing Tips:* Windwalker• *IBM Softkeys:* Crime Wave• Gauntlet II• Stunt Driver• Thexder II• Wing Commander• *IBM Reader Review:* Copyright•

and much more...

For a complete back issue list, send a 75¢ stamp to Computist.

Special Software Sale

(while they last)

These software packages are **NEW** (shrink-wrapped except for the one copy of Sound Master that I opened in order to find out what it was). They're software packages that someone ordered and then canceled and we were unable to return.

SubLogic Scenery Disk 2 (Phoenix, Albquerque & El Paso)

SubLogic

(All Apple II's) **$5.00**

For use with Jet and/or Flight Simulator v2.0. Each scenery disk covers a geographical region of the country and includes major airports, radio-nav aids, cities, highways, rivers and lakes located in that region. Enough detail is available for either visual or intrumental cross-country navigation.

SoundQuest CZ Master

Sound Quest In

(Commodore Amiga) **$10.00**

For use with the Casio CZ-101, CZ-1000, CZ-3000, CZ-5000 and other compatable synthesizers. Included are file management and bank editing features, patch mixing and random voice generation features. Compose and mix your own music using many of the package options available.

Send orders to Computist at the address listed on the Back issue order form below.

Issue	Mag	Disk	Issue	Mag	Disk	Issue	Mag	Disk	Issue	Mag	Disk
Core1	□		22	□	□	46	□	□	70	□	□
1	□		23	○	□	47	□	□	71	□	□
2	○		24	□	□	48	□	□	72	□	□
Core2	□		25	□	□	49	□	□	73	□	□
3	○		26	□	□	50	□	□	74	□	□
4	○		27	□	□	51	□	□	75	□	□
Core3	□		28☆	□	□	52	□	□	76	□	□
5	○		29	□	□	53	□	□	77	□	□
6	○		30	□	□	54	□	□	78	□	□
7	○		31	□	□	55	□	□	79	□	□
8	○		32	□	□	56	□	□	80	□	□
9	○	□	33	□	□	57	□	□	81	□	□
10	○	□	34	□	□	58	□	□	82	□	□
11	○	□	35	□	□	59	□	□	83	□	□
12	○	□	36	□	□	60	□	□			
13	○	□	37	□	□	61	□	□			
14	○	□	38	□	□	62	□	□			
15	○	□	39	□	□	63	□	□			
16☆	□	□	40	□	□	64	□	□			
17	○	□	41	□	□	65	□	□			
18	○	□	42	□	□	66✱	○	□			
19☆	□	□	43	□	□	67	□	□			
20	□	□	44	□	□	68	□	□			
21	○	□	45	□	□	69	□	□			

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

☆ Limited supply — first-come-first-serve basis.
○ Out-of-print — only "Zeroxed" copies for sale.
✱ Issue 66 is laser printed on 8½ by 11 paper.

Back Issue Order Form

Back Issue and Library Disk Rates

	Quantity	US, Canada & Mexico	All others
Back issues	5 or less	$4.75	$8.75
	6 to 9	$3.75	$6.00
	10 or more	$3.00	$5.00
Zox back issues*	any qty.	$4.75	$8.75
Library disks	5 or less	$5.50	$7.50
	6 to 9	$4.00	$6.00
	10 or more	$3.00	$5.00

Note: Total back issue and library disk orders to get quantity discounts. (ie. ordering 5 back issues and 5 library disks means that you pay the the quantity 10 price of $3 each for both.)

**Due to the time and effort involved in making Zox copies, their price will remain at $4.75 each for US, Canada & Mexico and at $8.75 for all other Foreign.*

Shipping is included in all the prices shown.

What's a library disk?

A library disk is a 5¼ inch floppy diskette that contains programs that would normally have to be typed in by the user. Documentation for each library disk can be found in the corresponding issue.

•Library disks are available for all issues of COMPUTIST.

For a complete back issue list, send a 75¢ stamp to Computist.

_____ Number of back issues. $ _____

_____ Number of Zox back issues. $ _____

_____ Number of library Disks. $ _____

Washington state residents add 7.8% tax $ _____

Total enclosed $ _____

Name _____

Address _____

_____

City _____ State _____ Zip _____

Country _____ Phone _____

VISA

MC _____ - _____ - _____ - _____ Exp. _____

Signature _____ 84

• US funds drawn on US bank. • Most orders shipped within 5 working days, however please allow up to 4 weeks delivery for some orders. • Large orders are shipped UPS so please use a street address. • Offer good while supply lasts. • Call (206) 832-3055 to use a credit card or send check/money order to:

COMPUTIST 33821 E Orville Road Eatonville WA 98328