78 COMPUTIST

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

Table of Contents

COMPUTIST

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

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

COMPUTIST
33821 East Orville Road
Eatonville, WA 98328
(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.

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

The BBS (Bulletin Board System)

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

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

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

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

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

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

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

United States Code title 17, §117

COMPUTIST

78

U.S. $3.75

Canada & Mexico $7

Editorial message

This issue is late. I know, it's dumb to state the obvious but I wanted to tell you that the next one (#79) will be out early so we can catch up.

This issue also seems to be a bit sparse. Between an extra large Product Monitor, another installment of Krackowicz and a beginners guide to copy deprotection, the softkeys and APT's kinda got lost. But don't worry, they'll be back in time for issue #79.

Postage pains

The USPS has decided to raise our postal rates again. This is especially annoying as the bulletin that they sent me shows that the 3rd Class rates went up more than any other class. Regular subs are sent by 3rd Class mail. After this issue is mailed, I will be recalculating the cost of postage. The new rates will be printed in issue #79. If you want to save a buck or so, send in your renewal early, before the rates go up.

New Technology?

I get a lot of magazines and technical publications. Some of the info is interesting and could apply to Apple IIs. A reader commented that he would be interested in a summary of any new technology that might apply to Apple II and Mac computers. Is this something that a lot of you would be interested in seeing? Write and let me know what you think. Write and let me know of any new technology that you have heard of.

As an example, a Japanese company has demonstrated a 64Mbit chip. That's right. A memory chip that stores 64 times more data than the 1Mbit chips that are so common today. The really interesting part is that the chip only needs 1.5 volts to work and the speed is 50 nanoseconds. The lower voltage is good news for portables and the speed is faster than the 1Mbit chips commonly used today. (80ns vs 50ns)

Stack eight of these 64Mbit chips together and you have 64 Mbytes. That opens the way for a microcomputer to have 64Mbytes of RAM. That's larger than most of the hard drives that computers are using today. (40Mbyte hard drives are the most popular size according to Micro-D's dealer newsletter.)

Sounds good to me. More memory and a multitasking operating system have been on my wish list for some time. What do you think?

The Court, the IRS and Computist

We finally made it to court and (wonder of wonders) everyone else did too. It took seven tries to get there, almost a year of delays, and it only took the court about 10 minutes to approve our plan of reorganization. And our lawyer said this was a simple Chapter 11 case. (Unbelievable.) So, the IRS gets to charge us interest & penalties and we get to keep printing Computist. That's what counts! If you can spare so bucks, send a donation to our IRS fund so we can pay them off quickly and not have to pay too much interest and penalties.

What's next

Well, Vince Andrews has dropped off a bunch more CDAs with lots of source code and explanations. One Computist reader sent a bunch of MAC softkeys. And there are a dozen disks that I'm going to check out as soon as this issue is put to bed.

Also, I've written to several suppliers of (new and used) Apple II parts. When I get all of the info I'm going to tell you how to upgrade your Apple II and II+ into a IIe or how to upgrade your IIe into a IIgs. Hopefully, since this is mostly used equipment (reconditioned with warranties), the cost will be reasonable whatever that means.

See you in issue #79!

Table of Contents

Jeff Hurlburt TX

The PRODUCT MONITOR

RATINGS

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

Now that you've done with tannenbaums and roasting chipmunks on the open fire... Finally, you can relax and start cashing in on those juicy sales at the local computer emporia.

Worlds of Ultima: The Savage Empire

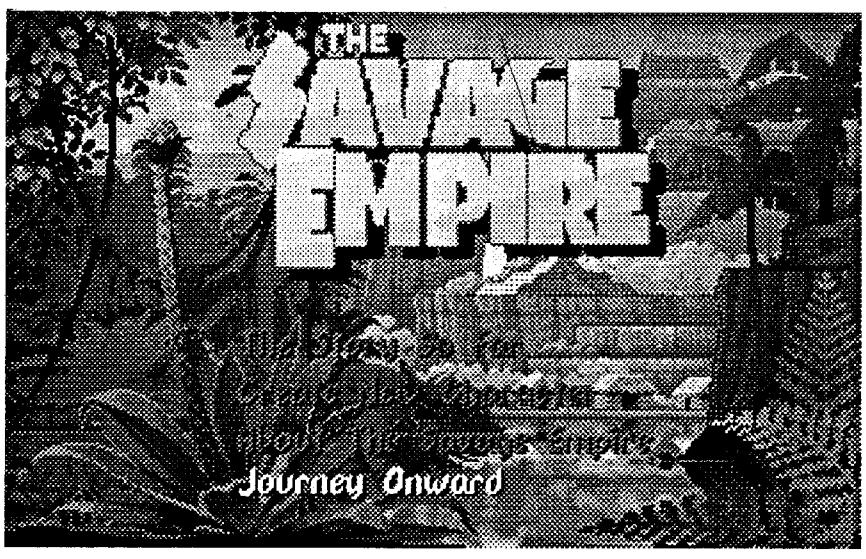
★★★

$59.95 for EGA-VGA 640K PC; 10Mhz, EMS mem recommended
Origin

Tump, tump, te-tump-tump-tump, ... strange drumbeats, twittering birds, exotic music and brightly lit jungle scenery all tell you "I'm not in Ultima anymore". Actually, though, you are— only here the wizards are called shamen, the major monsters are prehistoric beasts, and no one is quite sure where "here" is! According to the 44-page pulp-magazine-style scene-setter booklet ("Ultimate Adventures", Vol 59, No. 11), it all starts back in mundanesville with a too-real dream of a native beauty battling some kind of dinosaur, an oddly deformed moonstone, AND an experiment gone awry in Professor Rafkin's lab. TZZAPPOOM! You, a reporter named Jimmy, and Rafkin are moongated to Eodon, Ultima's version of "Lost World".

The dream turns out to be an accurate forecast; soon you find yourself on a quest to rescue Princess Aiela of the Kuraks (The girl) from the clutches of Darden the Huge, chief of the Urali tribe (The bad guy). Jimmy and Rafkin should be in your party; but your group was separated soon after arriving. You'll have to find them (especially since Doc Rafkin may be able to tinker up some primitive rifles and grenades to even the odds). Then, there's the 500 square feet or so of lab that got teleported; it's out there somewhere, just loaded with useful stuff. Fortunately, there's no shortage of primitive weaponry and food— you can walk into any hut and pick up whatever you want; and some of the 'monsters' (like tigers and dinosaurs) can be skinned and eaten. Better yet, a Kurak warrior named Shamuru has volunteered to join. Funny thing is, the guy is a dead ringer for your old Brittania companion Shamino; he even claims to "know your face, from somewhere ..."!

You do not, naturally, just walk over to the Uralis, bop Darden, and save Aiela. For starters, the roughly 400 square miles of Eodon is swathed in jungle flora, much of it impassable. Some of the fauna, like the man-sized warrior ants and snake women, can be pretty rough too. Numerous rivers and lakes plus limited-access cliff-ringed plateaus further multiply the effective distance of any journey. Besides, of the eight known tribes, only Urali land is unmarked on your map. Most of the tribes are friendly enough; particularly since, as a real live adventurer, you're just the one to trouble-shoot some knotty problem. The Barrako queen, for instance, needs someone to rescue her daughter (another princess!) from a giant ape. The befuddled shaman on an island is sure to be of help, once you recover his mind, stolen by a powerful rival far to the south. ...



Ultima VI' and **'Savage Empire** arrived in the same package. I chose to play **'Savage Empire** first because 1. It has a neato picture on the box; and 2. It's a more recent release. (Also, good, sweet, lovable, handsome King Lord British was getting to be a real pain.) From what I've seen of each game, the formats are virtually identical.

All adventuring activity takes place on a high-detail 'map' showing each party member, any nearby natives or monsters, trees, cliffs, ladders, huts, and items. Your view is top-down, at a slight slant to allow a 3-D perspective. To enhance realism, there is always some animation (e.g. natives move around, rivers flow, etc.). Good AdLib sound effects supplement an excellent musical score; and you'll find such expected embellishments as day-to-night brightness changes. You can converse with each of the numerous personages encountered via text inputs— a picture temporarily replaces the map— otherwise, most adventuring control is via convenient mouse click-and-point. In "Party mode", members of your group automatically follow your lead; but you can split-off and direct individuals.

Savage Empire does not drench its considerable landscape— roughly equivalent in size to several 'old Ultima' continents— with hordes of monsters. Probably, it's just as well. Lacking a tactical 'break-out' display, combat in **'Savage Empire** is decidedly clumsy; not, in fact, much better than **Ultima I**'s large-map bump-into-and-bash format. The major benefit of combat, aside from clearing troublesome obstacles, is the resulting Experience boosts. These lead to Level advances, more hitpoints, and improved Strength, Dexterity, etc. attributes. Shaman magic, cast via assorted skulls and plants, is similarly low-key. The Eagle Eye, Heal, Protection, Charm ... spells are very helpful; but, your main 'punch' comes from mundane implements.

The somewhat diminished role of conflict means 'Savage Empire is very good about letting you explore, soak up the atmosphere, and work out solutions in relative peace; and, with a large, attractive, challenge-packed gamescape, there's plenty to see and no shortage of things to 'work out'.

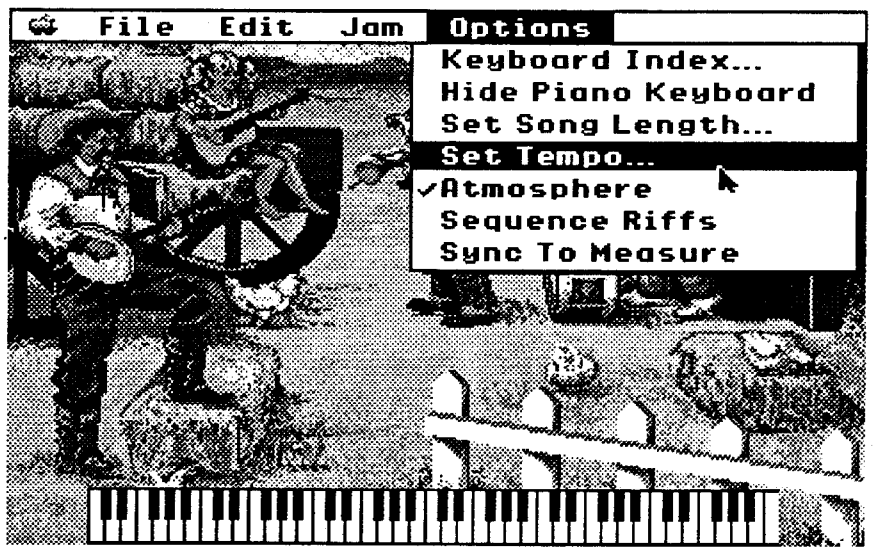
'Ultima-tized' Zork? No; but close. Supplied with fold-out directions card, clever pulp "magazine", and poster-size map, the first "Worlds of Ultima" release is a refreshing, high-entertainment-value new direction from Ultima's creators.

Loom

★★★

$59.95 for CGA-VGA 640K PC
Lucas Films

In some time long past or far in the future, most knowledge and power is the exclusive province of the Great Guilds: Glass makers, Shepherds, Blacksmiths, Clerics, and Weavers. When all but one of the Weavers is mysteriously swept away by some Dark Force, the fabric of time and space is threatened; for it is Weavers who, on The Great Loom, craft the very threads of reality. Alas, the lone remaining Weaver is Bobbin Threadbare, a mere apprentice of the craft!



As Bobbin your quest comes down to finding out who (or what?) is responsible for the disappearance of your fellow guild members; and, once you do, having the wherewithal to handle the situation. Your Book of Patterns (i.e. "drafts" or spells) is nearly blank; but you do know one, the draft of Opening (good for opening things and whisking aside curtains). You also possess a Weaver's Staff; and, as a member of the most powerful of guilds, you command some degree of awe from those you encounter. (Everyone, for instance, knows it is death to peek beneath a weaver's cowl.) Once you've searched Weaver's Island and discovered any drafts to be found there, your first major challenge is to find the way off and into the larger world and greater adventure beyond.

The first time I started **Loom** it seemed certain that Lucas Films had decided to go ahead and produce a computer movie. The game's VGA artwork and smooth animation do place you IN the adventurescape. Point to a hut opening, click, and Bobbin walks in. (The screen shows Bobbin in the hut). If you click on some more distant point (and there is some reasonably direct way to get there), Bobbin will travel the path, diminishing in size as he moves away from the foreground. All complex actions are pre-choreographed; so, if you're near some object you would normally use (e.g. a raft); the program will execute the necessary actions (you climb on). Add a vivid AdLib soundtrack of Tchikovsky melodies and it's no wonder the game is nearly as much fun to watch as to play (or that younger visitors still ask me how Bobbin is doing).

Though **Loom** presents close-ups and verbose text for many of the shepherds, glass makers and others you meet along the way, virtually all player input consists of clicking some choice or playing four-note drafts. A draft is learned by encountering it in some context which hints at its function. For instance, you see sheep being lulled to sleep by a shepherd and a draft is played/displayed. ("Aha", you say, "I'll bet that's the Sleep draft.") To weave a draft you sound the appropriate notes on a staff at the bottom of the screen. (The farther you advance in the adventure, the more notes are added to Bobbin's staff and the more drafts he can use.) You record drafts (in the game's Book of Patterns) in pencil, because, if the adventure is restarted, whoever plays will encounter different sequences. For the musically adept player, an "Advanced" mode presents drafts as sound only.

Supplied, with stereo 'scene setter' tape, on six (360K format) diskettes, **Loom** spans crystalline cities, lush pastures, forge-lit castles, and the labyrinthine caverns of a dragon's lair. The game is too easy to challenge experienced adventure gamers for more than a few hours; but, of course, this was never the objective. **Loom** is the computer version of the richly illustrated fantasy lore book on your coffee table: nice to browse through and fun to share, especially with younger computists.

Jam Session

★★★

$49.95, for 768K Apple IIg
Broderbund

If the weekends and holidays seem to bring visitors flocking into your Computer Room, it's a safe bet that at least one will come up with something like: "What's all this stuff about 'IIgs sound'?" You might boot-up Activision's "Music Studio" or EA's "Instant Music"; and your visitors will be suitably impressed. OR, you can go the 'No Mercy' route, start up Broderbund's **Jam Session**, click on "Chicken Jamboree", "Walkbass", "Jamtana", ... and just blow 'em away!

Jam Session doesn't merely play twenty lively << STEREO >> arrangements, it SHOWS the players, in sound-synced animated super-res! The five "groups" include Country-Western, Rock, and Heavy Metal (four members each), plus a Blues duo and a Classical pianist. Adding to the fun, you get tempo and keyboard-jam control of riffs (runs of notes played as embellishment), individual "scale" notes, and sound effects (e.g. chicken "cluck, cluck"'s). These are arranged, by instrument, in the upper four keyboard rows— A, S, ...,' may be "Guitar riffs"; Z, X, ..., / might be "Synth scale"; etc.. The Shift key switches to four more banks for a total of eight per song. Clicking "Edit" lets you add, delete, and change riff notes; and these changes can be saved to disk.

As the name implies, **Jam Session** is chiefly for 'play along', not composition. Riffs can be fairly long and can greatly change a song's sound; but the basic melody remains. Similarly, clicking "Keyboard Index" shows current bank assignments; but the display cannot be maintained during play. You could make detailed notes and actually 'arrange the music'. Mainly, however, you're supposed to 'get a feel' for what's where and 'wing it' like the other members of the group. Sometimes, you'll sound 'just okay'; sometimes, especially with a little practice, everything will click into place and you'll sound Great! And, when you're ready for that 'special performance', just switch on the program's Record option. Your masterpiece is 'in the can' ready for replay and saving to disk.

At present, you can not import entirely new songs or graphics; nor can you change group-song matchups or instrument sets. That is, pending release of a FULL-featured Editor utility, the JS package doesn't do Everything. Jam Session does deliver loads of entertainment, to say nothing of real IIgs

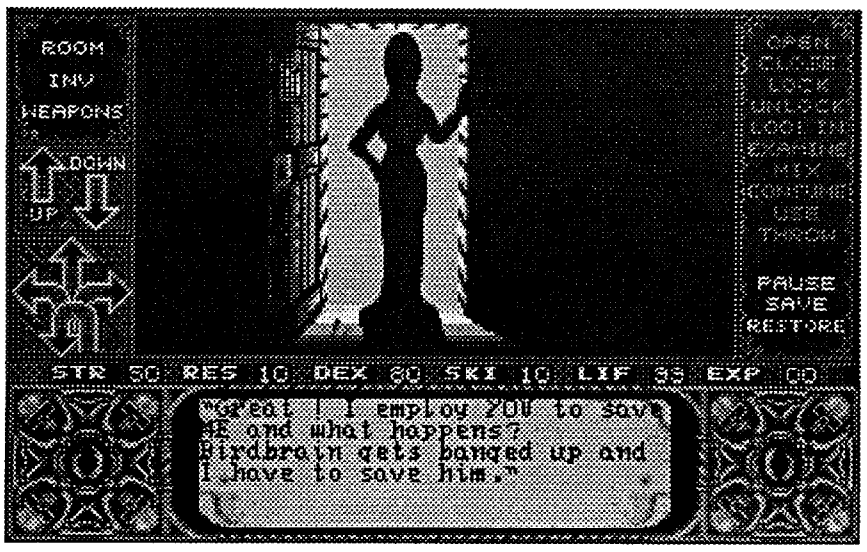
user Satisfaction: "THAT's Neat!", "Wow!", etc., etc.!

Elvira, Mistress of the Dark

★★

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

When Elvira inherits the old family castle, she immediately decides the sprawling three-story keep, towers, and battlements will make the perfect horror-theme nightclub. (As for the dungeon, torture chamber, and catacombs.., who knows?! If "Elvira's" is a big enough hit, it could be 'out with the cells and in with the sell'!) Unfortunately, the 'old family' has other plans. Like, when it comes to hot nightspots, these guys might as well be dead— well, in fact, they ARE dead, sort of. Thanks to great, great grandma (and sorceress) Imelda, there's a vampire upstairs, a zombie in the kitchen, a werewolf in the stables, gremlins in the hedge maze, and ghoul guards roaming the grounds. If someone doesn't find The Chest soon and get rid of Imelda, things could get REALLY bad!



You already know whom Elvira has hired to de-creep her castle. Your quest is to collect the six keys needed to open a hidden chest, find the chest, get the magical implements, and zap Imelda. There is no "time factor" as such; but the ingredients for needed spells are limited and your attributes (e.g. Strength, Dexterity, Hit Points, ...) are difficult to maintain. (Only weapons Skill improves steadily with experience.) Eventually, your resources will run out. The challenge is to complete the quest before this happens.

Imagine a single-character, relatively 'open-air' version of FTL's "Dungeon Master"/IIgs, and you've got a pretty good grasp of **Elvira'**. Basically, 'what you see is what you play'. This includes mouse 'click-and-look' exploration of the richly detailed castle rooms, sheds, moat, gardens's, etc. and combat. The latter is real-time and pits you against animated opponents in semi-arcade one-on-one slash-and-parry duels. (Magic can be used; but only when an enemy is first sighted, before he/she gets within 'hand combat' range.) You'll see ghoul guards wield their swords, werewolves transform and charge, Imelda's handmaidens cast spells, vampires squirm and wail as you drive the stake home, and much more!— like Imelda herself, a killer falcon, brown-robed fanatics, and lots of Elvira!— all in animated VGA accompanied by AdLib sound effects and music. Also, besides a maggot-infested corpse, expect to see yourself with throat cut or torn out, face blue from lack of air, head bobbing in a kettle, ...— i.e. this game is not for the weak of stomach.

In the realm of PC adventuring, **Elvira'** is 'leading edge' stuff guaranteed to draw you into the scenario. Regrettably, unless forewarned (and forearmed), the game is all too likely to leave you wandering, winless, and wanting an explanation or two. The problem certainly isn't "Realism"; the problem is "Fairness". **Elvira'** 's world builders didn't quite finish the job. Every role-play adventure begins with an unspoken contract: 1. If you use available information and resources in a moderately adept manner, you can succeed without recourse to outlandishly good luck or ESP; and 2. Should you commit some fatal blunder, you will quickly be killed-off; NOT allowed to continue, zombie-like, on and on in ignorance. That a player might, deep into **Elvira'**'s quest, have good reason to wonder if he or she is 'lost', 'dead', with no hope of winning— this is bad enough. That your doubts are well founded, is even worse. You do not, for example, have any way of knowing that using up your crossbow bolts (e.g. to zap maze gremlins) leaves no way to kill the werewolf or the Grey Knight. Nor is there a warning that the plants, bugs, and other key spell ingredients are limited, non-renewable resources.

"Okay, NOW I know; so, the problem is fixed. Right?" Maybe. Misusing your crossbow bolts is the only for-sure route to adventurer 'zombiehood'. Running out of magic or destroying an artifact could produce a similar result; but it's less likely. Even so, after restarting the game one or two times, I finally consulted the **Clue Book** and ended up using it as a kind of second manual. **Elvira'** is very good about permitting any number of speedy Saves and Restores; it is rather weak when it comes to supplying critical information. A very determined, experienced Computist reader might succeed without the "optional" booklet's maps and hints; but, I doubt that the experience would be especially entertaining. Too much **Clue Book** stuff belongs in the game program or in the manual as "Journal" readings, etc.. When you buy the game, get 'the book'.

After finishing **Elvira'**, I was mulling over some notes and trying to decide how the game stacks up when in walked Gorbash and Baywoof. Neither has won yet; but, despite having to rely upon the **Clue Book**, both gave it enthusiastic 'thumbs up's. With hints and maps, the game still offers a solid challenge; and, I admit, replaying most of the adventure proved to be nearly as absorbing as the first try. Even my six or seven attempts at the Hedge Maze (before finally 'getting it right') produced more fun than frustration.

Yes, **Elvira'** 's design is seriously flawed (i.e. the game, not the TV hostess). A player must use the Clue Book and is certain to spoil a few challenges. Another notable weakness is that 3-D views fail to show maze and corridor openings to the right and left. Together with the absence of a compass, this makes navigation a good deal more difficult than it should be. On the other hand, 'warts and all', the game IS fun to play— far too much fun to pass up. Supplied with manuals on both 5.25" and 3" media, **Elvira'** 's displays, sounds, and scenario grab you, immerse you in the castle's 'atmosphere', and hold you for 30-40 hours of horribly enjoyable adventuring.

Learning DOS

★★★

$50.00 for 256K PC
Microsoft

There are all sorts of ways for new PC users to get a handle on the Disk Operating System. One is to borrow someone's DOS manuals (Baywoof loaned me a couple) and slough your way though each and every aspect of the system. Granted, there is something to be said for learning-by-screwing-up: pain hurts, and, eventually, you avoid pain by doing things right. A better approach, one a good deal faster and more 'cost effective', is to march down to your favorite software emporium and pick up a copy of Microsoft's **Learning DOS**.

Now, suppose I were to ask: What must an instructional package do to 'work'? "It should be fun"; "It should teach the material"; "It should match the presentation to the user's knowledge"; "It should reward success"; ... Fine. But, if the response was "It should place the user firmly in control of the presentation", then give yourself a pat on the nose. If, after all, you want to become a skilled DOS user, then putting YOU in charge of the 'what' and 'when' of the presentation is the surest way to guarantee success.

To accomplish this, **Learning DOS** first divides things into handy topic areas (e.g. Getting Started, Working with Files, Organizing Your Files with Directories, Using Installed Applications, ...) and then lets you decide where to start. 'Really new' PC users can begin at square one, with a key-by-key introduction to important keyboard functions. OR, you might try the Self Test and zero-in on the areas that need the most attention. Throughout the presentations, you can always PgUp to go back a frame or press Ctrl and choose from Print screen, return to the main Menu, hop to lesson Troubleshooting, see the Index ... or just Quit for now (with 'Save') and resume your DOS quest after lunch.

Each lesson is packed with descriptions, examples, and chances for you to practice using commands and seeing the results. All of this, including work files and directories, is simulated; so, for example, there's no need to worry when you try your first FORMAT. The package automatically adjusts coverage for the DOS version under which you boot (through version 4.00) and includes separate sequences for users with a hard disk and those intending to use a machine without one. (For-hard-disk lessons will run on a non-hard disk machine.)

The proof of LD's effectiveness came in going through the lessons. Despite hours of pouring through manuals, just a couple of LD sessions turned up several new, very useful 'discoveries'. (One of the big problems in learning-by-manual is all those questions you don't know enough to ask.) Better yet, the package includes a DOS Quick Reference you can copy to hard disk. Now, whenever you need to know what a command does, its correct format, see some examples, etc.; you can enter HELP and viola!, the Reference is ready. Supplied with 34-page startup manual (itself, a handy DOS beginner's reference), **Learning DOS** is every new PC user's first genuine software bargain.

OmniKey/ULTRA

Keyboard for standard or (with optional key caps) Dvorak layouts

★★★★

$149.00 for PC XT, AT, Tandy 1000SX, AT&T 6300, and PS/2 except model 25 Northgate Computer Systems

Like just about everyone else, when we went shopping for a PC our attention focused upon factors like speed, display capabilities, hard disk performance, and sound boards. The unit's "standard 101 type" keyboard might just as well have been invisible. Toward the 'close', the salesman trotted out an optional same-price keyboard, which won approval because it evidenced a slightly less mushy feel; and that was that— NOT a cool move! In retrospect, it's obvious that no secretary or other word processing pro would make such a mistake. They understand the importance of a crisp, sure 'feel'.

Partly, of course, the problem is that to even get your hands on a good keyboard in many shops, you must specifically ask to try one. It also helps to know what to look for. As noted in past joystick reviews, the major source of user discomfort and fatigue is not high tension. Most often, when a low quality, otherwise "okay" stick leaves the user oddly worn out after relatively little action, the culprit is absence of reliable feedback. This is the "shadow boxing syndrome": the less a joystick (or mouse + pad, or keyboard) 'tells' you about its status via touch, pressure, sound, etc., the more over-movement and extra effort you must expend to use it. A good keyboard 'talks' to your fingers and to your ears.

Northgate knows about feedback; so, all **OmniKey** series keyboards employ ALPS click/tactile key switches. When you press a c/t key 1. you feel the key move down; 2. you feel and hear a THLICK! as the switch activates; 3. you feel and hear the key 'bottom out' at the end of the press. Your keystroke is very similar in principle to the recommended correct golf swing or tennis stroke: power, contact, follow-through. The result is a natural, comfortable 'feel' vastly superior to the mash-and-clack (all-power stroke) response of low-quality units.

The 112-key **'/ULTRA** places Function Keys 1-12 to the left (like the **OmniKey/Plus**) and adds Special Function Keys 1-12 along the top. The latter, via an SF Select key, can be set to duplicate the Function Keys or as Shift +, Ctlr +, or Alt + Function keys. A second 'setup key', Rate Select, lets you set Auto Repeat rate (.54 - 125 characters/second), the Delay between keypress and start of repeat (.20 - 1.85 seconds), and startup KB validation time (.008 - 1.85 seconds). These settings require just two keypresses. The **'/ULTRA** 'wakes up' with SF keys set as Shifted Function keys, a 10.9 char/sec repeat rate, .5 second Delay, and .008 second Validate.

To accommodate MS DOS users, the \ and * keys are placed within easy reach just below **Enter**. Rear panel DIP switches let you swap these keys (to move \ to the bottom row) as well as rotate the positions of left Ctrl, Alt, and Caps Lock. A key cap puller and the needed alternate key caps are included. (The unit arrived with left Ctrl in row 3, which, as an also-IIgs user, suits me just fine.) To help avoid the common "U>S>A>", etc. typo, you'll find a Comma/Period Lock key; and, for speedier numeric entries, Enter is duplicated just to the right of the numeric keypad. **'/ULTRA** adopts the generally preferred diamond Cursor key arrangement; and, for those used to the inverted-T pattern, sets the center "Omni" key to duplicate the Down Arrow function. Insert and Delete appear as extra-size keys just below the cursor diamond.

Done in off-white "computer chrome", **'/ULTRA** boasts a slim 20.25" x 7.25" table-hugging 5lb. metal case equipped with rubber-tipped flip-out angle adjusters. Low-glare white or gray key caps and LED status indicators in-set on a sloping panel above the numeric keypad round out the attractive design. Rugged, pretty, and a pleasure to use, the **OmniKey/ULTRA** lives up to its name from the moment you plug it in. Complaints? Well, it would be nice if the 'wakeup' Key Repeat rate were higher; and, perhaps, the chord should be powder blue instead of gray. Neither 'weakness' appeared to bother the first visitor to try the unit. After a few minutes of DOS diddling, Baywoof just stopped and commented: "You know, this keyboard has a NICE feel!" I suspect his old 'part of the package' board is headed for the shelves.

Supplied with comprehensive manual, chord, and dust cover, the **'/ULTRA** does carry a higher price tag than the average plastic-cased "101 type" unit. Still, this is a classic 'you get what you pay for' situation. Whether shopping for a new PC or upgrading an established system, you owe it to yourself to check out the **OmniKey/ULTRA**. As the chief connection between you and your PC, your keyboard can dramatically impact productivity and enjoyment of a rather sizable investment. You might as well get a good one.

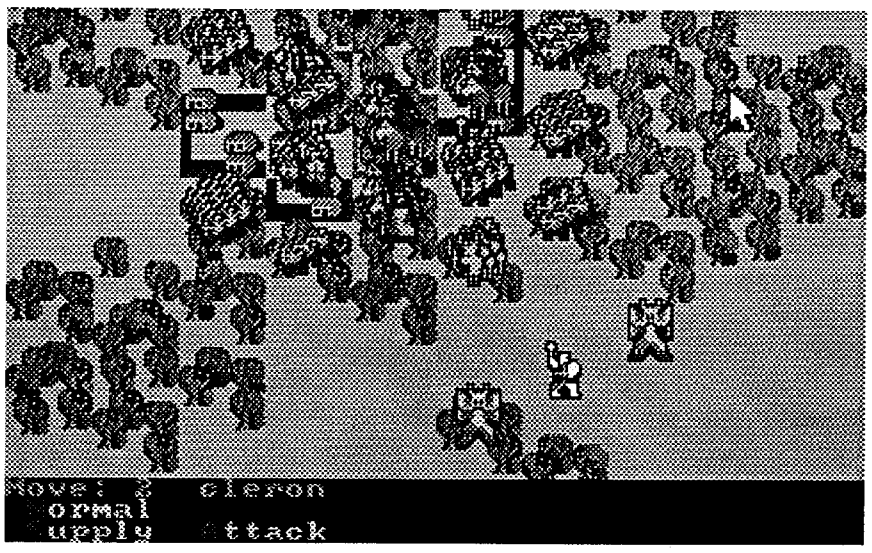
(Other OmniKey boards include the / PLUS at $119.00, /102 at $99.00, and /101 models at $89.00.)

Sword of Aragon

★★★

$39.95 for CGA-EGA 384K PC
Strategic Simulations

As heir to the Duke of Aladda (recently slain in an orc raid), you inherit his dream: to bring peace to the West and East Realms by uniting the twenty feuding city states of the old Aragonian Empire. This is NOT, you soon realize, as easy as it sounds. Several strategically placed cities have been under the rule of orcs, goblins, giants, trolls, or titans for so long that they have become veritable monster fortresses. Just to the north, a power-hungry despot, Pitlag the Pitiless, has begun to subjugate peaceful neighbors; and Aladda is certainly 'on the menu'. Fortunately, Lord Pitlag's logging operations have won the enmity of the West Elves; but you have yet to show the mysterious forest dwellers that Aladda is any different. The Dwarves and Plainsmen are similarly suspicious of anyone claiming altruistic motives.



Your big problem is Lucian III of Tetrada, the old Empire capital in East Realm. He regards Aladda and other West Realm cities as 'lost', barbarous, and fit only to serve as pawns in his games of assassination and conquest in the wealthy East. Orcs, dark mages, and demons are a sampling of those with whom Lucian makes common cause, to say nothing of his evil human allies. Aside from ruling the largest city with the biggest army, Lucian poses another kind of obstacle: he is, de facto, "the Emperor"; YOU are merely one of many lords, and, at that, of a West Realm city state! Part of your quest (i.e. the "adventure part") is to acquire the Crown of the West, Scepter of the East, and other major symbols of legitimacy.

Sword of Aragon is, primarily, a strategy and tactics challenge combined with a hefty dose of "Santa Paravia'" economics: rich cities with large, contented populations and developed industries produce the biggest incomes and the most powerful armies. Each turn is one month, during which you spend income to build production, defenses, and armies in cities under your control. You also cement agreements with potential allies, move individual "commanders" and multi-unit armies (shown as stacked icons on the large, scrolling 'empire map'), pursue wealth-enhancing rescue and monster-removal missions, initiate major invasions, and respond to various large and small-scale attacks.

It is moderately rewarding to see city incomes grow, but the big payoff for your efforts is the chance to direct your custom-equipped and trained units of infantry, bowmen, cavalry, (horse bowmen, mounted infantry, etc.) together with individual Knight, Warrior, Ranger, Priest, and Mage commanders in battles against computer-managed adversaries. Set on scrolling, multi-screen, 'tactical level' maps (complete with trees, bridges, walls, ...), these combats feature partially-animated unit icons, flexible click-and-move controls, full-range equip options, and ready access to spell casting. (Depending upon Level, Mages, Priests, and Rangers can employ such powers as Teleport Unit, Pyrotechnics, Disintegrate, Heal, Create Tower, etc..) '**Aragon** does not, however, favor a 'special powers'-based strategy. Mages and Priests are very important, but slow to develop and few in number. The core of your punch remains the 'grunt': chiefly, heavy infantry backed by bow units.

It should go without saying that one does not devote large chunks of column space to hints and advice (re. issue #75) for a game one does not like. As noted in the earlier column, you may encounter one bug: when responding to the initial game-protection question, the correct answer will sometimes be the word after the one requested. Other complaints include sketchy documentation of unit characteristics and overly slow income growth. (Again, see issue #75 for 'cures'). So much for the 'warts of Aragon'. SSI's **Sword of Aragon** is a unique, long-play strategy & tactics-plus-adventure challenge offering multiple, speedy Game Save/Restore, decent 'old-PC' sound, and crisp, colorful graphics. Restoring The Empire isn't easy; but it probably supplied as much solid gaming 'fun time' as anything I played in 1990.

Donald's Alphabet Chase

★★★

$14.95 for 128K Apple II
Disney Software

When Disney's new alphabet game arrived, I popped it into a drive and started playing with barely a glance at the manual. The result was a 'different kind' of gaming experience.

Moving from room to room in and around Donald's house, the player presses alphabet keys to help Donald find and chase down pesky "silly letters" (who swing from light fixtures, scamper along fences, hop on furniture, etc. before being collected). I quickly discovered that, when you get all the letters in any location (listed in a box at the screen's upper left corner), the program plays a portion of the Alphabet Song; so, I tackled the chase on a room-by-room basis. One can, however, press any un-caught letter's key; and Donald will go to the correct room and find the letter selected. "Cute hires artwork, nice animation, decent 'old-Apple' sound", I decided, "but, really, too easy— there's not even a score! Only a three-year-old could get much from this kind of challenge. .. Aha!!" Sure enough, right on the box it says "Ages 2-5, Preschool".

A "Challenge", to be sure, is where you find it. For young pre-readers, pressing alphabet keys— especially, A key to get a particular letter— can be a genuine challenge as well as a valuable learning experience. (In Center for Giftedness parlance, we are talking about developing eye-hand coordination, fine-motor skills, letter recognition, ...AND slipping in a bit of 'computer literacy' preparation.) That the player gets to try out new skills On A Computer and Make Fun Things Happen— well, this is heady stuff indeed. Add the powerful benefits of parental involvement— a 12-page booklet suggests several parent-child '**Chase**' activities—and you've got a strong, attractive, and entertaining route to building Reading foundation skills.

ORCA/M

★★★

$69.95 for 512K Apple IIgs
Byte Works

All in all, there is probably no more rewarding computing activity than actually taking control of the machine. Mastering any of the "high level" programming languages is a decent start; but, especially on the IIgs, the surest route is mastery of Assembly Language.

Unlike "machine coding", where you go to the monitor and make entries like A5 FE ..., in assembly language you are using a text editor and can enter LDA $FE. If you've told the assembler that address $FE is named COLOR; then LDA COLOR is possible. Assembly language also permits Labeling any statement; so, instead of 20 00 03 to activate a sound effect, you might enter JSR HONK. Named variables and labels make it a cinch, using a good editor, to add, delete, and move around pieces of your program; and, naturally, the product is vastly easier to read and debug. Most of today's assembly language packages are "macro assemblers". This means that you CAN use and/or create macros (i.e. procedures callable via a label); not that you must. Almost certainly, you will want to make some use of the macros found in Math, I/O and other Libraries.

From the very start, ORCA/M (sometimes, in its nearly identical "APW" form) has been the standard IIgs development environment. As a result, familiarity with and access to the Byte Works product is assumed in programming examples found in Apple manuals and valuable learning aids like **Exploring the Apple IIgs** ($22.95) and **Exploring Apple GS/OS and ProDOS 8** ($21.95, both by Gary Little, from Addison-Wesley). The same applies to on-disk programs and libraries available as supplements to the Little books and Apple's **Programmer's Introduction to the Apple IIgs** ($32.95, from Addison-Wesley). Naturally, ORCA/M's libraries of Equates and Macros 'fit' the Toolbox Reference volumes like a glove.

All ORCA language packages, including Pascal, C, and Assembly, 'operate under' the ORCA Shell. The Shell is an operating system which, besides allowing access to ProDOS commands, 'glues together' the Editor, various language assemblers, assorted utilities, and the Linker. This does NOT, incidentally, mean that you must start up the Shell to run programs created using an ORCA language. You CAN create files which execute only when the Shell is active as well as Shell batch files; you can also create stand-alone "GS Applications", CDA's, NDA's, etc., and, via the MAKEBIN utility, produce old style BRUNable files. The Shell lets you customize your working environment (e.g. name a "Work Disk", set a preferred printer init string, etc.) and, via the LOGIN file, have the system start with your preferred environment in place.

An advantage of becoming familiar with the ORCA/M assembler is that, in the process, you become familiar with most of the 'tricky part' in learning to use any language development system. In ORCA, only the particular language assemblers, Editor settings (e.g. tab stops, etc.), and Libraries change from language to language. Once you master the Shell, learning to program in any ORCA language is, chiefly, "just" a matter of learning the language. Currently, the Shell comes in two 'flavors': the all-text version, included when you buy ORCA/M, and the super-res windows-and-mouse "Desktop" version. The latter, supplied with current GS/OS, includes such advanced debugging features as single-step execution with variables monitoring and a graphics output window.

"Okay; so, how do you write an assembly language program?" Here's a typical example: 1. You boot the ORCA diskette. 2. Enter ASM816 to set the current language to Assembly Language. (Your LOGIN file could do this for you.) 3. Enter EDIT BUGNOSE to edit the program named BUGNOSE. If the BUGNOSE source file (a text file) is found, it is loaded; if not, it isn't. Either way, you end up in ORCA's full-featured text Editor with tabs and other defaults set for convenient Assembly Language entry. 4. You make any desired changes to BUGNOSE (or, if from scratch, enter your new BUGNOSE program), exit the Editor, and do a quick SAVE. 5. Now, entering ASML BUGNOSE will assemble and link your stuff and create an exe file which you can execute from the Shell by entering BUGNOSE. (NOT, of course, a good idea if BUGNOSE does something gross, like setting all RAM to FF's.) 6. If you'd like to have BUGNOSE in BRUNable form, you can enter MAKEBIN BUGNOSE.EXE BUGNOSE.BIN. See, easy!

So far, I've used ORCA/M to develop several utilities. While, in a few cases, the machine code entry route was a possible alternative, some projects (e.g. a super-res picture packer/unpacker called "Super Pac") would have been nearly unthinkable. ORCA/M now ships with the "ZapLink" version of the Linker; so, assembly-link time is better than ever— most 'small stuff' takes less than a minute. Assembler error checking, fortunately, is very good; and, thanks to the Shell 'glue', it's a snap to hop back into the Editor, correct an error, and be ready for another try. Two caveats: though the system can work with one drive and 512k, life is much easier with two drives; and, to run with the super-res Desktop system, plan on at least 1.25MB of RAM. ORCA/M's 370-page manual is comprehensive, well-indexed, and very readable; it is also not particularly well organized. Expect to end up with documentation plastered with your own index tabs.

If, at long last, you're ready to create your own windows, use mouse inputs in YOUR programs, etc.. If, in short, you want the power (and fun!) of writing your own IIgs software, then get out your Toolbox References, check out the books mentioned above, and, very definitely, check out ORCA/M!

Reach for the Stars: The Conquest of the Galaxy, Third Edition

★★★

$44.95 for 512K Apple IIgs
Strategic Studies Group

From the developers of such wargame favorites as "Panzer Battles" and "'American Civil War'", **Reach for the Stars** III moves you into the distant future where battles decide the fate of planets and, eventually, of the galaxy. This much-reworked version of the first Keating and Trout product offers 1-4 players a shot at the original scenario, a new Advanced Scenario, and (thanks to user-adjustable production costs, movement factors, victory conditions, etc.) countless variations of each. To facilitate one-on-one and solitaire play, any unfilled slots are taken by computer players. These can range from inept bunglers (i.e. "jelly rolls") to Enhanced Veterans capable of putting together formidable alliances (i.e. "guns").

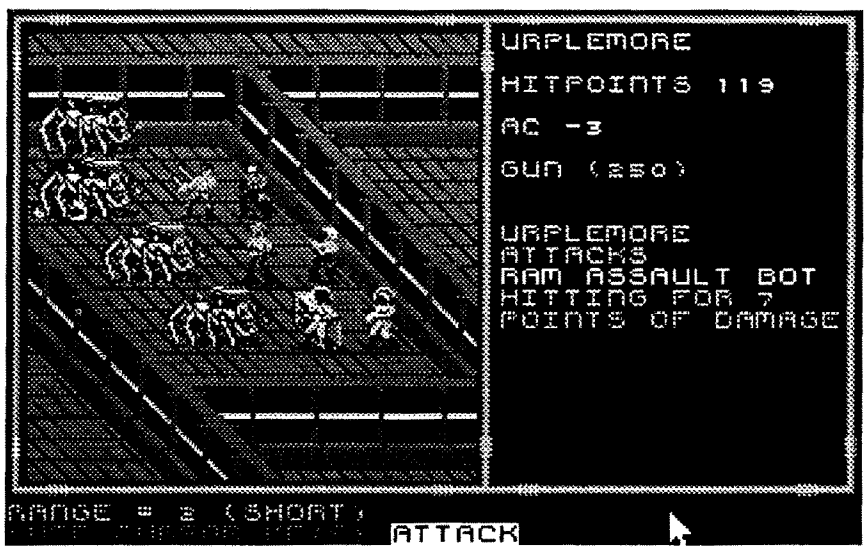
Anticipating that many spacewar jelly rolls will be attracted to the game, RFTS designers include a well-documented Tutorial Scenario. Here you get hands-on practice with setting planetary production targets to build ships, develop drive technology, enhance industrial capacity, ... You also explore prospects for expansion and manage your military assets for defense and conquest. Like any reasonably complex piece of entertainment software—the better flight/combat simulators are a good example—RFTS takes some serious 'getting into'. Pull-down menu/stat displays, quick save/load, and point-and-click mouse controls relieve players of most record-keeping and 'bean counting' chores. Still, you must enjoy the kind of planning and attention to detail that produce battlefield victories and strategic gains. If, in short, you are for-real wargamer material, then RFTS is waiting with crisp, colorful super-res star maps and a strategy wargaming platform good for many hours of absorbing play.

Buck Rogers: Countdown to Doomsday

★★★

$49.95 for CGA-VGA 640K PC
Strategic Simulations
(Clue Book, $12.95)

Hard upon a string of swords & sorcery victories, the SSI-TSR team has teleported their D&D adventuring model into the 25th century, and not a moment too soon. Led by Buck Rogers, New Earth Organization rebels have just driven oppressive Russo-American Mercantile (RAM) forces from the planet; and everyone knows RAM will be back. It's 'No more Mr. Niceguy', even if it means total war and a Countdown to Doomsday!



Beginning as new NEO recruits, your party of six soon discovers that RAM is building a Super Laser to sterilize the cities of Earth. Closing down this project is your primary mission, one which takes you to an asteroid outpost plus several bases and villages on Mars, Venus, and Mercury. Don't expect to encounter many evil wizards; do expect tough meetings with RAM guards, pirates, desert apes, (hyper-snakes, etc.) and assorted 'special purpose' robots. Your party will also face automated security defenses, ship-to-ship combats, and dangerous boarding clashes. Occasionally, a friendly Desert Runner, reformed computer personality, ..., or even Buck himself will pitch in; mainly, though, saving Earth is up to you.

After four long-play SSI swords & sorcery quests, 'sci-fi D&D' definitely takes a bit of acclimating. While Magic is gone, your characters can acquire new "Skills" and a few weapons which approach the gratifying destructiveness of a Fireball. Tactical combat looks and works much like that in the S&S games; which is to say it works very well, offering quick, flexible click-and-point control. Still, while you can look forward to more varied tactical setups (e.g. more barriers, rooms to hole-up in, etc.), modern attack and defense options clearly lack the range supplied via magic. By "Buck Rogers II'" ("Revenge of RAM"?), it will be time to uncover heretofore unimagined Founder artifacts, psi powers, and Vulcan death grips.

Boasting 3D-perspective forward-view VGA 'exploration mode' displays, AdLib music, and decent effects, **Buck Rogers'** is the most attractive, smoothest running implementation yet of SSI's popular questing format. Look forward to a reasonably flexible, clue-packed scenario, stiff combats, and 30-50 hours of entertaining sci-fi adventure. (Includes volume 1 of the Buck Rogers Inner Planets Trilogy: First Power Play, a TSR paperback by John Miller.)

Crystal Quest

★★

$49.95 for 512K Apple IIgs
Casady & Green

When "Mocking Board" first delivered REAL sound to the II+/IIe market I had the opportunity to try (and try, and ...) a simple, but clever demo arcade named "Thunder bombs". WHY?, I kept wondering, was I playing this twitsy game? The answer was sound!— specifically, a very rewarding SFPLURT!!! synced to a churning 'fireball' whenever I popped an enemy ship. Well, **Crystal Quest** offers a much superior move-around-and-zap-space-monsters challenge PLUS highly effective IIgs sound effects.

In each of an (apparently) endless series of marauding monster "Waves", the object is to mouse-guide your icon-sized craft around the screen and collect crystals. When you get them all, a Gate at the bottom of the screen opens and you can zip through to face the next Wave. Hampering your efforts are mines sprinkled amidst the crystals and the need to adjust for momentum effects in ship movement and missile launches.

Your Big problem is the monsters which periodically enter through portals on either side of the screen. Dumples squirm toward your ship and grunt, Pests lay mines, Zarklephasers spin-off missiles, Menaces slide along the borders and fire laser bursts, etc.. Popping these turkeys is very satisfying and makes it easier to collect crystals and grom valuable Smart Bombs (one of these wipes out all monsters currently on the screen). Potting monsters also ups your score; but the major boosts come form collecting special Super Crystals (which drift onto the screen and move around) and getting through a level in one piece. To encourage competition, CC starts a 'Today's Top Scores' listing each session and maintains a large all-time High Scores roster on-diskette.

Whether or not to risk hanging around after a Gate is opened (to get all the Smart Bombs or wait for a Super Crystal) is up to you. Whether or not to quit play after a few waves probably isn't. Crystal Quest is guaranteed to grab any arcade devotee for hours of action-filled monster-sfplurting.

Fast Frames, Updates, Etc.

Stick-y Stuff

OMEGA

☹

$29.95 Joystick for Apple II series and IBM

Beeshu

Size: 4.25"W x 5.0"D x 1.75"H x 3" Handle
Centering Tension: approx. 70 gm.
Handedness: none
Centering Error: 2.8%H, 1.6%V
Centering Defeat: none
Movement Range (Horiz.): 70 degrees
Useful Range (X/ Horiz.): 33 degrees
Centering Adjust: two thumbwheels
Connector: Apple DIN & IBM adapter
Case Access: four screws, remove feet

Available in lime green and three alternative da-glow colors plus off-white, Beeshu's **Omega** boasts rounded contours and a smooth fire-button-topped lotus handle. PB1 (the second switch) is duplicated, left and right, on the top front of the up-sloping case.

This is an attractive 'gumdrop', compact and light enough for hand-held or on-desk operation, IF, that is, it operated! The killer spec is "Useful Range", the handle swing in degrees over which X output (Y for vertical swing) actually changes. **Omega** is an analog device; but with response end-points only 16.5 degrees from center, the stick acts more like a four-way switch. Occasionally, this won't matter— as when a product cares only which direction you're pointing. Too often, however, it will matter very much. All sorts of driving and flight/combat simulations demand close, full-range control for best performance. Taking on one of these with **Omega** is like playing with half your skill tied behind your back.

Where are they now?

As noted in past joystick reviews, the usual test procedure includes a week or so of 'living with' a stick. This turns out to be fairly revealing when it comes to evaluating user comfort and game performance, but doesn't say much about longer term concerns. How does a stick stand up to continued daily use? Do comfort and performance evaluations 'prove out' over the long haul?

For starters, several sticks were picked-out by visitors, donated to PBS tele-auctions, etc. soon after the reviews. These include Wico's second **'Commander** (gray case, rod-like handle), Suncom's **Tac 1+**, Control Marketing's **Pro 6000**, and Kraft's **Premium III** (original, black-handle + red button version). All missed the long-term trial.

Among the units destined to 'stay and take it', the CH Products **Flight Stick** retains its excellent 'command console' feel and a solid hold on its position next to the IIgs. The venerable **Mach III**, another CH stick, continues to be a smooth, reliable performer after many, many sessions of "Elite", "Conan", etc. on the II+. Suncom's unique **Starfighter** gets only an occasional workout. Thanks to pressure-transducer technology and bolt-anchored handle, the **'fighter** is perfect for high-energy visitors used to slamming around cartridge-game switcher sticks. (It's a good bet that, around 2500 A.D., some archaeologist will dig up a **'fighter** and write a paper on the "indestructibility of 20th century devices". He or she will also, no doubt, comment upon the incredible hand strength of earlier humans.)

An Epyx **500 XJ** has yet to reach me in ready-to-work form. Both of the repaired units continue to perform well enough on Apple applications which require just moderately precise centering and which locate 'center' near the middle of X and Y pot ranges. Being a hand-held-only unit and, at that, one incorporating interesting innovations that 'kind of work' (e.g. push-button centering), the **'XJ** is, at best, a 'second stick' choice for the experimentally inclined user.

Evil times have befallen one highly rated unit. After a few months of fine action-game sticking, the **Advanced Gravis Analog Joystick** (for Apple) developed a serious up-glitch in the vertical pot. I removed the defective resistance element— just un-solder the leads, pull out the wiper assembly, and pop out the element— and, sure enough, the resistance track showed a worn spot. More to the point, the resistance layer proved to be so thin as to virtually guarantee a short life. A call to the manufacturer seemed to confirm this analysis but produced some good news as well. According to Ron Haidenger, Gravis product manager, the company switched pot suppliers several months ago (i.e. shortly after sending out the review sample). If you own a Gravis stick purchased after reading the review, there's a good chance your pots are of the better quality variety.

WHAT are they now?!

In the highly favorable review of Suncom's **Analog Plus**, quite a bit was made of how nice it was to see pressure-transducer technology "come of age". Indeed, WHEN it happens, it will be nice to see. The **'Plus** is a very good stick loaded with design innovations; but it does NOT employ pressure transducers. (Well, it Looks, Feels, and Responds like a PT stick; AND it's from Suncom, maker of "Starfighter".) As fate would have it, **Analog Plus** is the only stick I did not completely disassemble and examine for the reviews. Recently, I opened the case to scrape off a bit of plastic so the centering adjustments would slide more easily. The white circuit board flopped out and the Truth was revealed!

Instead of the expected pressure sensor pads activated by rubber-tipped arms extending from the handle, the **'Plus** employs straight-line board-molded resistance tracks. Two wipers are mounted at right angles on separate slotted plates. (One has a horizontal slot, the other has a vertical slot. The bottom tip of the handle fits through both slots.) Moving the handle slides each wiper along its track to produce X and Y resistance outputs. (Clever!) The Analog Plus doesn't merely "employ pots"; it IS a highly rugged, dual-output potentiometer. It also remains a popular, proven performer on "Stormovik", "Stellar 7", and many other PC simulations. (Hint: For better, more positive feeling triggers, open the handle and carefully remove the two springs.)

My chief **'Plus** complaint, that centering tension may be too high for prolonged high-action applications, has proven out. On Origin's "Wing Commander", fifteen minutes or so of fierce space combat is likely to force most players to a two-handed grip. One solution is to switch sticks; another is to view this as an opportunity to build wrist strength. More daring users (with access to a pointy-tipped soldering iron) can disassemble the case, slide out the resistance assembly, and 'adjust' tension by punching spots around the flexible ring which anchors the handle. (Remember, a decorative rubber boot surrounds the handle on top. Take care to punch only the ring.) Since the sliding plates are centered via springs on the resistance assembly, a circle of 8-12 small holes will reduce handle tension without affecting X-Y centering accuracy.

Lo-Mem Boots

As mentioned last issue, many of the newer PC games demand a hefty chunk of your 640K base memory. Some examples are "Stunt Driver", "Ishido", "Stratego", "Elvira", Stormovik", "Altered Destiny", "Prince of Persia", "Savage Empire", ... (i.e. not ALL of the 'good stuff', but you get the idea). Starting one of these from the usual Dosshell selector will result in mid-play bomb-out or, if you're lucky, generate an immediate "Not enough free memory" message.

One cure is to routinely rely upon memory manager software to move chunks of startup applications out of precious 'base RAM' into either expanded or extended memory. Also, DOS supports moving some buffers and drivers into expanded memory. The 'catches' here are (1) The product to be run may, itself, want to use all or most of any expanded memory; and (2) Your favorite DOS and/or any available memory manager software may not recognize the presence of your particular brand of expanded memory board. (Fortunately, most products which make use of expanded memory don't care whether DOS knows about your board or not.)

The alternative 'cure' is a good deal simpler and more straightforward. Vendors know when their stuff is 'pushing the limits'. Typically, you will find some suggestion to the effect that if (?!) you experience loading problems, try booting with a bare bones DOS. Experienced PC types see this suggestion and don't even bother trying a 'standard' boot. They pull out a lo-mem boot diskette and run the game with no problem.

What is a lo-mem boot ("LMB") diskette? To create one, you stick a diskette into drive A and enter "format a: /s". This formats the diskette and writes the three system files DOS needs to perform essential functions. With current DOS versions, you can expect about 590K of free memory after booting such a diskette (DOS, etc. is using only about 50K).

Since you will usually want a mouse driver, you should create an autoexec.bat file to load one when the LMB diskette boots. (Look at your 'standard boot' Autoexec.bat file to see how it loads the mouse driver.) For example, to startup the Microsoft mouse:

```
ECHO ON
C:
CD \
MOUSE
A:
ECHO OFF
ECHO Welcome to Lo-mem Boot.
```

Here, the mouse driver, MOUSE.COM, is in the drive C (hard disk) root directory. Once the newly created autoexec.bat is written to your LMB diskette, booting it should install your mouse and leave roughly 580K of free memory. This is enough to run just about any current entertainment product with all of the 'bells and whistles' the program offers.

There are a few notable exceptions. Given the above setup, games like Origin's "Wing Commander" and "'Savage Empire" will run; but, for full sound and graphics, you will also need a few hundred K of expanded memory. On a '286 PC/AT, this means you plug in an "Expansion Memory" board. (With 512K installed, typical prices start around $150.) To install a board's Expanded Memory Driver (included on a diskette when you buy the board), you add a config.sys file to your LMB diskette. The following config.sys one-liner activates an Everex "RAM 3000 Deluxe" board:

```
DEVICE=C:\EMM.SYS /C C800
258,00
```

Booting an LMB diskette with this config.sys and the autoexec.bat mouse installer will start DOS, 'hook up' expanded memory, and load your mouse driver. Since the ems driver uses some base RAM, 'available memory' may drop to 569K or so. Applications which employ expanded memory (e.g. "'Savage Empire") allow for this. They demand less base RAM than some other large applications (e.g. "Elvira") which use no ems memory.

Soon, perhaps as early as spring, we should have a DOS which can handle ALL available memory. Once applications designers can easily 'get to' the EXTENDed memory most AT users already have onboard, LMB diskettes and/or tricky memory management utilities will be for "old DOS"-stuff-only. For now, make a couple LMB diskettes. Add the mouse installer (autoexec.bat) to each; add the ems installer (config.sys) to the one you'll boot to 'turn-on' expanded memory.

Note: If you are a new or soon-to-be new PC user and much of the above vaguely resembles Greek (and you're not Greek); then check out the review of Microsoft's "Learning DOS".

Indianapolis 500: The Simulation
★★★

Picking the 1990 'Sleeper of the Year' was easy. Seldom has such a fine product received so little attention as **Indianapolis 500'** ($49.95 from Electronic Arts). Setup options include the expected choices of team & car (March Cosworth, Lola Buick, Penske Chevrolet), action format (practice, qualifying, race) and race length (10-200 laps). After a few hours practice, you will want to check the manual and fine-tune car settings for Tires (rubber mix, pressure, and size), Shocks, Camber, Fuel, airfoil Wings, and Gear ratios. To give you a taste of how different settings affect handling and speed, three loadable settings arrangements are supplied on-diskette. Your own favorite settings can also be saved and loaded.

Nice; but the game's best up-front feature is that, just seconds after booting, you can be in a car and racing with barely a glance at the manual or 'setup card'. Joystick control is smooth, offering steady command of wheel position plus natural feeling (fwd/back) throttle and breaking positions. Zooming around the "Brickyard" your cockpit view shows crowded stands, pitstop lanes, flagmen, ... and the track-burning competition— all in colorful, expertly animated 3d-perspective with solid AdLib sound. Alternative views include TV-type sky and track positions with option for instant replay (e.g. of those spectacular crackups). So far, **Indy' 500'** has been good for hours of first rate racing— and that was BEFORE I decided to look at the manual and 'get serious'! (For CGA-VGA 640K PC)

Tangled Tales/PC
★★★★

Yet once again there is cause for joy in PC-ville. Last year Origin's whimsical three-part adventure earned four stars as a II+/IIe release. In 16-color EGA the partially-animated characters, scenery, maps, and dungeons look even better. As in the II version, sound is zilch; but TT's attraction remains. Expect an engaging, long-play scenario packed with interesting personages, places, and challenges. ($29.95 for CGA-EGA 384K PC)

Wing Commander
★

Supposedly, if someone does high-action cockpit-view space combat in fast, smoothly-animated, VGA with stirring music and decent sound (both via AdLib), the result is a sure winner. If you add a blow-em-away intro, insert movie-quality between-mission scenes, and finish with thorough, attractive documentation there SHOULD be absolutely no doubt. Why, then, doesn't Origin's ambitious new arcade quite 'deliver the goods'?

Call it "Sock", "Punch", "Pow"; whatever term you use for the moments when you fire off a shot (SHHVROOM!!, ...) and FEEL it take a bite out of the target— THAT's what **Wing Commander** is missing. A Kilrathi (the bad guys) Jalthi fighter swoops into point-blank range, you unload a blast, and ... nothing! "That's because they have shields!" Uh, uh. Shield hits are supposed to flare and make a FRZZZZAP!! sound (or SZFOOFF!!), or something. And, at WC's level of detail, at least some real hits should munch a piece out of the haul. Everyone knows that space fighters don't automatically blow on the first hit. The 'bottom line' is realism. The closer a game gets to it, the more easily any 'obvious' omissions can render the whole thing UN-real. Flying a **Wing Commander** mission can be moderately entertaining; but, without 'the Punch', it's too much like a pillow fight. ($69.95 for EGA-VGA 640K PC + 300K-500K ems)

The Passion
★★

Accolade's third **Test Drive** ($59.95 for EGA-VGA 640K PC) matches you against a pair of computer competitors in a LONG-running California road duel. 'Choice of weapons' includes Chevrolet's CERV III prototype (a real monster), Pininfarina's super-responsive Mythos, and Lamborghini's road-hugging Diablo— you are not, in short, exactly helpless! On the other hand, the computer guys don't have to handle tricky turns with KB or 'switcher' stick, you do.

"Whoa! My joystick says 'Analog' right on the case!" Maybe so; but, in TD-III/PC, ANY deviation from 'center' is treated like a KB keypress— that smooth-responding analog wonder might just as well be an Atari four-banger. You CAN learn to drive this way; but it's never likely to feel 'natural', and switch-steering most certainly damps the 'road feel' characteristics of each car. Add just-adequate AdLib sound effects and the surprise is that cruising the five-section course (Scenic Coast, Coast Hills, Valley Farms, ...) turns out to be a very enjoyable, low-pressure challenge. Just turn on the radio (you get a choice of Rock, Country, or Classical music) and drive. Day becomes night, it may rain, cows, slow-moving trucks, etc. may lumber onto the road, the highway patrol WILL be watching, and you may or may not place in the Top Driver's roster (maintained for each section and the course). So what; this is laid-back "racing" in the California style. You've got plenty of 'lives', five maps showing alternate routes, and miles of finely-detailed scenery to explore.

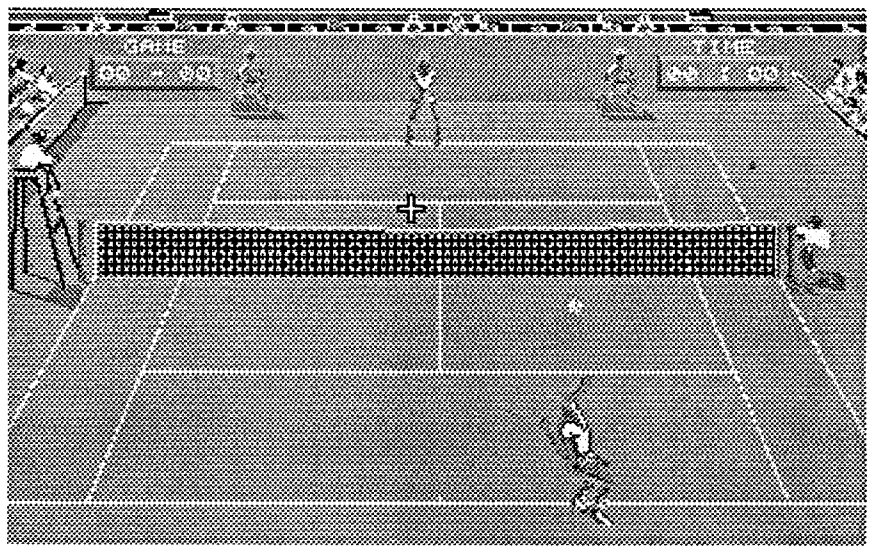
The Biota Company

A new IIgs educational software maker, Biota, has sent in a few beta samples of

learning materials produced using photography-to-super-res techniques. The approach, demonstrated via a nature learning unit on "Bogs", looks promising. J.D. Curnow, the chief developer, did not say how close Biota is to some finished products— by now they may have several. If Education is 'your game', you might send J.D. a note and ask.

Pro Tennis Tour
★★★

When you recall that the first video game was called "Pong", it has to be at least a little ironic that tennis is the last popular sport to be successfully 'computer gamed'. Blue Byte's **Pro Tennis Tour** (from UBI Soft/Electronic Arts) is the first simulation to deliver both the look and feel of 'real tennis'. The "look" is a from-the-stands full-court view from behind the player figure you control. To get the "feel", **Pro Tennis'** combines just the right amount of player joystick input with plenty of realistic computer fill-in. On serves, you 'fire' to toss, swing a cross-hair aimer to your target area as the ball arches up, and fire to hit. On volleys, the cross-hair indicates where your opponent's shots will land. Your main task is getting into position and timing your return button press. Distance from the ball, side, and timing all affect shot placement.



Pro Tennis' includes a mode for practicing serves and returns as well as full-game workouts against a computer player. In Tournament mode, you compete in matches for a ranking among 64 computer players. The game also includes a second-human-player option; but, since only one player is allowed a joystick, you can write it off along with single-player KB and mouse options.

Supplied with manual and commands card, **Pro Tennis Tour** makes good use of 'old-PC' sound and smooth animation to guaranteed many hours of easy-to-get-into, addictive court action. ($39.95, for CGA-EGA 512k PC)

Super Quest Champs?

Could be. In a recent letter, Brian Gillogly notes that he and his brother, Bob, not only reached the Mega Crown (thus restoring the balance of Science and Technology in the 23rd century), they have gone on to map every square of the monster-packed maze! Since **Super Quest** (a 1983 SoftSide release for 48K II+/IIe) spans well over 1000 'rooms' and game saves are limited to a few special locations AND there is no character resurrection option— well, mapping the entire thing is an astonishing achievement. Brian also asks about something which may have teased other SQ players: What's in the two 'unreachable rooms' (#450 and #451)? Good question. Perhaps they will play some role in **Super Quest II**.

Bomblets, Not Bombs

Due to a typo error (mine), a few products last issue ended up with ratings of **Defective** (💣) instead of the intended **Poor** (☹). Just cross-out the bombs and draw-in unhappy faces.

While I'm on the subject; and, since it has been a while since Ratings were last described, some explanation may be in order. To interpret a rating, just imagine that you've finished trying out the product and exclaim "Excellent!". **THAT is a four-star rating.** "Unbelievable!", "Awesome!!!", etc. stuff which breaks new ground in some area (e.g. graphics, sound, speed, etc.) may reach five stars. But, if you find yourself telling a friend "Oh, it was okay", the product probably rates just a one-star Fair.

When an otherwise attractive product doesn't work due to some fixable defect, it may be tagged **Defective**. Since the idea is to warn readers to "watch out, for now", usually, only highly-advertised, high-interest products are candidates. Nowadays, **Defective** ratings are very rare. More often than not, I'm able to reach the vendor via telephone, obtain needed explanations and/or assurances, and pass these along. Any review then goes 'on the back burner' in the expectation that the vendor will fix the problem and send me an update.

On some imaginary Scale of Excellence, the biggest gap is between **Fair** (★) and **Poor**. Were a friend to ask if he or she should purchase a **Fair** product; you might say "Maybe, if ... ". You would never recommend a **Poorly** performing game, utility, or piece of hardware. This explains why so few **Very Poor**'s turn up. **Poor** is bad enough!

Conversely, **Good** (★★) is, frequently, good enough. Depending upon such highly subjective criteria as a game's setting and scenario, the way a painter utility arranges its tools, etc., a two-star product could easily rank among your favorites. Of course, if you like what a review says about a **Very Good** (★★★) game, language package, plug-in board, or whatever, your chances of getting 'the right stuff' are even better. Basically, the ratings say what they mean.

Getting the Red Out

Granted, a software vendor has the right to protect its products any way it sees fit; and, when it employs documentation (instead of screwing-up the software), it deserves a pat on the nose. The dark red code sheets technique employed by some vendors comes close to being an exception. It's nice that they are machine-uncopiable (though, naturally, anyone determined to steal the product need only invest an hour or so to make a copiable hand copy). It's too bad that, under normal lighting, the codes are nearly invisible.

After missing the correct response twice on a newly arrived game employing tiny print on such a red background, I was carping about "protection that spoils player enjoyment ...", when Gorbash turned from a nearby computer and suggested: "Maybe you could bleach it out." Well, when an M.I.T. chemistry Doctor of Science says "try bleach", you try bleach. Viola! A dip in a tray of Purex plus some drying/flattening between paper towels in an old magazine and the job is done. (Warning: bleach is bad stuff; so, be careful.) The red is gone and blue print stands out clearly on a light-yellow background. Magic!

Quick! Before She Melts

Okay, the ghoul gleeclub crowding into Elvira's castle may not melt, massacre, or otherwise dispose of her right away; but, it's a cinch property values are dropping like a rock. If YOUR creep cleanout chores are behind schedule, then maybe these hints (gleaned from play, NOT from the "Clue Book") will help get Things moving:

•Your best bargain among the available spell recipes is "Spagetty Confusion". Mine was good for many, MANY valuable Tangled Mind castings.

•The surest way to get through the Hedge Maze is to explore the whole thing. The idea is to zap all of the gremlins BEFORE you get to their nest. Do frequent Game Saves.

•The best place to find bodies in the moat is in the middle channel. (Incidentally, it takes twenty 'steps' to go around the castle.)

•Finally, here's a little attributes booster suggested by Baywoof: Load your saved game into "Xtree'" or similar editing utility; hop to the latter part of the file (around byte $5110); locate your Strength, Dexterity, and Resistance attribute numbers; change them to $63's (99 decimal). Now, just save your enhanced character, swagger into the castle, and clean their clocks! (*Note:* I tried this and also upped my hitpoints as well. For best results, do a "Wooden Heart" heal after making the changes and restarting the game.)

Next

Try not to empty your post-Christmas piggy bank on one spree. Incredibly, there is MORE to come! For example ...

Altered Destiny (Accolade): Trading TV sets with a barbarian super-hero can get you into some very strange places.

Stormovik (Electronic Arts): Armed to the teeth and hugging the ground at 900-plus kMh, you and your SU-25 attack fighter are just the medicine to cure a terrorist uprising in Eastern Europe.

Star Saga One (Master Play): Questing 'beyond the boundary' puts you IN and in-control-of your own sci-fi adventure.

Logo Plus (Terrapin): ProDOS comes to the realm of the turtle.

Ishido (Accolade): Elegance, in stone!

Wizardry: Bane of the Cosmic Forge (Sir-Tech): Not Wizardry VI; a whole new Wizardry in a new VGA + AdLib format!

Vendors

Accolade: 550 S. Winchester Blvd., Suite 200, San Jose, CA 95128
atten: Meli da Mongelluzzo (408-985-1700)

Ad Lib: 220 Grand-Allee East, Suite 960, Quebec, QC, Canada G1R 2J1
atten: Jill Carette (800-463-2686)

Addison-Wesley Publishing: Route 128, Reading, MA 01867
atten: Abigail Genuth (617-944-3700)

Advanced Gravis Computer Technology: 7033 Antrim Ave., Burnaby, BC, Canada, V5J 4M5
atten: Ron Haidenger (604-434-7274, in USA call 800-663-8558)

Beeshu: 101 Wilton Avenue, Middlesex, NJ 08846 (201-968-6868)

Biota: 551 Cedar St., Richland Center, WI 53581
atten: J.D. Curnow

Broderbund: 17 Paul Drive, San Rafael, CA 94903
atten: Jessica Switzer (415-492-3200)

Byte Works: 4700 Irving Blvd. NW, Suite 207, Albuquerque, NM 87114
atten. Patty Westerfield (505-898-8183)

Casady & Green: 26080 Carmel Rancho Blvd., Suite 202, Carmel, CA 93923 (800-359-4920)

Center for Gifted and Talented: University of Houston/University Park, Farrish Hall #123, Houston, TX 77004
atten: Theresa Monaco

CH Products: 970 Park Center Drive, Vista, CA 92083
atten: Dan Hayes (619-598-2518)

Electronic Arts: 1820 Gateway Drive, San Mateo, CA 94404
atten: Lisa Higgins (415-571-7171/ orders: 800-245-4525))

Epyx: 600 Allerton St., P.O. Box 8020, Redwood City, CA 94063
atten: William Lanphear IV (415-368-3200)

Janklow Bender: 257 Park Avenue South, New York, NY 10010
atten: Kim Adamo

Lucasfilm Games: P.O. Box 10307, San Rafael, CA 94912
atten: Betsy Irion (800-STARWARS)
dist: Electronic Arts

Microsoft: 1 Microsoft Way, Redmond, WA 98052
atten: Marilyn McKenna (800-541-1261)

Northgate Computer Systems: P.O. Box 59080, Minneapolis, MN 55459-0080
atten: Mona Hendrickson (800-526-2446)

Origin Systems: 110 Wild Basin Road, Suite 330, Austin, TX 78746
atten: Greg Malone (512-328-0282)

SQ Players: Brian and Bob Gillogly, 1073 CR 37, Fremont, OH 43421-9651

Strategic Simulations Inc.: 675 Almandor Ave, Sunnyvale, CA 94086
atten: Linda Blanchard (408-737-6800)
dist: Electronic Arts

Strategic Studies Group: 1747 Orleans Ct., Walnut Creek, CA 94598

Suncom: 290 Palatine Road, Wheeling, IL 60090
(312-459-8000)

UBI Soft: 511 Sir Francis Drake Blvd., Suite C, Greenbrae, CA 94904
atten: Mkt. Director dist: Electronic Arts

Walt Disney Computer Software: 500 South Buena Vista, Burbank, CA 91521
atten: Kirk Green (818-567-3340) re. Janklow Bender

James A. Hodge — CT

Speeding up Rocket Ranger

It's time to take Rocket Ranger (by Cinemaware) off of the Most Wanted list. It's NOT COPY PROTECTED. A copy of a copy of a copy of the original plays the same game as the original. The only bug I found occurs after beating up the guard when you're rescuing Jane and Otto (Rocket Ranger "loses" his head). The aspects of the game that need attention are the lack of speed and a hard-to-read "Secret Decoder Wheel".

The game can gain speed from two sources. If you have 1.5 megabytes of memory you can set up an 800K ramdisk and copy disk 2 (/Rocket.Ranger2) into memory. This speeds the frequent disk accesses, saves the disk drive, and makes the game more enjoyable to play. The second source of speed would be a TransWarp GS from Applied Engineering. The TransWarp can reduce game time by half; time drops from the 2-2.5 hour range to about 1.25 hours, and Rocket Ranger becomes very responsive to the joystick.

The only drawback to a game played at transwarp speed is that the Nazi guards are VERY difficult to beat by 1944. The solution is to slow the TransWarp from "transwarp" to "normal" for fist fights that occur later in the game, and also if/when you get to the Moon. To enable access to the control panel requires an edit to the disk / Rocket.Ranger1. Perform the following edit:

Blk	Byte	From	To
$466	$1C	22 00 00 E1	68 68 68 18

(Author's Note: There should be a companion article in this issue of Computist about control panel lockout. See it for more detail.)

Advanced Playing Technique for...

Rocket Ranger

Cinemaware

The decoder wheel can be difficult to read. The numbers are small, and they are viewed through a piece of red cellophane. If you make a mistake selecting the amount of fuel, you (Rocket Ranger) either end up somewhere you don't want to be, or dead (which is also where you don't want to be).

The decoder wheel problem can take a little (or a LOT) more work. A simple solution is to print a copy of the decoder wheel in chart form. Such a chart was printed in Computist #71 on page 22, but it was poorly formatted and contained errors. A more difficult, but more technically satisfying, solution is to implement the decoder as a CDA. Read on, and choose the method that appeals to you.

To begin with, I printed my own chart using the following Applesoft BASIC program. I felt that I would need to enter less than half the numbers on the decoder to be able to produce a chart. The decoder had 27 * 27 (or 729) values, but 27 of them were zero (from point A to point A) and the values for point A to point B were the same as the values for point B to point A. It turned out that only 13 * 27 (351) numbers were needed. A small sample of the program output follows.

Sample printout (partial)

	A	A	A	B	C
	L	R	T	R	A
	G	A	L	A	N
	E	B	A	Z	A
	R	I	N	I	D
	I	A	T	L	A
	A	.	I	.	.
	.	.	C	.	.
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.
ALGERIA	0	28	20	43	
ARABIA	28	0	41	57	
ATLANTIC	20	41	0	32	
BRAZIL	43	57	32	0	
CANADA					

This is an (obviously) incomplete example of the output from the Rocket Ranger Decoder program, but I included it to show the format of the chart. It is 92 columns wide. To print it requires the ability to print in a condensed format (12 char/inch recommended) on an 8.5 inch page or a wide carriage printer with 15 inch paper (12 cpi still recommended). Once you get the program (by entering it, downloading it from the Computist BBS, or by getting the disk for this issue) you can fiddle with the format 'till it suits you. Printer initialization strings are printed in lines 85 and 95. The format I finally settled on was to set the printer for 132 columns, 12 char/inch (horizontal), and 8 lines/inch (vertical) in line 85, and change to 6 lines/inch in line 95. I had to include top-of-form commands for my printer. Consult your printer manual for the printer control commands you'll need.

Rocket Ranger Decoder

```
10 REM rocket.ranger decoder
20 DIM CNT$(27),AMT%(27,27)
30 FOR I = 1 TO 27: READ CNT$(I):
   NEXT
40 FOR I = 1 TO 26
50 FOR J = I + 1 TO 27
60 READ AMT%(I,J):AMT%(J,I) =
   AMT%(I,J)
70 NEXT
80 NEXT
85 REM ? CHR$ (4) "PR#1" :REMprint
   init string - pg. wdth 12 cpi,6
   lpi,tof
90 GOSUB 200: REM print heading
95 REM print init string - 6
   lpi,tof
100 FOR I = 1 TO 27
110 PRINT MID$ (CNT$(I) + "...
   ...." ,1,13);
115 T = 14
120 FOR J = 1 TO 27
130 POKE 36,T: PRINT AMT%(I,J);
140 T = T + 3: IF J / 3 = INT (J /
   3) THEN T = T + 1
150 NEXT
160 PRINT : IF I / 3 = INT (I /
   THEN PRINT
170 NEXT
180 PRINT CHR$ (4) "PR#0"
190 END
200 REM heading
210 FOR I = 1 TO 11
220 FOR J = 1 TO 27
230 POKE 36,(J - 1) * 3 + 15 + A:
   PRINT MID$ (CNT$(J) + "......
   ..." ,I,1);
235 IF J / 3 = INT (J / 3) THEN A
   = A + 1
240 NEXT
245 A = 0: PRINT
250 NEXT
260 PRINT
270 RETURN
1000 DATA ALGERIA,ARABIA,
   ATLANTIC,BRAZIL,CANADA
1010 DATA COLUMBIA,CONGO,E.◊
   AFRICA,EGYPT,ENGLAND
1020 DATA FRANCE,GERMANY,ITALY,
   KENYA,LIBYA
1030 DATA MIDEAST,NIGERIA,PERSIA,
   PERU,SCANDINAVIA
1040 DATA SPAIN,SUDAN,U.S.A.,U.S
   .S.R.,VENEZUELA
1050 DATA W.◊AFRICA,YUGOSLAVIA
1101 DATA 28,20,43,37,44,25,29,
   21,16,14,17,15,31,10,19,23,30,
   50,22,13,24,36,26,40,18,12
1102 DATA 41,57,59,65,30,18,14,
   34,29,27,24,26,20,17,31,12,
   70,32,33,19,58,23,61,37,22
1103 DATA 32,27,34,37,39,33,18,
   19,26,22,43,29,31,35,42,44,
   25,17,36,23,38,30,24,28
1104 DATA 49,27,38,48,52,50,45,
   53,47,46,42,54,37,61,25,55,
   40,44,41,60,28,33,51
1105 DATA 28,56,62,51,31,36,38,
   39,64,44,47,48,55,40,34,35,
   57,10,41,26,42,43
1106 DATA 52,64,59,47,46,50,48,
   62,49,57,45,66,17,51,43,61,
   26,58,12,39,53
1107 DATA 22,23,36,33,34,32,17,
   24,26,15,31,54,40,29,20,55,
   39,50,21,27
1108 DATA 15,37,32,33,30,14,23,
   24,25,19,67,35,31,12,61,28,
   60,34,26
1109 DATA 27,24,20,19,25,12,11,
   18,17,63,28,26,13,50,22,54,
   29,16
1110 DATA 08,11,13,40,21,22,33,
   29,53,15,12,30,32,19,41,28,20
1111 DATA 12,10,37,17,20,30,28,
   52,16,09,27,35,21,42,25,15
1112 DATA 09,35,18,14,32,24,58,
   10,16,29,39,15,45,31,13
1113 DATA 33,14,12,29,25,57,17,
   11,26,38,18,44,27,08
1114 DATA 28,30,22,27,65,39,36,
   18,63,34,59,32,29
1115 DATA 13,19,22,56,27,15,16,
   43,25,46,26,11
1116 DATA 28,18,64,21,25,23,46,
   16,52,36,10
1117 DATA 34,49,36,27,17,47,40,
   43,16,24
1118 DATA 72,26,32,21,54,20,62,
   41,23
1119 DATA 60,51,62,33,66,20,43,61
```

```
1120 DATA 20,31,37,14,47,38,18
1121 DATA 28,34,24,39,23,19
1122 DATA 56,32,55,22,25
1123 DATA 45,24,40,42
1124 DATA 53,44,17
1125 DATA 35,49
1126 DATA 30
```

Checksums

```
  10-$BADD   200-$2BED  1107-$9EF9
  20-$1990   210-$C09B  1108-$0967
  30-$5AA7   220-$7C80  1109-$3CBB
  40-$3257   230-$87D7  1110-$005A
  50-$7636   235-$E427  1111-$851D
  60-$237E   240-$D9E3  1112-$6BD5
  70-$04BD   245-$4CA0  1113-$AA88
  80-$ABA7   250-$2AF4  1114-$AA54
  85-$8410   260-$DCBC  1115-$A308
  90-$5096   270-$2796  1116-$8BF1
  95-$8174  1000-$B10E  1117-$797D
 100-$4ABA  1010-$14E7  1118-$A463
 110-$2310  1020-$C277  1119-$0522
 115-$1402  1030-$3994  1120-$96C9
 120-$F966  1040-$F8F8  1121-$1A44
 130-$ABCB  1050-$C8DD  1122-$B753
 140-$C37B  1101-$4CC0  1123-$E427
 150-$551F  1102-$54F1  1124-$64A5
 160-$2F1A  1103-$6D5E  1125-$A3A7
 170-$20E7  1104-$179B  1126-$0394
 180-$5212  1105-$DDB4
 190-$R25D  1106-$CF07
```

Rocket Ranger Decoder CDA

So much for the low tech approach. The work I did up to this point was preparatory to creating a decoder CDA. I had to be sure I understood the numbers and that I could get into the control panel. I tried implementing a CDA in ORCA/Pascal by converting the BASIC decoder program, but it was too large (15K) to suit me. Also, the Pascal program wasn't a good implementation of the decoder wheel; it asked you to enter your country of origin and your destination and it returned only the value you needed to make that particular trip. It did prove that a CDA worked with Rocket Ranger.

The next step was to code the CDA in machine language, using the ORCA/M assembler. The first attempt used macros and produced 4K worth of code. I replaced the macros with my own subroutines and squeezed and eventually got the program into 2K. Now it works just like the decoder wheel; tell it where you're flying from, and it shows you how much lunarium is needed to reach any of the 26 destinations. This helps when you're trying to plan a trip.

Without getting into an extended lecture on machine language, the program works as follows; first, it set the BASIC I/O hooks using the TextReset toolcall, initializes some stuff, and prints the title. Then it prints the 27 countries in a 2-column format menu, with each country preceded by a number. It prints an input prompt and waits for you to select a country. After you enter a number in the range of 1 to 27 the program reuses the menu loop to convert the menu numbers to fuel quantities and highlight (inverse) the name of the country you selected. At this point, you can hit a key to select another country, or enter "0" to exit the CDA to the control panel.

Points of interest for the technically minded; the Toolbox Reference says an application should never use the TextReset toolcall, but, what do they know? It's the quickest way to get into an environment where I can control I/O, it works without any apparent problems, and the system restores the I/O hooks to whatever they were when you exit the control panel. I use the FWEntry toolcall (FirmWareEntry) to make use of the Tabv and GetIn routines in the monitor. Tabv forces the cursor to go where the values in $00/0024 (CH, or cursor horizontal) and $00/0025 (CV, or cursor vertical) tell it to go. It's used in my GotoXY routine. GetIn is used to get input from the user, along with a character count. In this case I'm getting a number and I use the Dec2Int toolcall to convert the input in the buffer to an internal (hex) format value. Int2Dec converts a hex number (INTernal format) to a decimal value string (e.g., sending Int2Dec the value $19 would return "25"). I use it to display the menu numbers (1-27). I saved space (50%) by storing the values for fuel quantities in hex (2 nibbles) instead of ASCII (2 bytes). I use the Int2Hex toolcall to convert a hex byte into 2 displayable ASCII characters (sending Int2Hex $19 would produce the 2 character string "19").

The source code for the Rocket Ranger Secret Decoder CDA is presented below in ORCA/M / APW format.

```
; Rocket Ranger Decoder CDA
; by James A. Hodge - 11/20/89

rrdcda   start

Tabv     equ   $FB5B   monitor routine
GetIn    equ   $FD6A   monitor routine
FWentry  equ   $2403   Misc. tools firmware call
Int2Hex  equ   $220B   internal fmt. int -> ASCII hex
Int2Dec  equ   $260B   internal fmt. int -> ASCII dec.
Dec2Int  equ   $280B   ASCII digits -> internal fmt. int
TextReset equ  $050C   sets BASIC I/O hooks
WriteChar equ  $180C   print a single character
WriteCString equ $200C print null terminated string
ReadChar equ   $220C   get single character

         dc    i1'cda_strt-cda_name'
cda_name dc    c'Rocket Ranger Decoder'
cda_strt dc    i4'rrd'
shutdown dc    i4'overnout'

; Rocket Ranger Decoder

rrd      anop
         phb             save data bank
         phk             transfer pgm bank
         plb             to data bank reg.
         ldx   #TextReset
         jsl   $E10000   set BASIC I/O hooks
         stz   menu_ans  init menu/answer switch
         pea   title|-16
         pea   title     push adr of title
         ldx   #WriteCString
         jsl   $E10000
; Put up the menu
menu     lda   #1
         sta   i         initialize loop index
         lda   #0
         sta   x         start at 0th column
         lda   #4
         sta   y         in the 4th row
         lda   #country  move relocatable addr.
         sta   cntry     to workspace
         lda   #^country
         sta   cntry+2
menuloop anop
         jsr   gotoxy
         lda   menu_ans
         bne   answers if flag <> 0 print amount
         lda   i         else, print country #s
         pha             number
         pea   dec_dig|-16  store # at dec_dig
         pea   dec_dig
         pea   $0002     length
         pea   $0000     unsigned
         ldx   #Int2Dec
         jsl   $E10000
         bra   menu1
answers  anop
         lda   i
         sta   dest
         lda   org1
         sta   org
         jsr   calc      set dec_dig = amount
menu1    anop
         pea   dec_dig|-16  country number / answer
         pea   dec_dig
         ldx   #WriteCString
         jsl   $E10000
         lda   menu_ans
         beq   printit   if flag = 0 don't inverse
         lda   i
         cmp   org1      set inverse if dest = org
         bne   printit
         pea   $008F     set INVERSE printing
         ldx   #WriteChar
         jsl   $E10000
printit  lda   cntry+2   print country
         pha
         lda   cntry
         pha
         ldx   #WriteCString
         jsl   $E10000
         lda   menu_ans
         beq   skipit    if flag = 0 don't bother
         pea   $008E     set NORMAL mode
         ldx   #WriteChar
         jsl   $E10000
skipit   lda   #12
         clc
         adc   cntry     increment cntry pointer,
         sta   cntry
         inc   y         row #,
         lda   i         and loop index
         inc   a
         sta   i
         cmp   #28
         beq   q_and_a for i = 1 to 27
         cmp   #15
         beq   fixxy
         brl   menuloop  if it's not 15, continue
fixxy    lda   #19       htab to col. #19
         sta   x
         lda   #4        vtab to 4th row
         sta   y
         brl   menuloop
; Now it's time for questions and answers.
q_and_a  anop
         lda   #0
         sta   x
         lda   #19
         sta   y
         jsr   gotoxy
         lda   menu_ans
         bne   go_on     if flag <> 0 ask "continue?"
         pea   q_org|-16 print origin question
         pea   q_org
         ldx   #WriteCString
         jsl   $E10000
         jsr   get_int   get origin
         sta   org
         sta   org1      needed if org/dest get
                         swapped
         cmp   #1        test org
         bcc   q_and_a if org < 1
         cmp   #28
         bcs   q_and_a if org >= 28
         inc   menu_ans  twiddle the flag
         brl   menu      to print answers
go_on    pea   continu|-16  print continue question
         pea   continu
         ldx   #WriteCString
         jsl   $E10000
         pea   $0000     result space
         pea   $0000     echo flag - off
         ldx   #ReadChar
         jsl   $E10000
         pla
         and   #$007F
         cmp   #$0030    cmp to '0'
         beq   alldun
         stz   menu_ans  zap the flag
         brl   menu
alldun   anop
         plb
overnout rtl
; Fiddle as necessary and get amount of lunarium
; needed to travel from point A to point B.
calc     anop            routine to calc amount
         lda   dest
         cmp   org
         bne   sortod    if org <> dest
         lda   #' '      if org = dest
         sta   dec_dig
         rts             go no further
sortod   bcs   itscool   if org < dest, branch
         pha             dest
         lda   org       swap dest / org if
         sta   dest      org > dest
         pla             old dest -> new org
         sta   org
         lda   dest
itscool  sec             dest - org - 1
         sbc   org       dest - org
         dec   a         minus 1
         sta   tmp       save it for time being
         lda   org
         dec   a         (org - 1) * 2
         asl   a         to create correct offset
         tax             into offsets table
         lda   offsets,x
         clc
         adc   tmp
         tax
         lda   #0        zap accum.
         longa off
         sep   #$20
         lda   amnt,x    load hex digits
         rep   #$20
         longa on
         pha             push hex val.
         pea   dec_dig|-16  adr. for result
         pea   dec_dig
         pea   $0002     push len. of result
         ldx   #Int2Hex
         jsl   $E10000
         rts
gotoxy   lda   #0        zero reg.
         sep   #$20
         lda   x
         sta   >$000024
         lda   y
         sta   >$000025
         rep   #$20
         pea   $0000
         pea   $0000
         pea   $0000
         pea   $0000
         pha             a reg.
         pea   $0000     x reg.
         pea   $0000     y reg.
         pea   Tabv
         ldx   #FWentry
         jsl   $E10000
         pla             Y reg., dont' care
         pla             X reg., .
         pla             A reg., .
         pla             P, .
         rts
get_int  anop
         longa off
         sep   #$20
         lda   #$A0
         sta   >$000033  make the prompt ' '
         rep   #$20
         longa on
         pea   $0000
         pea   $0000
         pea   $0000
         pea   $0000
         pea   $0000     a reg.
         pea   $0000     x reg.
         pea   $0000     y reg.
         pea   GetIn
         ldx   #FWentry
         jsl   $E10000
         pla             y reg., don't care
         plx             x reg., char. count
         pla             a reg., don't care
         pla             processor status
; Convert input to an internal format integer.
         pea   $0000     result space
         pea   $0000
         pea   $0200     string pointer -> 0/200
         phx             string length (from get_ln)
         pea   $0000     0, result unsigned
         ldx   #Dec2Int
         jsl   $E10000
         pla             get result in A
         rts
title    DC    h'0C0D0D0F',c'Rocket Ranger Secret
                Decoder',h'0E00'
q_org    DC    h'0B',c'Country of Origin:',h'00'
continu  DC    h'0B',c'Hit any key, or 0 to exit: ',h'00'
menu_ans DS    2
i        DS    2
x        DS    2
y        DS    2
cntry    DS    4
org      DS    2
org1     DS    2
dest     DS    2
tmp      DS    2
dec_dig  DC    c'  - ',h'00'
; The country strings are fixed length at 12 bytes each.
country  DC    c'Algeria',5h'00'
         DC    c'Arabia',6h'00'
         DC    c'Atlantic',4h'00'
         DC    c'Brazil',6h'00'
         DC    c'Canada',6h'00'
         DC    c'Columbia',4h'00'
         DC    c'Congo',7h'00'
         DC    c'E. Africa',3h'00'
         DC    c'Egypt',7h'00'
         DC    c'England',5h'00'
         DC    c'France',6h'00'
         DC    c'Germany',5h'00'
         DC    c'Italy',7h'00'
         DC    c'Kenya',7h'00'
         DC    c'Libya',7h'00'
         DC    c'Mideast',5h'00'
         DC    c'Nigeria',5h'00'
         DC    c'Persia',6h'00'
         DC    c'Peru',8h'00'
         DC    c'Scandinavia',h'00'
         DC    c'Spain',7h'00'
         DC    c'Sudan',7h'00'
         DC    c'U.S.A.',6h'00'
         DC    c'U.S.S.R.',4h'00'
         DC    c'Venezuela',3h'00'
         DC    c'W. Africa',3h'00'
         DC    c'Yugoslavia',2h'00'
; The amnt strings get shorter because data is mirrored
; about the diagonal. The distance from point A to point B
; is the same for travel in either direction. NOTE: These
; "strings" are stored as hex digits to save space, but are
; displayed as ASCII decimal digits.
amnt     DC    h'28204337442529211614171531101
                923305022132436240181 2'
a2       DC    h'41575965301814342927242620173
                11270323319582361372 2'
a3       DC    h'32273437393318192622432931354
                244251736233830242 8'
a4       DC    h'49273848525045534746425437612
                55540444160283351'
a5       DC    h'28566251313638396444474855403
                435571041264243'
a6       DC    h'52645947465048624957456617514
                3612658123953'
```

```
a7      DC  h'22233633343217242615315440292
            05539502127'
a8      DC  h'15373233301423242519673531126
            128603426'
a9      DC  h'27242019251211181763282613502
            2542916'
a10     DC  h'08111340212233295315123032194
            12820'
a11     DC  h'121037172030285216092735214222
            515'
a12     DC  h'093518143224581016293915453
            113'
a13     DC  h'331412292557171126381844270 8'
a14     DC  h'28302227653936186334593229'
a15     DC  h'13192256271516432546261 1'
a16     DC  h'2818642125234616523610'
a17     DC  h'34493627174740431624'
a18     DC  h'722632215420624123'
a19     DC  h'6051623366204361'
a20     DC  h'20313714473818'
a21     DC  h'283424392319'
a22     DC  h'5632552225'
a23     DC  h'45244042'
a24     DC  h'534417'
a25     DC  h'3549'
a26     DC  h'30'
; Offsets are the distance from the 'amnt' label to
; the beginning of each of the amount strings.
offsets DC  i2'0,a2-amnt,a3-amnt,a4-amnt,a5-amnt'
        DC  i2'a6-amnt,a7-amnt,a8-amnt,a9-
            amnt,a10-amnt'
        DC  i2'a11-amnt,a12-amnt,a13-amnt,a14-
            amnt,a15-amnt'
        DC  i2'a16-amnt,a17-amnt,a18-amnt,a19-
            amnt,a20-amnt'
        DC  i2'a21-amnt,a22-amnt,a23-amnt,a24-
            amnt,a25-amnt'
        DC  i2'a26-amnt'
        end
```

Converting source code from ORCA/M to Merlin

It is a simple (but detailed) matter to convert to Merlin source format. Most of the major changes are ILLUSTRATED in figure 1. The chart is intended to provide examples of the changes needed, but it is NOT a complete list of all the changes. For example, remove or convert ALL occurrences of "anop" to " ". There are many instances where addresses are pushed on the stack, and they will all need to be changed. Remove LONGA and START directives. I think I've provide enough hints for someone to successfully convert the ORCA source to Merlin. In addition, there are a few other details to consider, but the Merlin disk and manual has examples to point you in the right direction. You're on your own if you want to convert to some other assembler, but the chart should still provide some guidance.

Once you've got the source code, you can assemble and link it. There are no macros needed. The resulting file will need to have its filetype changed to "CDA". You need to create a "/DESK.ACCS" subdirectory in the "/SYSTEM" subdirectory on the disk "/ROCKET.RANGER1" and copy the CDA into it. Now the CDA will be loaded whenever you boot up Rocket Ranger. Obviously, it can be included in the /DESK.ACCS subdirectory on any bootable P16 or GS/OS disk if you decide you can't get through the day without planning a trip.

Important Note: There are a couple of warnings to keep in mind if you use the CDA. One of them is; the game expects a button click to verify that the amount of lunarium is correct. Entering the control panel requires pressing the Open-Apple key, which the program will interpret as a button click. The second warning is; when you're in the control panel, the clock keeps ticking. If you spend too much time in one of the CDAs, you'll come back to the game to find that the Nazis have made huge advances in their campaign. It may be possible to stop the clock by hitting the space bar before you enter the control panel, but you'll have to execute the three-finger salute (open-apple control escape) "crisply".

There are a couple of additional edits you might want to make to the disk / Rocket.Ranger1. They are optional and purely cosmetic in nature. In Block $579 starting at Byte $181 is the following text string:

"Click button to transfer lunarium (see decoder wheel)"

I would recommend changing it to:

"See Rocket Ranger Decoder CDA for the lunarium needed"

This is the message that is at the top of the map display when you have to select your destination. It is easiest making this change with a disk editor that lets you enter text directly (like Copy 2+). For those of you without this capability, the hex values are:

```
53 65 65 20 52 6F 63 6B 65
74 20 52 61 6E 67 65 72 20
44 65 63 6F 64 65 72 20 43
44 41 20 66 6F 72 20 74 68
65 20 6C 75 6E 61 72 69 75
6D 20 6E 65 65 64 65 64
```

After making all these changes you'll never have to worry about losing your secret decoder wheel in the clutter on your computer desk!

Playing Tips for...

Rocket Ranger

Cinemaware

I suspect that some of you are wondering if I ever actually played the game, or if I just played around with the game. Well, I did both, and I managed to beat the Nazis and win. It was worth all the trouble! It's a very entertaining game, and the "punchline" at the end of the game is very satisfying. I did have to build myself a secret weapon to win, though. I have a memory that ranks anywhere from poor to lousy, so I printed a "scorecard" to keep track of what was where. The program code and a sample of program output follow:

Chart: A Scorecard Maker

```
10 REM Rocket Ranger Chartmaker
15 REM set nc (# columns) = 8 or
   14
20 DIM CNT$(27):NC = 8
30 FOR I = 1 TO 27: READ CNT$(I):
   NEXT
80 REM ? CHR$ (4)"PR#1": PRINT
   CHR$ (9)82+(nc=14)*50"N"
90 GOSUB 200
100 FOR I = 1 TO 27
110 PRINT CNT$(I);
120 FOR J = 0 TO NC: POKE 36,12 +
   J * 8: PRINT "|";: NEXT : PRINT
130 GOSUB 200
150 NEXT
155 PRINT CHR$ (4)"PR#0"
160 END
200 REM column maker
210 FOR J = 0 TO NC
220 POKE 36,12 + J * 8: PRINT
   "+";: IF J = NC THEN 230
225 FOR K = 1 TO 7: PRINT "-";:
   NEXT
230 NEXT
240 PRINT
250 RETURN
1000 DATA ALGERIA,ARABIA,ATLANTIC
   ,BRAZIL,CANADA
1010 DATA COLUMBIA,CONGO,E.
   AFRICA,EGYPT,ENGLAND
1020 DATA FRANCE,GERMANY,ITALY,
   KENYA,LIBYA
1030 DATA MIDEAST,NIGERIA,PERSIA,
   PERU,SCANDINAVIA
1040 DATA SPAIN,SUDAN,U.S.A.,
   U.S.S.R.,VENEZUELA
1050 DATA W. AFRICA,YUGOSLAVIA
```

Sample scorecard printout (partial)

```
          +------+------+--
Algeria   |      |      |
          +------+------+--
Arabia    |      |      |
          +------+------+--
Atlantic  |      |      |
          +------+------+--
Brazil    |      |      |
          +------+------+--
```

The chart is 27 "lines" down the page (one "line" for each country) and either 8 or 14 columns wide, depending on the variable "NC" (set in line 20). Set NC based on your paper width. Each "line" is 2 lines on the page; there is the top border of each space and the space itself, so the printout is actually 55 lines. It needs no special printer control commands. The DATA statements are copied from the BASIC decoder program.

I used some antique output devices (pencil, black and red felt tip pens) with the chart to keep track of everything. I used one column per game. I used the following notation convention;

- pencil in the letter "A" to note those countries I had agents in
- erase the letter "A" and pencil in a dash mark ("-") to show where I had pulled an agent out
- draw a RED slash through the "A" if the agent was killed (Oh, NOoooo!)
- pencil in a couple of letters to indicate targets as agents reported in. I used the following:

RF = rocket factory
a swastika symbol = secret lunarium base
sci = captured scientist
bwc = brain washing camp
FF = fighter factory
BF = bomb factory
art = stolen art treasures

- if I captured a target I used the BLACK felt tip to highlight the entry

I kept a notepad to record hints from the agents about where targets might be. I know, this sounds like a lot of bookkeeping, but it helped me defeat the Nazis. A world atlas, or almanac, is useful.

There is also an element of strategy to be considered. The Huns move "clockwise", starting in Europe and moving south and east, and then making the jump across the Atlantic to South America before heading north to Canada. Concentrating your agents in the Mid-East and in northern and eastern Africa, and then moving them south and west in Africa seems to turn up rocket factories and secret bases with fair reliability. Focus on capturing rocket factories and secret bases. Infiltrating a country slows the German advance, but you don't want to tie up too many agents. It's VERY important to capture a secret base early in the game and get the country infiltrated, so you'll have a supply of lunarium. Secret bases are ALWAYS in the desert or the jungle. You may have 5 agents working for you, but you can always go scouting on your own. While I'm waiting for my agents to report I'll go scout Canada, Columbia or Venezuela. If I turn up something, fine. If not, it's one less country to worry about. By the time I get back to the War Room my agents are ready to report. This also helps you avoid getting busted for dereliction of duty for loitering more than a year at Fort Dix.

Another element of strategy is knowing when (and how) to duck. It will help you survive the boxing matches, dogfights, ack-ack and gunfire. For boxing, pull the joystick towards you. For dogfighting try to stay high and to the left of the screen. For ack-ack and gunfire, stay low. You won't win the game without fighting, but covering up at the proper moment can help you survive.

My final piece of advice; if you get to the moon and beat the Amazon guards, DON'T RELAX when you see the screen that says "Congratulations...", because THE GAME AIN'T OVER! Stay alert and keep your joystick ready! Lotsa luck!

Curing the "Control Panel Lockout Blues"

There seem to be quite a few GS programs that prevent (lockout) user access to the control panel. I have three objections to control panel lockout:

1) I can't change my TransWarp speed settings
2) I can't get into the monitor (through Visit Monitor), and
3) I can't use the other Classic Desk Accessories (CDAs).

In the interest of freeing the control panel, I thought the following notes would be useful.

The various firmware vectors are located in bank $E1 starting at $E1/0000. While it is possible to change them directly with a tiny piece of machine language, most of the software I've seen uses the SetVector tool call. A member of the Miscellaneous tool set, it's documented on page 14-61,62 of the 2GS Toolbox Reference, vol.1.

To use the SetVector call requires that 1 word (2 bytes) and 1 long word (4 bytes) be pushed on the stack. The word is the vector number to be changed, and the long word is the 3 byte address (padded with an additional byte) to change the vector to. The X register is loaded with the value $1003 and a JSL to the tool locator is made. The code might look like this:

```
F4 12 00      PEA  $0012    control panel vector #
F4 xx xx      PEA  $xxxx    loader may change
F4 xx xx      PEA  $xxxx    these values!
A2 03 10      LDX  #$1003   set vector toolcall
22 00 00 E1   JSL  $E10000
```

Any instruction that pushes data on the stack can be used, so in addition to the PEA instruction, you might find PHA, PHX, PHY, PEI, and PER used. It's unlikely (but not impossible) that someone might use the PHB, PHD, PHK or PHP instructions.

To locate a SetVector toolcall, scan the disk for the sequence "A2 03 10". How do you tell if it's the guilty party? By examining the code, educated guessing, or trial and error (or all of the above).

To disable the lockout I make the following changes:

Blk	Byte	From	To
		22 00 00 E1	68 68 68 18

A listing of this patch would look like this:

```
PLA
PLA
PLA
CLC
```

This leaves the stack pointer set correctly and the carry flag clear (to indicate a successful toolcall).

Softkey for...

Silpheed, v. 1.09

Sierra

Silpheed uses a password to reduce the usefulness of unauthorized copies. While this allows someone to produce as many backup copies as they want, it can be a real nuisance when you just want to play the game. If you make a mistake in identifying the picture they display, the game restarts itself, a process that takes quite a bit of time

Figure 1: ORCA/M and Merlin differences

	ORCA/M		-vs-		Merlin 8/16, 16+	
rrdcda	start					
	longa	off				
	longa	on				
	dc	i1'cda_strt-cda_name'				
cda_name	dc	c'RR Decoder'			str	'RR Decoder'
cda_strt	dc	i4'rrd'			adrl	rrd
rrd	anop			rrd		
	pea	title-16			pea	^title
	sta	>$000024			stal	$000024
title	DC	h'0F',c'Deco',h'0E00'		title	hex	0F
					asc	'Deco',0E00
dec_dig	DC	c' -',h'00'		dec_dig	asc	' -',00
country	DC	c'Algeria',5h'00'		country	asc	'Algeria',0000000000
amnt	DC	h'282043'		amnt	hex	282043
offsets	DC	i2'0,a2-amnt'		offsets	dw	0,a2-amnt

and disk swapping (if you've got only one 3.5 inch drive).

It seemed a simple matter to disable the password check (in a segment called "copychecr"), but then the game wouldn't save/restore properly. Apparently, there are several locations that are referenced in both the password and disk routines. I finally decided on a scheme that would "blend in" with the original code, fooling the program into answering its own question.

Using my OMF file utilities (detailed in another article, probably around here somewhere) I determined that there were over 30 locations referenced by the copychecr segment, and over 10 of those were referenced by other segments. An older disassembly (done with Glen Bredon's Sourceror) had enabled me to document what copychecr did. A disassembly with the ORCA/Disassembler gave me a better idea of the locations I needed for my patch.

The resultant patch, while lengthy, is aesthetically and technically pleasing. It's makes use of the relocation dictionary, so that it's relocatable. I saw the game run in two different locations, due to whether or not I was using a RAMdisk for a second drive. It's possible it would run in yet another location if it was launched from the Finder, so the patch needed to be relocatable. This patch "tags" all the locations used by the original code, so the save and restore routines function normally (necessary if you ever hope to win the game).

The following changes should be made to a copy of the disk /SILPHEED1:

These first two edits kill the lines "TYPE IN ..." and "ENEMY SHOWN BELOW".

Blk	Byte	From	To
$448	$7C	22	AF
$448	$8F	22	AF

This edit is the "keyfake" code (shown below).

Blk	Byte	From	To
$448	$11C	08 E2 20 AF	08 AD 00 00
		00 C0 00 30	0D 00 00 F0
		02 28 6B 8F	02 28 6B AD
		10 C0 00 29	00 00 0A 18
		7F C9 08 F0	69 00 00 85
		04 C9 7F D0	6C B2 6C 6D
		16 AD 00 00	00 00 85 6C
		D0 03 4C 00	A9 00 00 E2
		00 22 00 00	20 B2 6C D0
		00 CE 00 00	23 A9 0D 80
		22	1F
Blk	Byte	From	To
$449	$130	03 00 3F 00	02 00 E0 00
		18 EB 06	02 20 3E
$449	$1F6	F8 00 02 D9	EA 00 02 DF
$449	$1FE	FD 00	0B 01
$44A	$5	00	07
$44A	$C	04 01 02 D9	EF 00 02 7C
		60	61
$44A	$22	12 01	F6 00
$44A	$31	1D 01 02 D9	E3 00 02 E1

The patches in blocks $449 and $44A are made to relocation records. Two of them (bl$449, b$1FE and bl$44A, b$5) are cRELOC records (type $F5) and need to be pointed AWAY from my code, while the rest are cINTERSEG records (type $F6) and are used to relocate/patch the operands of the instructions in lines 26, 27, 32, 35 and 38 in the keyfake routine.

After applying these patches, the screen will now show the message "SUPER DOG-FIGHTER FLIGHT SCHOOL", the name of the fighter will appear below it, the message to center your joystick will appear next, and the fighter will be displayed at the bottom of the screen. All you need to do is center the joystick and press a button.

It seems to me that Sierra could have squeezed things down and rearranged the disks to make the game more convenient to use from a single 3.5 drive. Until they do that, I recommend setting up a RAMdisk of 500+ K, copying Silpheed1 or its major files to RAM, and using the real 3.5 drive for Silpheed2. (The files from Silpheed1 require less room than those from Silpheed2.)

The file that is patched is "GAMEPLAY", with a MOD Date of 25-MAY-89 and a CREATE Date of 8-MAR-04. The file "SILPHEED.SYS16" has a MOD Date of 25-MAY-89, 15:19 and a CREATE Date of 25-MAY-89, 14:53.

The original Silpheed keyboard input routine:

```
08DE:08          PHP      ;get input, bl 448, b 11c
08DF:E2 20       SEP   %00100000  short accum.
08E1:AF 00 C0 00 LDAL  $00C000
08E5:30 02       BMI   H08E9
08E7:28          PLP
08E8:6B          RTL
08E9:8F 10 C0 00 STAL  $00C010
08ED:29 7F       AND   #$7Fclear hi bit
08EF:C9 08       CMP   #$08^H  "<-"
08F1:F0 04       BEQ   H08F7
08F3:C9 7F       CMP   #$7Fdelete char
08F5:D0 16       BNE   H090D
08F7:AD 40 B3    LDA   HB340   char count
08FA:D0 03       BNE   H08FF
08FC:4C 85 09    JMP   H0985
08FF:22 3C 0A 01 JSL   $010A3C
0903:CE 40 B3    DEC   HB340   char count
0906:22 3C 0A 01 JSL   $010A3C
090A:4C 85 09    JMP   H0985
090D:C9 15       CMP   #$15^U  "->"
090F:D0 15       BNE   H0926
0911:AD 40 B3    LDA   HB340   char count
0914:C9 09       CMP   #$09^I  tab char
0916:F0 6D       BEQ   H0985
0918:22 3C 0A 01 JSL   $010A3C
091C:EE 40 B3    INC   HB340   char count
091F:22 3C 0A 01 JSL   $010A3C
0923:4C 85 09    JMP   H0985
0926:C9 0D       CMP   #$0D    <cr>
0928:D0 08       BNE   H0932
092A:A9 01       LDA   #$01
092C:8D 44 B3    STA   HB344   <cr> flag
092F:4C 85 09    JMP   H0985
```

Silpheed Patch Source

```
* silpheed keyboard routine replacement
*
* This patch needs patches to the
* relocation dictionary to guarantee that
* it will work. The patches are:
*
*  Block Byte  To
*  $448  $7c   AF
*  $448  $8f   AF
*
*  $449  $130  02 00 E0 00 02 20 3E
*  $449  $1F6  EA 00 02 DF
*  $449  $1FE  0B 01
*
*  $44A  $5    07
*  $44A  $C    EF 00 02 7C 61
*  $44A  $22   F6 00
*  $44A  $31   E3 00 02 E1
*
         org   $8de     block 448, byte 11c
         tr    adr
         mx    %00
         php
         lda   $9087    $02/3e20, 1st pass flag
         ora   $b348    $02/60e1, 'nother flag
         beq   keyfake  if both are 0
         plp
         rtl
keyfake  lda   $b346    $02/60df, # of fighter name
         asl
         clc
         adc   #$b3e3   $02/617c, names addresses
         sta   $6c
         lda   ($6c)
         adc   $b340    $02/60d9, char. count
         sta   $6c
         lda   #0       clear accum.
         sep   $20      short a
         lda   ($6c)    load a char
         bne   $0926
         lda   #$0d     <cr> character
         bra   $0926
```

Edward Eastman — NE

This is a follow up to my recent Epson Label Maker submission. After using this for a couple of months, I decided that there were some improvements that could be made. For instance, when you get done editing a line, it should automatically move down so you can edit the next line without hitting 'Z'. Change line 3030. There is also a bug in the routine to reedit after printing. It comes back with one line greater than allowed. Change line 6080. Change line 3000 for better length accuracy. One more thing is that it will print ctrl-characters when entering them, possibly messing up the screen. Change lines 2040 & 2050.

```
2040 Q$(L) = Q$(L) + A$:: IF A$ >
   CHR$ (31) THEN PRINT A$;
2050 PRINT " ": GOTO 2000
3000 VTAB (L + 2) * 2 - 1: IF
   Q$(L) = "" THEN LW(L) = 0: GOTO
   3030
3030 PRINT "="LW(L) / 1000" IN.";:
   CALL - 868:A$ = "Z": GOTO 1020
6080 L = 1
```

For those of you who like 80 columns, here are more modifications. Even if you don't have an 80 column card (gasp), or don't want to use it, I recommend making these modifications then deleting or REMing lines 90, 6005, and 6090.

```
90 PRINT CHR$ (4)"PR#3": PRINT
500 FOR L = 1 TO LI: GOSUB 3000:
   NEXT :L = 1
2010 IF A$ = CHR$ (13) THEN PRINT
   " ": GOSUB 3000:A$ = "Z": GOTO
   1020
3030 PRINT "="LW(L) / 1000" IN.":
   CALL - 868: RETURN
6005 PRINT : PRINT CHR$ (27) CHR$
   (17)
6090 PRINT CHR$ (4)"PR#3": PRINT
6110 PRINT "START FRESH? (Y/N) ";:
   GET A$: IF A$ = "N" OR A$ =
   CHR$ (110) THEN 230
```

In older 80 column cards, there is a bug that positions the cursor at 40 (or is it 41?) instead of 1 on a HTAB 1 command. If your card does this, you will need to change the '+1' in line 2000 to '+2', and the 'HTAB 1' in lines 3010 and 6110 to 'HTAB 2'. Add the following lines if you can not enter an 'ESC', or if your screen does something weird while entering ctrl-keys.

```
1000 VTAB 20: PRINT : PRINT "A/Z
   TO CHANGE LINE NUM. OR 'P' TO
   PRINT": PRINT "'ESC' TO EXIT OR
   'RTN' TO EDIT LINE #"L;: GOSUB
   4000
2000 VTAB (L + 2) * 2 - 1: HTAB (
   LEN (Q$(L)) + 1): GOSUB 4000
4000 INVERSE : PRINT " ";: NORMAL
   : PRINT CHR$ (8);
4010 WAIT ( - 16384),128:A$ = CHR$
   ( PEEK ( - 16384) - 128)
4020 POKE - 16368,0: RETURN
```

J L Walters has made a point of collecting the complete works of "Krakowicz", one of the more prolific writers on copy de-protection in the "early days" of Apple computing. Some of the material is dated but all of it is interesting. Beginners should read from start to finish, old hands can probably skim thru quickly. There are 22 parts in all. We'll print them, in order, in upcoming issues. My personal thanks to Mr. J L Walters for the time and effort he spent in putting together this collection and for sending it to us. Presented now are parts 3 and 4.

Krakowicz

The Basics of Kracking (part 3)
Memory moves, binary files, and kramming for the finals

In the last episode, we pondered the starting address of a program and ways to find it in spite of the protectors' subterfuge. This time we'll discuss how to get the program into saveable format, even if it's too long to save as a BFILE. Although we'll be referring at first to single-load programs, most of these techniques are applicable to programs with disk access.

Before we begin the process, let me philosophize for a few seconds on the procedures and practices to be used. This is a discipline: perhaps not so demanding as championship karate or the unification church, but it requires knowledge, patience, and attention to detail. I urge you to begin each adventure in kracking with a sharp pencil, plenty of paper, and a good eraser. From this point forward in our quest, record-keeping will occupy an important part of the total activity. If you have a printer, print out any pertinent sections of code and write in your own comments about what it means. Write down every address of interest, and keep especially careful notes of the nature and sequence of all memory moves, starting points, and tricks used by the protectors. Do this not just because it's character building, but because unless you have exceptional recall, all programs will eventually blend together into a warm and fuzzy memory. Keep good notes on everything you learn, and remember: "Those who cannot recall the mistakes of the past are doomed to repeat them."

Suppose you have loaded in, reset with your old monitor ROM, and finally located the starting address to the greatest game ever written: "HYPERSPACE ANDROID CLONE KILLER" or "HACK". The starting address is $4123, and the game occupies memory from $800 to $B000. You already know that if any memory above $9D00 has been used by the program, DOS is dead, and you can't save the program to disk with a DOS command. As you also undoubtedly know, if the program were smaller you would have the option of booting a disk and saving the game as a binary file. Let's take just a second, though, and review what happens to memory when you boot a disk.

First of all, don't use a master disk, since the DOS on a master is loaded first into $1600-$3FFF and then relocated to the higher regions of memory. Booting a 48K slave disk will disturb only $0-$8FF and $9600-$BFFF, and if your program lives within or can be rearranged to fit these boundaries, you can safely boot a slave disk and save the program as a binary file.

An old method of saving a binary file is well-known to those of us who bought Apples in the dark ages before the DISK II, but there are now maybe half a million (!) Apple owners who are unfamiliar with the cassette port and its use. In general, almost any cassette recorder that has a tone control can be used, but for some reason the cheaper ones are generally better. To use one, plug both cables into the correct connector ("in" means into the computer, not into your recorder), and turn the tone control almost to the top of the treble range. Save a small BASIC program (refer to the manual for use of the BASIC commands) at any old volume control setting. Try loading the program back in several times, increasing the volume control setting until the program loads reliably. You'll find that the tape works very well, even on long files, especially when the same recorder is used to record and playback.

What's good about the tape system is that even when DOS is completely dead, the monitor commands for tape I/O are still active (assuming you didn't wipe them out of your old monitor ROM). See the reference manual, page 46 for a complete description. With tape, you can always save any part of memory at any time! (Worth keeping in mind for those crucial situations when the system crashes just as you are finishing your term paper on the word processor). The cassette routines use only locations $3C-$3F and $42-$43 in zero page, and the only part of memory you shouldn't try to save is $C000-$C0FF— some terrible things can happen if you try. In most cases, it's best to save a long program in two files so it can be reloaded in between 800 and 9600 after DOS is in memory. For our example of "HACK", the necessary monitor commands are:

```
0.4FFFW                    (Long wait)
5000.AFFFW                 (Longer wait)
```

After booting a disk, you can reload with:

```
1000.5FFFR                 (Reload first half)
BSAVE HACKLOW,A$1000,L$5000
1000.6FFFR                 (Load second half)
BSAVE HACKHI,A$1000,L$6000
```

Note that in the tape read and write commands, unlike DOS, the actual starting and ending locations are listed. Be sure you understand the one-byte difference between the two before you use them.

There are also occasions when you would like to save Applesoft or Integer BASIC

programs loaded in from a modified DOS on a protected disk (Arcade Machine and the Rapid-Fire series from SSI are examples). This is simple with the tape recorder, since the monitor routines are totally ignorant of the operating system in RAM. If you can list a BASIC program, you can usually save it to tape. Try the following with one of the above programs: load in a program module (anything in Arcade Machine except the main menu), then hit reset while it's running. Type D6:00 (this removes the Applesoft internal "protection"), then C081 to select the mother board ROM (unless you have an Apple II with Applesoft on a ROM card, then it's C080 to select slot 0). Type control-C and you should be able to list the program and then save it to tape with the "save" command (sometimes an additional fairly trivial protection scheme is used with Applesoft programs: deleting the first line number so it won't list. It will still save to tape and you can reconstruct the line number at your leisure). Remember that the basic "LOAD" and "SAVE" commands don't allow a file name to be added. If there are more than a few files on the disk, this is a very tedious way to krack a program, but back in the middle ages before DEMUFFIN PLUS it was sometimes the only way. You also have to be wary of binary routines which are called from or modify the BASIC programs.

Yes, you're right, getting out and hooking up the tape recorder is a cramp in the Calvins, so it's usually left for emergencies when nothing else works. In general, it's best to learn how to manipulate memory to scrunch your program down into a DOS file (it will always have to be done, anyway). In the best of all possible worlds, your DOS would be in ROM memory, and would allow you to save any program that resided in RAM memory. In the real world, it's generally necessary to reduce a program to a file that can be loaded in by DOS from a normal disk (we'll talk later about those that can't be). This process is usually called "memory moving", and the purpose is to "tuck in" all the pieces of the program that lie outside the normal program memory of $800-$9600 allowed by DOS. The other half of the process is the "unfolding" of the tucked-in portions of memory after the program is reloaded under DOS. To gain perspective on the process, let's look at memory maps with DOS active and with and without "HACK" in memory.

Note: The Old Monitor ROM/Autostart ROM affect only the HiRAM memory locations from $D000-FFFF. Items shown in paraenthesis are the same for either ROM. Inspector/Watson are 2K ROMs that use the unused ROM space under Integer BASIC. DOS 3.3 actually uses memory from $9600-BFFF.

Before we begin the discussion of the techniques of memory moving, let's restate the objective: We're trying to arrange all the program into a small enough space that we can BSAVE a file under DOS (the DOS manual will tell you that the largest binary file you can save is 128 sectors, but if you change location $A964 (43364) to $BF(191) you can save a file as large as the entire RAM memory). Remember that booting a slave disk will mess up $0-$8FF and $9600-$BFFF, so the largest file it's practical to save is about 145 sectors (you can, with care, overwrite much of the screen memory and pages 2 & 3 to save a BFILE of about 151 sectors, but that requires knowledge and considerable care).

Looking at the memory map with HACK, you can see that the memory from $9600 to $B000 will have to be stored somewhere else to bring the file size down, and the page from $800-$8FF will have to be stashed temporarily during the disk boot to restore DOS. To find out what areas of memory are free, search through all memory with the Inspector and look for blank pages. The following trick will help: Before you load the original, clear all of memory to zero (or any other byte you like) with:

800:0 N 801<800.95FFM

Then you'll be able to see unused memory areas. This doesn't always work, since many areas are copied to a second location and not used afterwards, so if you're hard pressed for storage memory, it's a good idea to scan through once with the Inspector set to decode ASCII to detect suspicious sectors (lately, some of the protectors have taken to storing garbage such as source code in unused pages of memory and on empty disk sectors). Note down any pages that are totally clear, any that are all one byte, regardless of what is, or any that contain junk. Let's assume for this example that locations $1000-$1FFF and $8000-$8FFF are blank. We have $1A00 ($B000-$9600) bytes of memory "leftover" or outside of the DOS boundaries, so they will all fit into the $2000 blank locations that we located.

Store the excess bytes in the holes by typing:

8000<9600.A5FFM

1000<A600.AFFFM

or equivalent; the split can be any way that helps you keep track of the process. Finally, stash the memory from page 8 with **1B00<800.8FFM**. Remember that this is only temporary. before you do anything else, boot your 48K slave disk, then restore page 8 with **800<1B00.1BFFM**. Before you do anything else, save the program with **"BSAVE HACKALL,A$800,L$8E00** (nine out of ten times you'll forget to change $A964; consider changing it in the DOS in memory before you initialize the disk so it will be permanent). You can now take a deep breath and relax: all of the program memory is safely tucked away. All that's left is to write a short program to reverse the memory storage.

Two short routines, similar to those shown in our first basics lesson are required. Again, let's review the steps necessary from here to run the game:

1. Load the (compressed) game into $800-$95FF.
2. Move the piece of memory at $8000-$9FFF to $9600-$A5FF.
3. Move the piece of memory at $1000-$19FF to $A600-$AFFF.
4. Jump to the starting address at $4123.

The following program will take care of steps 2-4. It may not be immediately obvious that this program must be stored within the compressed program in a page that is both empty and unaffected by the memory moves you are about to make. In this case, page $1C is safe.

```
1C00    LDY #$0       Clr Y-reg
1C02    LDA $8000,Y   Get a byte at 8000+
1C05    STA $9600,Y   Store it at 9600+
1C08    INY           Incr. counter
1C09    BNE $1C02     If not pagend, redo
1C0B    INC $1C04     Incr. source hibyte
1C0E    INC $1C07     Incr. dest hibyte
1C11    LDA $1C07     Get the dest hibyte
1C14    CMP #$90      If 90, we're done
1C16    BNE $1C02     If not, do more
1C18    LDA $1000,Y   Repeat the process
1C1B    STA $A600,Y   for the second
1C1E    INY           block
1C1F    BNE $1CA8
1C21    INC $1C1A
1C24    INC $1C1D
1C27    LDA $1C1D
1C2A    CMP #$1B
1C2D    BNE $1CA8
1C2F    JMP 4123      and jump to the starting
                      location
```

This may seem hard at first, but the form is so constant that you'll be able to write these moves in your sleep after a few tries with the mini-assembler (the place you'll most likely mess up is in the "CMP #$90" by typing "CMP $90" —watch it carefully!).

Time out for a brief discussion of one of the subtle points of memory moves. Although you're generally able to make your memory moves non-overlapping, you can have a problem moving large amounts of memory. The memory move routines shown above are "forward" memory moves: that means that each page move is one ahead of the one just moved. Sometimes you will need to move, for instance, locations $6000-$8FFF to $8000-$AFFF. If you use the forward moves as shown, you can see that the first page (page $60 or $6000-$60FF) will land at $8000-$80FF, smack on top of the original page that was supposed to be moved later to page $A0 ($A000-$A0FF). To avoid this conflict, you can use what's called a "backwards" memory move. This technique moves the last page first and works "down" in memory instead of up. In this example, page $8F is first moved to $AF, then $8E to $AE, etc. This way, when it finally comes time for page $60 to be #moved to page $80, the original page $80 will already have been moved. A typical routine for this is:

```
1000    LDY #$0
1002    LDA $8F00,Y
1005    STA $6000,Y
1008    INY
1009    BNE $1002
100B    DEC $1004
100E    DEC $1007
1011    LDA $1007
1014    CMP #$5F
1017    BNE $1002
```

OK—all that remains is to get to the start of the earlier memory move routine when we "BRUN" the game. This is accomplished by putting the code for "JMP $1C00" or 4C 00 1C at location $7FD-$7FF and making this the first location of the program. We can then save a complete, functioning version of HACK with "BSAVE HACK,A$7FD,L$8E03". This creates your final, 145-sector file of HACK which will BRUN whenever you wish.

A few helpful hints

1. Always keep a few initialized 48K slave disks nearby—it's alarming how fast a disk fills up with slightly different 145-sector versions of the program undergoing kracking.

2. Make your program names as descriptive as you can, especially when saving a program in pieces. It's very disturbing to return to a kracking effort after a long weekend to find programs on the disk titles "HACKHI", "HACKHIGH", "HIGH", "HH", etc. And not be sure what each one is. Better to type in a few extra letters to let you know that it's "HACK WITHOUT 9600UP" or "HACK 4000-B000 ONLY".

3. Whenever possible, compress the game to the minimum number of sectors by doing a few more memory moves before and after saving. Your friends will appreciate your thoughtfulness in maximizing the number of games per disk and minimizing modem time.

4. =>VERY IMPORTANT<= When you think you have a complete, working version, check it out thoroughly on all levels and in all modes. It's extremely embarrassing to have to issue a "product recall" when you learn a month later that HACK crashes on level 47 just as the Hypergalactic Frog is about to devour New Pittsburgh on the Mars colony...

The Basics of Kracking (part 4)
"Where do I begin?"

Several previous episodes of this column have dealt with the relatively simple techniques which can be used to save a single-load file to disk as an unprotected binary program, and it is now time to explore the larger area of multiple-program disks, programs with disk access, and the approaches used to protect them from being copied. We will begin with sizing up a disk protection scheme, deciding on a basic approach, and beginning the unprotection process (the subject of boot-tracing as another means to the same end will be described in a future episode, since it is generally used with more sophisticated protection schemes). The subject is truly mammoth, and will require several episodes to complete. For now, settle back, open a cold beverage of your choice, and let's begin a journey into the first level of disk protection: The modified DOS (as we have often mentioned before, two stalwart friends in this quest are "Beneath Apple DOS" (BAD) by Worth and Lechner, and Randy Hyde's "DOSSOURCE". It is possible to krack disks without them, but with about the same ease as performing an oral appendectomy).

Apple's DOS, combined with the division of hardware between the disk controller card and the disk analog board, is a veritable playground for those who produce disk protection. There are literally thousands of different things which can be done to make copying a disk difficult, challenging, and (maybe someday) impossible. In so doing, they provide hours of very ingenious puzzles, boundless intellectual stimulation, and not incidentally, the incentive to learn much more about programming, the Apple, DOS, assembly language, and treachery than we would otherwise have the desire to learn.

By far the most common technique used to protect entire disks is to make modifications to the operating system, and specifically to the Read/Write Track and Sector (RWTS) routines which defeat ordinary copy programs (COPYA and SUPER DISK COPY 3.X are examples, but we'll see later how both of these can be used to our advantage). To find the most efficient approach to defeating these protection techniques, we

Figure 1

Location	Without HACK	With HACK
F800-FFFF	Monitor ROM / Autostart ROM	Monitor ROM / Autostart ROM
F000-F7FF E800-EFFF E000-E7FF D800-DFFF D000-D7FF	↑ Integer BASIC ↓ INSPECTOR WATSON ↑ Applesoft ↓	↑ Integer BASIC ↓ INSPECTOR WATSON ↑ Applesoft ↓
C800-CFFF C000-C7FF	Peripheral slot ROM space Soft switches & slot ROMS	Peripheral slot ROM space Soft switches & slot ROMS
B800-BFFF B000-B7FF A800-AFFF A000-A7FF 9800-9FFF	↑ DOS ↓	(Empty) ↑
9000-97FF 8800-8FFF 8000-87FF 7800-7FFF 7000-77FF 6800-6FFF 6000-67FF	↑ Program memory	↑ Hack program (Empty) (Empty)
5800-5FFF 5000-57FF 4800-4FFF 4000-47FF	↑ (Hi-res page 2) ↓	↑ (Hi-res page 2) ↓
3800-3FFF 3000-37FF 2800-2FFF 2000-27FF	↑ (Hi-res page 1) ↓	↑ (Hi-res page 1) ↓
1800-1FFF 1000-17FF 0800-0FFF	↓ (Text page 2)	↓ (Empty) (Empty)
0000-07FF	(Zero pg, Stack, Text page 1)	(Zero pg, Stack, Text page 1)

need first to spend a fair amount of time describing DOS and RWTS from a krackist's viewpoint.

(On the fundamental principle that giving a man a fish allows him to eat for a day while teaching him to fish allows him to eat for life, we will not dwell on the subject of "copying" as such. Many of the techniques described here are, however, very useful in deciding how to go about copying a disk. perhaps an aspiring author out there will build from the introduction given here to pursue the subject in depth...?)

Before we can get to the core of the matter, we must understand much more of the processing and encoding systems used by DOS to store information on the disk. This is fairly heavy stuff, but your kracking ability depends more than anything else on your knowledge of this subject. Try your best to work through it now, and the rest of the process will be much easier.

We already know that each track consists of 16 sectors which each represent one page (256 bytes) of data. A sector actually consists of two separate parts, an address field, which tells DOS which sector it is, and a data field, where the actual bytes are stored. To begin a trip around the disk, let's look first at the byte sequence taken from a normal, unmodified DOS disk at track 0, sector 0 (as we mentioned earlier, the terms "byte" and "nibble" are often used interchangeably to refer to the data read off the disk. The use of "nibble" is not really accurate in reference to DOS 3.3, but persists for historical reasons).

Take a look at figure 2. The first few FF's [1] are known as gapbytes, but they're correctly termed syncbytes, and we'll treat them as simple separators for now. Next are the three most important bytes on the disk, D5 AA 96 [2]. This sequence may not occur anywhere else on the disk except at thestart of an address field, and serves as a unique identification marker. These bytes are known by all sorts of colloquialisms, including "address header", "header bytes", "leader bytes", "prolog", and others. They will always, repeat always, occur on at least sector 0 of track 0 of every Apple disk which boots under DOS 3.3 (The first law demands it).

The next four sequences encode the volume number [3], track number [4], sector number [5], and checksum [6]. Each number is a single byte, written in an old-style encoding scheme called 4+4 nibblizing. This is a format for storing data on the disk in which the even bits of a byte are stored in one 8-bit sequence (representing one-half of the original byte or one nibble), and the odd bits are stored in the second "byte" (the requirement for this sort of "byte-splitting" or nibblizing was established largely by the limitations imposed by disk drive hardware. You can find much more information in BAD, pp. 3-12 to 3-21, but an oversimplification is that, in the old days, at least every other bit read from the disk had to be a logical "one", or the circuitry that read the disk "forgot" where it was and what it was doing). If you are interested in more detail on the mechanics of the 4+4 scheme, refer to the very first kracking korner file on CYCLOD and the file on WAY OUT. Figure 4 lists the values of nibbles of interest to us in this format:

We can now decode the four groups of bytes as: [3] vol number (FF+FE) = $FE (254 decimal), [4] track number (AA+AA) = $00 , [5] sector number (AA+AA) = $00, and [6] checksum (FF+FE) = $FE. The first three are self-explanatory, and the last is used to detect any errors which may creep in after many hours of disk use. Following these is a sequence of bytes [7] used to mark the end of the address field. A total of three bytes (DE AA EB) are written to the disk, but only the first two are checked when the field is read. this pair of bytes is known variously as "closing bytes", "trailers", or the "epilog". Finally, there is another series of gapbytes [8] which separates the address field from the following data field.

The data field has a similar structure.(See figure 3.) Where the gapbytes [1]) are same group that ended the address field. The data marker bytes [2] are also called by all the names mentioned for the address marker, and are interpreted by DOS as "here comes the data..." The big stretch of 342 bytes [3] is a very complex way of storing 256 bytes on a disk, following some compromises made with the original laws of disk recording. Without going into exactly why, each "byte" can represent only 6 bits of an original byte, which means that each byte has two bits left over. Packing these together at 6 bits each requires another 256/3 or 86 diskbytes, for a total of 256+86=342 "bytes", which no longer represent a nibble or half a byte, but 3/4 of a byte (make up your own name for it, there's no real agreement what it should be called).

Following the data is a single checksum byte [4], which will give zero when exclusive-ORed with all the other bytes from the data, and then the same active closing bytes that were used in the address field [5]. Finally, more gapbytes [6] pad the space between this data field and the address field which comes next.

This sequence is repeated 15 more times to make a complete track, and there is usually a large "gap" of up to 128 FF's separating the last and the first sectors on the track. One final item of interest is that the sectors do not normally follow each other in numerical sequence of 0,1,2...etc. The actual sequence (called "skewing" or interleaving) is chosen for speed of reading and writing, and can vary on some DOS's which are otherwise strictly identical in format to DOS 3.3.

Figure 2: Address Field

```
—FF FF FF FF D5 AA 96 FF FE AA AA AA AA FF FE DE AA EB FF FF FF FF —
          / \      / \   / \   / \   / \   / \      / \
 _____(1)__/   \_(2)/   (3)   (4)   (5)   (6)     (7)    \____(8)__
```

Figure 3: Data Field

```
—FF FF FF D5 AA AD —342 BYTES OF DATA— (CKSUM) DE AA EB FF FF FF—
         / \      / \                /     \     / \      / \
 ____(1)_/   \_(2)/   \_____(3)____/       \(4)/     (5)    \__(6)__
```

Figure 4: 4 plus 4 Conversion Chart

AA + AA = 00	AE + BA = 18	BA + EA = 60	BE + FA = 78	EB + AA = 82	EF + FA = DA
AA + AB = 01	AE + BB = 19	BA + EB = 61	BE + FB = 79	EB + AB = 83	EF + FB = DB
AA + AE = 04	AE + BE = 1C	BA + EE = 64	BE + FE = 7C	EB + AE = 86	EF + FE = DE
AA + AF = 05	AE + BF = 1D	BA + EF = 65	BE + FF = 7D	EB + AF = 87	EF + FF = DF
AA + BA = 10	AE + EA = 48	BA + FA = 70	BF + AA = 2A	EB + BA = 92	FA + EA = E0
AA + BB = 11	AE + EB = 49	BA + FB = 71	BF + AB = 2B	EB + BB = 93	FA + EB = E1
AA + BE = 14	AE + EE = 4C	BA + FE = 74	BF + AE = 2E	EB + BE = 96	FA + EE = E4
AA + BF = 15	AE + EF = 4D	BA + FF = 75	BF + AF = 2F	EB + BF = 97	FA + EF = E5
AA + EA = 40	AE + FA = 58	BB + AA = 22	BF + BA = 3A	EB + EA = C2	FA + FA = F0
AA + EB = 41	AE + FB = 59	BB + AB = 23	BF + BB = 3B	EB + EB = C3	FA + FB = F1
AA + EE = 44	AE + FE = 5C	BB + AE = 26	BF + BE = 3E	EB + EE = C6	FA + FE = F4
AA + EF = 45	AE + FF = 5D	BB + AF = 27	BF + BF = 3F	EB + EF = C7	FA + FF = F5
AA + FA = 50	AF + AA = 0A	BB + BA = 32	BF + EA = 6A	EB + FA = D2	FB + EA = E2
AA + FB = 51	AF + AB = 0B	BB + BB = 33	BF + EB = 6B	EB + FB = D3	FB + EB = E3
AA + FE = 54	AF + AE = 0E	BB + BE = 36	BF + EE = 6E	EB + FE = D6	FB + EE = E6
AA + FF = 55	AF + AF = 0F	BB + BF = 37	BF + EF = 6F	EB + FF = D7	FB + EF = E7
AB + AA = 02	AF + BA = 1A	BB + EA = 62	BF + FA = 7A	EE + BA = 98	FB + FA = F2
AB + AB = 03	AF + BB = 1B	BB + EB = 63	BF + FB = 7B	EE + BB = 99	FB + FB = F3
AB + AE = 06	AF + BE = 1E	BB + EE = 66	BF + FE = 7E	EE + BE = 9C	FB + FE = DE
AB + AF = 07	AF + BF = 1F	BB + EF = 67	BF + FF = 7F	EE + BF = 9D	FB + FF = DF
AB + BA = 12	AF + EA = 4A	BB + FA = 72	EA + AA = 80	EE + EA = C8	FE + EA = E8
AB + BB = 13	AF + EB = 4B	BB + FB = 73	EA + AB = 81	EE + EB = C9	FE + EB = E9
AB + BE = 16	AF + EE = 4E	BB + FE = 76	EA + AE = 84	EE + EE = CC	FE + EE = EC
AB + BF = 17	AF + EF = 4F	BB + FF = 77	EA + AF = 85	EE + EF = CD	FE + EF = ED
AB + EA = 42	AF + FA = 5A	BE + AA = 28	EA + BA = 90	EE + FA = D8	FE + FA = F8
AB + EB = 43	AF + FB = 5B	BE + AB = 29	EA + BB = 91	EE + FB = D9	FE + FB = F9
AB + EE = 46	AF + FE = 5E	BE + AE = 2C	EA + BE = 94	EE + FE = DC	FE + FE = FC
AB + EF = 47	AF + FF = 5F	BE + AF = 2D	EA + BF = 95	EE + FF = DD	FE + FF = FD
AB + FA = 52	BA + AA = 20	BE + BA = 38	EA + EA = C0	EF + BA = 9A	FF + EA = EA
AB + FB = 53	BA + AB = 21	BE + BB = 39	EA + EB = C1	EF + BB = 9B	FF + EB = EB
AB + FE = 56	BA + AE = 24	BE + BE = 3C	EA + EE = C4	EF + BE = 9E	FF + EE = EE
AB + FF = 57	BA + AF = 25	BE + BF = 3D	EA + EF = C5	EF + BF = 9F	FF + EF = EF
AE + AA = 08	BA + BA = 30	BE + EA = 68	EA + FA = D0	EF + EA = CA	FF + FA = FA
AE + AB = 09	BA + BB = 31	BE + EB = 69	EA + FB = D1	EF + EB = CB	FF + FB = FB
AE + AE = 0C	BA + BE = 34	BE + EE = 6C	EA + FE = D4	EF + EE = CE	FF + FE = FE
AE + AF = 0D	BA + BF = 35	BE + EF = 6D	EA + FF = D5	EF + EF = CF	FF + FF = FF

This would be a good point, if you're not already very comfortable with the sequences described above, to get out a utility which will perform a "nibble read" of a disk track (Inspector, Nibbles Away, Locksmith, etc.), and read in a standard DOS track. Scan through the bytes until you come to the magical D5 AA 96 sequence, then compare all the bytes which follow it to the description given above. Try a few tracks and decode the start of several sectors until you become familiar with the appearance of them. You'll save yourself a lot of time and effort by becoming familiar with the appearance of normal DOS sectors and tracks.

Knowing that all these things are required to make a disk compatible with DOS 3.3 (and make it copy with COPYA), you can easily see how to make a protected or modified DOS: Simply change almost any one of the important bytes in either or both fields, and make the appropriate changes to the read and write routines in DOS. In order to appreciate what this means, let's spend a minute or two on the structure of DOS.

Just as was Gaul, DOS is divided into three main parts. The first one, called the command interpreter, has been described in considerable detail by Bert Kersey in the now-classic "DOS BOSS" program and the short series he authored for Softalk magazine. While DOS command changes may figure in a number of protection schemes, they are usually not the primary system, and will be described with a few examples where appropriate. The second third, known as the "file manager", is not commonly altered as a protection scheme. Cogent descriptions of this part of DOS are rare, but some hints about its function, liberally paraphrased and rewritten from BAD were given in the "TYPE ATTACK" file.

The crux of most DOS alteration schemes is the last third of DOS, the RWTS routines. These exist in everyone's 48K Apple between $B700 and $BFFF, and are the only subroutines which read from or write to the disk under anything resembling a normal DOS. After a command (keyboard or program) has been processed by the command interpreter, and the right part of the right file has been selected by the file manager, the RWTS routines are called on to do the crucial job of exchanging information between the Apple's memory and the diskette.

Space prevents us from listing all the routines, but those of particular interest are shown in figure 5.

As before, you are strongly urged to get as familiar as you can with these routines, using DOSSOURCE and "Beneath Apple DOS" as your primary references.

Returning to the subject of detecting and circumventing modified DOS's, you have a choice. You can either look for changes by inspecting a track, or you can search through the above RWTS routines for something that isn't normal. Neither approach will work 100% of the time, so it's best to become proficient at both. Figure 6 lists most of the crucial locations in RWTS that are commonly changed for the purpose of protection. Any of the locations can be modified, either permanently (which means that you can probably find the changes in the DOS image on tracks 0-2), or temporarily. The temporary DOS changes are much tougher to find than the permanent ones, since the changes may be erased after they have been used.

A good example of this was "MASK OF THE SUN" and "THE SERPENT'S STAR", where the main disk is protected (among other techniques) by using FF's for all the epilog bytes, but the save game is written out and read in using the normal DE AA's. A pair of subroutines was called to swap the bytes in and out as required. Much more devious was the protection scheme used by TSR on "Computer Dungeon" and "Theseus and the Minotaur", where epilog bytes were computed according to which track was being read.

There are many other examples of DOS modifications used to keep us at bay, including some secondary protection techniques,

but we'll look at those after we describe the basic approaches to unprotecting these disks in part 5 of the Basics of Kracking. In the meantime, study the format, become familiar with the standard tricks, and remember:

"The night shall be filled with music,
and cares that infest the day,
shall fold their tents like the arabs,
and as silently, steal away."

Henry Wadsworth Longfellow

The Overlord _____ NY

"Load Runner" Virus Fix

There seems to be some confusion about a virus called 'Load Runner'. This virus is characterized by a long period of inactivity before 'waking up' and erasing block 0 of the currently active disk.

The virus is from France, and does no actual harm to your disk data. It copies itself off an infected disk and into memory, and from there it copies itself onto every disk booted while it is still in memory. It is impossible to know if your disks are infected with it until one of them boots up with the Load Runner title screen and consequently is treated to a little block 0 surgery. After this surgery the disk won't boot, but your data can still be accessed by Copy II Plus or any other utility.

Load Runner's trigger appears to be an odd-numbered day in October (as read from your computer's clock). For example, an infected disk which is booted on October 13 will trigger the virus and erase block 0.

I have sent a disk to COMPUTIST with a Load Runner Virus Killer program which was written by some people out in California (I think). I do not know who these people are, but the program is very effective.

Send $2 for the program on disk or download the program from the COMPUTIST BBS. ..... RDEXed

When you run it, it will wipe Load Runner out of memory if it is there, and then test your disks one-by-one for Load Runner. If it is found, it will fix the disk and destroy that copy of the virus.

Be sure to check ALL of your disks (even data disks), because just one remaining infected disk could re-infect your entire library before the trigger is pulled and you are alerted!

Call The Oblivion (516) 922 4213
East Coast Connection Base #2

Rob Fiduccia _____ MD

Update on Ultima V DCS

Hello again. I'm writing to tell the readers that there is a new version of Ultima V Dungeon Construction Set and that it is not $8.00 anymore, it's free, but you'll still need to mail a disk and return postage with your letter please. There are too many programs and they're too big to send in or type in.

The Level Editor is now in assembly and the Chamber Editor has been completely reprogrammed by me with the Contents editor within it. The manual is no longer in a paper format but in text files. Each program with its own file. Plus 3 paper tours to get you started. I've made very helpful reference sheets for the Level Editor and Chamber Editor too. If you did send your $8.00 then you'll get it back with Ultima V DCS.

Please note that the programs were written on my Apple IIgs, so programs will go normal speed for non-IIgs users. If you have an Apple IIgs, programs will goes considerably faster if in FAST system speed.

You'll need to send 1 disk and $1 for shipping and handling. There is a new newsletter too if you wish for that. Tell me in your letter how you feel if we start a DCS club. Send comments and request to me at:

Ultima V DCS
2809 Broadview Terrace
Annapolis MD, 21401

What Ultima V DCS is

There are 7 full programs to use to design or modify a dungeon. Here they are and what they do in brief:

1. Level Editor

Edit any of the 8 levels in any of the 8 dungeons in the game. On screen top text editing. Fast assembly format. Save, load, or get information. Change level to level fast and easy. Menu controlled.

2. Chamber Editor

With this newly written version, you have fast on screen top text editing at your hands. Total map cursor movability. Menu driven. You can edit all 16 chambers in all 8 dungeons. Total control over all 256 tiles. Now the contents editor is within the program for fast edit of all 256 contents. Change all 8 Hallucinations to make unlimited chamber effects; like a moving bookshelf or tricky exits. Change all party member's positions. Put in new treasures to get. Put in monsters, move their position. Put ladders in chambers, make levels complex.

3. View Editor

This program is fun to use. It can make a door a wall, a fountain a trap or everything a hallway make peer a gem in return 'not work' but work. Fast and explanatory. Save, Reset or Load.

4. Entrance Editor

Change dungeon names, Words of Power, and collapse any dungeon or destroy it on Britannia or Underworld.

5. Type Editor

Make any dungeon any other type. The types are dungeon, mind and cave. The type of a dungeon is what its interior is like.

6. Chamber Resurrector

Use this program to bring back chambers after they've been victorious in. This program does all 8 dungeons at once.

7. Party Transferor

If you make a full dungeon scenario, you'll need to let your friend(s) or others use their party if they wish to on the edited Britannia disk.

CPR Agent _____ Canada

Pirates or Saviors?

There are two words that never fail to elicit groans and moans and instant boredom from readers of COMPUTIST - software piracy. Everyone has had it to the ears with the whimpering and whining over the "dishonesty" in the computer community. But I would like to address an angle that needs as much (if not more) attention than "dishonest pirates."

Many years ago I played in a teenage rock-and-roll band. Like most teenagers we had visions of grandeur of making our fortunes grinding out ear-splitting rhythms that would reshape the American teen culture. We were not that bad, but it became clear very quickly that it wasn't a matter of how good you were. Rather it was a matter of who you knew and what connections you had. Moreover, we discovered that the recording industry was a game with a whole set of rules all of its own. And the name of the game was MONEY!

I never lost my interest in the development of the music industry. Consequently, I see it as the biggest unspoken ripoff of the century that computer marketing companies charge anywhere from $50 to $100 for a computer program! An average album requires considerably more technical work than does any computer program. (I can see many readers shaking their heads already, but I'm convinced that it is true.) Besides the musicians themselves, there are literally dozens of other technicians of all sorts who work with, finance, and operate sound equipment that costs a small fortune. By the time an album is actually ready to market, it's not unusual for tens of thousands of dollars to have been paid out in its preparation. And what does it sell for? Most are in the $10-$15 dollar range - a far cry from the $50-$100 that the computer marketing people expect!

Further,I think the analogy here is a fair one. I'm sure that many computer programs cost tens of thousands of dollars to get on the market but the difference is in the selling price. Computer marketing people realize that they are catering to a specialized group and consequently try to bleed it for all they can. (I think the irony here is the same as it is in the recording industry - a very small percentage actually goes to the artist. The rest goes to the marketing people who are the real racketeers!)

It took the motion picture industry several years to figure out that the way to stop piracy of movies was to bring the price down to within reason. It wasn't that long ago that a movie on cassette was at least $100. Now you can buy almost any movie for $30. The difference? Now most people can afford their own original and the motion picture industry is doing more business than ever before. (Some companies still haven't figured this out!)

Consequently, when a computer company flashes a colorful wrapper and boasts the ultimate computer experience for $89 and it turns out to be akin to the way you moved your bowels Tuesday last, you are going to have Captain Blood doing the distribution trick like you never dreamed possible! He is anxious that his friends not get schmucked the same way he did!

There are growing numbers of computer operators who are fed up with getting burned by cellophane! I personally have a whole library of programs that I'm ashamed to admit I paid money for! Many are old, old, creations that run so slowly, old age itself smiles in the rear view mirror - all packaged like they were a computer genius' gift to the Christ child! These same programs are still touted in the catalogues as "educational", "new editions", of "unequalled quality" and are presented with color and sheen that would beguile the wariest of saints! Now may I ask "Who is the Pirate?" and "Who is dishonest?"

It seems to me that we simply have one manipulator hiding behind the skirts of the law and having the nerve to complain when another manipulator lifts those skirts and rapes the program for "flashing" a cellophane "top" and threatening to charge $100 for it! Get serious! Anyone who has been burned even once has been burned one too many times. It's not a matter of circulating programs to your friends. It's a matter of preventing your friends from making the same stupid mistake you made! I have several friends who send me programs regularly who say "I heard you were looking for this one. Take a good look, cross it off your list, and format the disk. Believe me, you don't really want this program in your collection." I have been very grateful to these friends on every count. They have saved me a great deal of money. And they were right! But I do feel badly about one aspect of this "piracy"

Figure 5

Address	Name	Function
B700-B749	—	Do 2nd stage boot load, RUN HELLO program
B793-B7B4	RWPAGES	Read or write a group of pages
B7B5-B7C1	CALLRWTS	Disable interrupt and call RWTS
B7E8-B7F8	IOBLOCK	RWTS parmlist-see BAD
B800-B829	PRENIB	Convert bytes to nibbles for writing
B82A-B8C1	WRITE	Write sector to disk
B8C2-B8DB	POSTNIB	Convert nibbles to bytes after reading
B8DC-B943	READ	Read sector from disk
B944-B99F	RDADR	Read an address field
B9A0-BA28	SEEKABS	Position read head to the desired track
BA29-BA68	NIBL	Write translate table
BA69-BA95	(EMPTY)	=>WATCH THIS SPACE<=
BA96-BAFF	—	Read translate table
BB00-BBFF	NBUF1	Buffer (prim) used to stash the nibbles
BC00-BC55	NBUF2	Sector buffer for nibbles
BC56-BCC3	WRADR	Write address field (only during init)
BCDF-BCFF	(EMPTY)	=>BE SUSPICIOUS<=
BD00-BDEC	RWTS	Main read/write origin
BDED-BE03	RDRIGHT	Good read, ck track #
BE10-BE25	RTTRK	Right trk, ck vol#
BE26-BE45	CRCTVOL	Right vol#, ck sect#
BEAF-BFB7	DSKFORM	Initialize disk
BFD8-BFC7	SECMAP	Sector interleave map
BFC8-BFFF	PATCHES	Corrections for small DOS bugs =>BEWARE<=

Figure 6

Address	Normal value	Purpose
B853	D5	Data Addr Marker 1-write
B858	AA	Data Addr Marker 2-write
B85D	AD	Data Addr Marker 3-write
B89E	DE	Epilog byte 1
B8A3	AA	Epilog byte 2
B8A8	EB	Epilog byte 3-not read
B8AC	FF	Epilog byte 4-not read
B8E7	D5	Data Addr Marker 1-read
B8F1	AA	Data Addr Marker 2-read
B8FC	AD	Data Addr Marker 3-read
B92A-C	D9 00 BA	Location for checksum compare
B935	DE	Epilog byte 1-read
B93F	AA	Epilog byte 2-read
B942	38	Set carry for I/O error
B955	D5	Addr Data Marker 1-read
B95F	AA	Addr Data Marker 2-read
B96A	96	Addr Data Marker 3-read
B991	DE	Addr Epilog byte 1
B99B	AA	Addr Epilog byte 2
BA29-68	*	Write translate table
BA96-FF	*	Read translate table
BC5F	FF	Sync Byte value
BC7A	D5	Addr Marker Write-1
BC7F	AA	Addr Marker Write-2
BC84	96	Addr Marker Write-3
BCAE	DE	Addr Epilog byte 1-write
BCB3	AA	Addr Epilog byte 2-write
BCB8	EB	Addr Epilog byte 3-write
BFB8-C7	*	Sector interleaving table

* See DOSSOURCE listing for correct contents.

and that is that the manufacturer can not hear my disk drive format his disk!

Do you really want to stop computer piracy? Perhaps some advice to computer companies is in order (do those guys actually read this publication?):

1) Put the cost of the program into its proper perspective. (There are 3 and 4 originals of some audio tapes in my home simply because members of my family want and CAN AFFORD their own!)

2) Get rid of copy protection schemes. When I started cracking computer programs, I did it to shore up the completely depleted budget of a small school that simply could not afford the time or money to keep sending programs back to the company for repair. I quickly learned that some of the protection techniques were far more interesting than the programs themselves! Now I crack programs for the sheer pleasure of it! Most of the time the protection scheme is far and wide the best part of the disk (and the only part that poses any "educational" worth!) I am honestly disappointed when I get a new program and it has no protection scheme.

3) And finally, say it the way it is on the wrapper. Customers show their appreciation for sales honesty by coming back to the same company again and again. And they keep their copies in their own library!

Now who is listening out there? Can you write as well as read? I've shot off my mouth, so now let's hear what you've got to say, big boy!

Softkey for...

Muppets on Stage (3.5")

Sunburst

Requirements:

original disk
fast copier (I used Copy II Plus)
block/sector editor

Step-by-step

1. Fast copy the disk - ignore errors
2. Search for 20 26 84 90 07 C9 FF F0 0B and change the 20 26 84 to EA EA EA. On my disk it was at track $0E, sector $0F, bytes $00-02.

Details

Actually I was moving this program back to a 5.25" inch disk so the schools could use it on computers with 5.25" inch drives. I first formatted the 5.25" disk with ProDOS using the FORMAT option on the opening menu of Copy II Plus. Then it was simply a matter of transferring files from the 3.5" inch disk to the 5.25" inch disk. Once all the files had been moved over, I booted the disk and it died after the opening graphics. I had expected as much. Through the process of elimination, I discovered that it was the LOADER1 file that was doing the disk check.

Next I used the CATALOG function of Copy II Plus to catalog the disk and give the file loading locations and file lengths. I discovered that LOADER1 loaded into memory at $4300. I then booted ProDOS, BLOADed LOADER1, typed CALL -151 to get into the monitor, and typed 4300L to examine the code at that location. I noticed that the code immediately jumped to $8400 (4C 00 84). I typed 8400L to check what was going on there. Sure enough there was a check for some special bytes and two routines that could send the program to it's death if the slightest error occurred. By eliminating the first 3 bytes at this location (20 26 84 to EA EA EA) the code is allowed to proceed and the program to run normally.

Whether you are using 3.5 inch disks or 5.25 inch disks the de-protection is the same. Just fast copy the disk, search for the signature check bytes, replace them with EA's and smile a lot!

Softkey for...

In Search of the Most Amazing Thing

Spinnaker/Tom Snyder

Have you ever had a friend bring you a disk and explain in a decidedly warm sort of way how much he likes the program and would you mind making a backup copy for him? You agree, take the disk home, and then discover that it does not work properly. What do you tell him? No matter how carefully you decide to word the bad news he is going to be heartbroken and/or think that you ruined the best program in his library! This very situation happened to me and it was my desperation that led to the following notes on how to de-protect this program AND how to fix the problem with it not running properly.

When I first examined this disk under the Nibble Editor of Copy II Plus, it seemed that removing the copy protection would not be too difficult. The address prologues had been changed from D5 AA 96 to AA D5 AB, the data prologues had been changed from D5 AA AD to AA D5 EB, the address epilogues from DE AA to DE AB and the data epilogues from DE AA to ED AA. After "customizing" the Sector Editor to use these same values, I was able to read in any sector on the disk and examine it. There was a DOS catalogue on Track $11 and everything else seemed pretty standard. I thought I would capture the RWTS from this disk and use Super IOB to swap the files to a regular DOS disk.

When I booted the original and jumped into the monitor, I thought I would check out the RWTS at $B800-$BFFF before actually going ahead with the process. Bad news! It had been altered - seriously altered, and I could tell this process would never work. Being obstinate, I tried it anyway. I was right - it didn't work.

I booted the original disk a second time and jumped into the monitor. I decided to try using more of the RWTS and to try using it with DeMuffin Plus. This time I started the move from $B600 instead of the $B800 and moved it down to $6600 where it would be safe while I loaded DeMuffin Plus. After moving it back to where it belonged and starting up DeMuffin Plus, I was able to transfer files as sweetly as you please from the original disk to my already formatted DOS disk.

The file transfer went just fine until the program arrived at a file called CRUNCHED STORE. Here the drive rattled and the program halted. Since DeMuffin Plus offered me the option of transferring more files, I said yes, went back to the beginning, and answered "yes" to the DO YOU WANT TO BE PROMPTED" question. This time I told it not to transfer the files that had already been copied AND the file that had given the trouble. The program went ahead and copied all the other files without incident.

Wondering if I hadn't somehow messed up a byte or two in the process of capturing the RWTS, I went through the entire process again - in fact, several times. All to no avail. For some strange reason DeMuffin Plus would not read that one single file from the original disk. I decided to check out the files that I had already transferred as well as the original program. I write-protected the disk with the newly transferred files on it and booted it. It loaded and ran beautifully. Then I noticed a "store" option as part of the program. I chose that option, the disk drive started up, and the program died. I decided this was good news and bad news - bad news in that I still had to get that file, and good news in that it seemed to me I could boot the original, go to this part of the program, allow the program to load this file, and then go into the monitor and pick it out of memory and save it to disk. The plan seemed feasible.

I booted the original and moved the program to the store option. I chose it and got ready to go into the monitor. However, in the middle of loading this file, the drive rattled and died! This was when I discovered that the original disk had a problem. Somehow it had become damaged and this file would not load. Now I had a REAL problem.

I reasoned that this REAL problem was, in actuality, a READ problem. It occurred to me that if I went into the original disk's DOS, I could disable the error checking routines and perhaps bypass the problem. But which routine would I disable (disabling all error checking routines is asking for trouble - sometimes in ways that you hadn't thought of!)

After contemplating this situation for some time, I remembered that the options of the Sector Editor of COPY II PLUS allow you to disable or enable checks for epilogues, checksums, and tracks. If I enabled each of these one at a time and scanned through all the sectors on the disk, I would find the problem and know which routine I would need to disable in the DOS. It would be tedious, but it should work.

As it turned out, it was tedious but it did work. There was a checksum problem on track $0E, sector $09. Pleased with myself, I booted the original disk and captured the RWTS yet another time. I hunted back through my back issues of COMPUTIST for those invaluable charts of Bill Jetzer's (COMPUTIST #60 pages 12 and 13) where Bill identifies the precise addresses in DOS 3.3 (and DOS 3.2) of all the markers, checksums, etc. in DOS. I noted that he identified $B98A as the address of the checksum check and that a 00 would disable this routine. However, when I went to this address in the modified DOS, the B7 that should have been there wasn't. I searched forwards and backwards through the code hoping to come across the same routine in a different location. No luck. Some of the code looked familiar but there were several changes - enough to make me very apprehensive.

I had to do something, so after perusing it all several times, I noticed that the whole routine ended like so many others - with a 38 60 18 60 (38 to set the carry flag if there is a problem and 18 to clear it if there isn't and then return). I changed the instruction at $BA17 from a 38 to an 18 (so that whether there was a problem or not the carry flag would be cleared and the program would proceed) and prayed that this was the right routine.

I loaded DeMuffin Plus for the ninth time in this fiasco, chose the file CRUNCHED STORE and pressed RETURN ever so gently. The drive started up and transferred my file as neatly as if nothing had ever been wrong. I kept my fingers crossed as I booted the newly created disk. I chose the store option. The drive started, up came the graphic, and I was presented with a cart for the shopping trip!

I took the disk and the copy back to my friend. He smiled warmly and wanted to know "How was it?" I hesitated like a program doing a signature check. "No problem," I lied. "It's a great program but...but..ah.....I think the copy will run better than the original." "Why?" he asked. "It just usually works out that way," I said.

Assuming that your disk has no problems here is the cookbook approach to the deprotection:

Step-by-step

1. Format a DOS disk (delete HELLO).
2. Boot the original and at the prompt use your favorite method to get into the monitor

6600<B600.BFFFM *move RWTS to a safe place*

3. Replace your original with your newly created slave disk.

6 ctrl P *to boot the disk and load a normal DOS*

4. Remove the slave disk and replace it with your disk containing DeMuffin Plus.

BLOAD DEMUFFIN PLUS,A$803
CALL -151 *to get back into the monitor*
B600<6600.6FFFM *to move the modified DOS back to where DeMuffin can find and use it*
A851G *to be sure DOS is connected*
803G *to start up DeMuffin Plus*

5. Follow the prompts and transfer all files.
6. Add a fast DOS (Pronto DOS is the best for handling the HiRes images used in the program).

That's it. I hope you find the most amazing thing!

A Crash Course in Software Deprotection

In most of the past Computist issues, there have been various articles on methods for deprotection of specific schemes. It is my hope that, in submitting this, all the most common protection schemes on the market today will be covered in such a manner that even the novice can understand what's happening. If nothing else, hopefully whoever reads this will at least get a basic knowledge of what goes on when someone sets out to crack a disk. So without further ado:

Hardware and Software Requirements for Basic Deprotection

To be an effective deprotector one must have the right hardware and software for the job. You must have a method of getting into the monitor. It is useful to have a way into the monitor such as crack card or (if you're using an older II) an old F8 crack-ROM or the equivalent in the form of a switchable crackrom.

Some of these ROMs are or were available on the open market, such as Lockbuster, Senior Prom (no longer available, though still supported by Cutting Edge Enterprises) and the ROMs sold by Don Lancaster. Other devices such as Master Key+, modified integer cards and some of the popular kracking cards can also get you into the monitor.

If you do opt to go for an integer card (again, on the older II's), it is useful to get the Inspector and Watson in ROM. Locksmith makes good use of these utilities, if you don't need/want the integer card but have the Inspector/Watson available to you. Master Key+ and the Senior Prom also do many of the same functions as Inspector and Watson.

It is also useful to have an NMI card. Several, such as Wildcard, are available in computer stores. You may need one with the ability to store the stack pointers after the interrupt. The NMI card is useful, but not a necessity.

It will prove most convenient to have more than one disk drive and some specific disk drives and accessories are extremely useful such as Track-Star and the Knowdrive.

Now that you know what hardware is useful or necessary, it is time to call some of your friendly deprotectionist and get some cracking utilities. First on the shopping list is a disk search utility like Tracer from CIA. Next on the list is a sector editor. Inspector and Watson, Copy//+ or Nibbles Away are all capable of handling our chores for us. You will also need COPYA from the system master. Also useful, but not necessary, is FID. Also needed are several programs written specifically for cracking. These include, but are not limited to, Advanced Demuffin, Demuffin Plus, Fastloader Create, COPYB, and others that will be mentioned in the series. Most of these (such as Super IOB) are available from your Computist back issue library.

How To Approach A Basic Deprotection Task

I am frequently asked, "How do you deprotect games?". The answer to this question is not that simple (I know, to most of you experienced readers, this is nothing new, but lately a lot of readers have been asking for articles geared toward the beginner, so skip ahead if any part of this article drags). Not every protection scheme is the same (some are, but not all). The very first thing you should do upon opening the package of a game is see if the game will copy with COPYA or a fast disk copier, such as Locksmith's FastCopy. If it does and it works then it is not protected (No Kidding, right?). Believe it or not, I have been given originals to crack that turned out not even to be protected! Not everything is protected.

If the game did copy but does not boot correctly there is probably some sort of nibble count or disk check routine. If this is

the case then look at the nibble count section of this series.

If the game does not copy at all, there is a possibility that the game has a modified DOS or a loader with a modified format. To learn about this method of protection check the modified DOS section of this series.

If the game loads all at once and you can remove the disk and continue playing without problems, then you have a single load program and should read the appropriate section.

Usually companies will use combinations of the above methods in an attempt to thwart those who know how to defeat one but not all of the above. If you make a game into COPYA format and make the correct modifications to make it work under DOS 3.3 (or ProDOS), and it still does not work, then you may have a nibble count. Occasionally single load games will have nibble counts or unnecessary disk access.

Some protectors go even further than this (for example Electronic Arts' Wasteland and subLOGIC's Flight Simulator II) although most of the schemes you will see will fall into one of the above categories. Some companies rely on unusual nibblizing schemes, self modifying code, moved catalog tracks and other methods of protection. Hopefully all of these methods of protection will be sufficiently covered in this series. I realize, of course, that this one article did not answer the question of how to approach deprotection completely, but it should start you out. Deprotection, naturally, is too broad a subject to be covered in just one article.

Deprotecting Single Load Programs

Single load programs are those that load up all at once and read no further from the disk. This means that you can remove the disk and continue playing the game to completion. Many older games are single loading, but today this type of game is a very rare thing. I love these games because when correctly deprotected, as a file rather than a whole disk, they are easy to store and many can be fit on the same disk. During this episode you will need a way into the monitor. NMI cards such as Wildcard, Replay and Crackshot will krack many of these for you at the press of a button, but this is undesirable because:

a) The file may not work every time

b) The file will be a lot bigger than it needs to be

To deprotect single load programs, you must do two things; first you must find the programs starting location and the areas used by the program; then you need to save out all of the parts and link them together to make one binary file. You will probably need to write a memory move routine as well. Sounds easy enough, right?

First of all we need to find the programs starting location. To do this we can first try several possible (commonly used) starting locations. This could possibly save us a little time. Boot up your protected disk and reset into the monitor once the game is fully loaded. Type 0800G <cr>. If it starts up and is ready to play, then 0800 was the starting address. If it does not, then try these locations in the same manner (Note:the G after 0800 is the monitor command to go from that location): 0801,0803,0C00,4000,6000, or any increment of 1000 that appears to be a possible beginning. It is also useful to look around the 0800 area for jumps to the starting location.

If these possible locations all turn up empty then we may need to try something else. Some programs (a few) will restart the game on a (normal) reset. The reset (autostart) is a programmable key and can be programmed to jump to the start again on reset. If this is the case then reset (into the monitor) and check the reset vector. This is at 03F2 and 03F3. Remember that in assembly addresses are stored in backwards order. Try this address as the starting location. For example if you look at the reset vector and find a BD and a 08 respectively then try 08BD as the starting location.

If the game does not restart upon hitting the autostart reset (and instead reboots or hangs) then we will have to try another approach. Because most games use graphics and possibly begin with a hi-res graphics screen, it seems logical that the software company would start the program and turn on the graphics almost immediately afterwards (some do this, others do not). Assuming that they do this, we should try and find where the program sets up its graphics. This is not too hard, because to set up graphics on an Apple, you need to access a series of softswitches. These are places, that upon being referred, execute some part of graphics (Note:there are softswitches for other functions. Some of these will be mentioned in later chapters).

The softswitches for apple graphics are as follows:

C050 - selects graphics
C051 - selects text
C052 - full screen
C053 - mixed text and graphics
C054 - page 1
C055 - page 2
C056 - lo-res
C057 - hi-res

To access any of these, one only needs to refer to these in the following one of the following ways:

```
LDA $C050
CMP $C050
STA $C050
ROL $C050
BIT $C050
EOR $C050
```

Many authors use the same pattern:

```
LDA $C054
LDA $C057
LDA $C050
```

and possibly with the addition of LDA $C052.

So now we know how to set up graphics, but what does this amount to? If we know that some of these are probably going to be used at the beginning, then we can search through memory with a memory search routine (the Inspector/Watson and the Senior PROM come with this function built in) for the bytes that set up graphics. If the game you are working on is so old that it uses lores you would search for 50 C0 (backwards remember) or 56 C0. If it uses hi-res you would search for 50 C0 or 57 C0. Write down the address where each of these are found in memory.

To see if an occurrence is the starting location, go to each address and try to find the beginning of the routine. Look backwards in memory (from the location of the graphics bytes) and find the end of the previous routine. Routines usually end in either a JMP or an RTS so the first you see will be the end of the last routine. Try the next instruction after this as the start. For example, if you find the byte sequence 50 C0 at location 085D and tracing back you find an RTS at 0851 then you would try 0852 as the start. Suppose you found a JMP $1000 at 0851 instead of the RTS. You would then try the instruction after this as the start ($854).

If it does not start here, we may need to search back farther. The routine that you were looking at can only be reached in one of three ways. It could be branched to by any of the branching instructions, but branching can only go about 127 bytes forward or backwards, so look forward about a page and if you do not see a branch into your routine, then check backwards. If you find a branch to your routine then try the beginning of its routine as the start.

The routine also may have been jumped to (i.e. JMP $xxxx where xxxx represents the previously suspected start location). So search for 4C xx xx in memory. If you find one then try the beginning of its routine as the start.

The routine may also have been JSR'd to so do the same as in the above paragraph (substituting a JSR for the JMP). Repeat these steps until you find a place only reachable by a JSR or JMP.

If none of the above methods work to find the starting location, there is yet another possibility. Many games expect you to hit a key before you start the game. To understand how this is done, look at the following code:

```
8000:   LDA  $C000   read keyboard strobe
        BPL  $8000   no key pressed
        STA  $C010   reset keyboard strobe
        JMP  $0800   if key pressed go to start
```

or if the key to be pressed is the space bar then add these two lines before the JMP:

```
CMP #A0
BNE $8000
```

So, we need to search memory for any access to the keyboard strobe and trace back in memory for the start as mentioned in the earlier discussion of the graphics setups.

Another way to find the starting location is to check out a restart key. For example, many older Broderbund games let you press control r to restart the game. You would again need to search for accesses to the keyboard strobe. Then you need to see if the byte compared is the ASCII value of the restart key. For example if control r restarts the game, then search for accesses to $C000 and then see if $92 is compared. You can find these hex values in most Apple reference manuals.

If everything else fails, you may need to resort to boot tracing (see appropriate section). If you find JMP tables (several JMP's in succession) you may be able to figure out which is the actual start location by trial and error.

Now assuming you have, somehow, found the starting location, what now? We must find out what areas are used by the program, and need to be saved out. How do we know what areas are used by the game? Well one way we can find out is to clear memory and then boot the game. If you reset out and scan memory you can (if they have not loaded trash onto blank pages) assume that any code that is not a bunch of zeros, is needed. Be careful in discarding areas of memory as just being trash. Sometime shape tables and other important areas may seem to be garbage, but really are not.

To clear out memory, type the following:

```
0800:00
0801<800.95FFM
```

As you become more experienced at this kind of deprotection, you will get to know which areas are needed and which are just trash. Once you have decided upon which areas must be saved (Note:if you have the correct starting location and your final copy does not work, then you probably did not save out everything needed), you need to put the pieces onto a standard DOS disk. To do this you need to boot up DOS and save out the pieces. But when you boot up a DOS 3.3 disk many important parts of memory are trashed. So instead we should use a 48K DOS-less disk because it only destroys 0-800 and 9600-BFFF. Because we may need to save parts of programs that reside within one of these areas, you may need to move them to a safe page of memory and save them one at a time. For example suppose we decide that our program uses A000-A000 for part of the program. We would move it down to a safe area of memory:

```
8000<A000.B000M
```

and save it out by booting the slave disk (C600G will boot a disk from the monitor), and typing:

```
BSAVE PART.ONE, A$8000, L$1000
```

(Note - be sure to keep track of where pieces should be located on your finished product!)

Also of importance is the fact that 800-8FF is destroyed upon hitting an old F8 reset. This is because when you hit reset a monitor prompt appears at the bottom of the screen. To print this the Apple must scroll the top line of the text page off to print at the bottom. Many times software artists will put valuable code here to screw deprotectionists that have a true old F8 rather than a Krack-rom with the ability to move this code to safety before entering the monitor. If you are caught by this problem (ie. you have a true old F8 as your only means of entering the monitor) I can only suggest that you get a Krack-rom with this ability, a copy-card, a Master Key+ or any comparable device that will allow you to do this, or learn how to boot trace effectively.

After you have everything saved and you know the correct starting location, you will need to write a memory move routine and put all of the pieces together as one binary file. The memory move routine will need to move all the pieces to their appropriate locations in memory (for example it would need to move the piece in the last example back to A000). The memory move writer in the Master Key+ utilities or Fastloader Create will write it for you. Otherwise you can just write your own (Did I forget to mention that you would need to know assembly?)

Now I realize that understanding all of the above crap may be rather difficult at first (even with all of the examples). So now I will go through the deprotection of a hypothetical game called Spy-man (The object of the game is to maneuver your agent through a maze chasing down KGB and other various underworld scum). Anyway let's assume this game has fairly decent graphics and the game is a fairly recent release. By searching for graphics bytes (C050 and C057) and tracing back to an RTS you find that 2000 is the starting location. By clearing memory, booting up the game, resetting and scanning through memory you find that the areas used by the program are 2000-8FFF and A000-AFFF. You would first boot up a slave and save out the first part:

```
BSAVE SPYMAN ONE,A$2000,L$6FFF
```

Then you would boot up the game again, reset and then move the other part to a safe place in memory:

```
1000<A000.AFFFM
```

and boot up a slave disk and save it as:

```
BSAVE SPYMAN TWO,A$1000,L$FFF
```

and you would then need to write a memory move routine (to move the code at 1000 back to A000 and then jump to the starting location at 2000). You would then load each part and the memory move routine and save it out as a file:

```
BLOAD SPYMAN TWO,A$1000
BLOAD SPYMAN ONE,A$2000
BLOAD MEMORY MOVE,A$900
BSAVE SPYMAN,A$900,L$8100
```

Here is a memory move for this example:

```
900:    LDY  #$00      this clears the y register
902:    LDA  $1000,Y   get byte at 1000+
905:    STA  $A000,Y   store at A000+
908:    INY            increment Y
909:    BNE  $902      go back unless page ends
90B:    INC  $904      increment source high byte
90E:    INC  $907      increment dest high byte
911:    LDA  $907      get dest high byte
914:    CMP  #$B0      if B0 we are done
916:    JMP  $2000     jump to the start
```

(Note:this routine is taken heavily from a routine written by Krackowitz and I do not want to assume credit for his work). You could also have used one of the memory move writers (as I stated earlier) and they may well be easier than writing your own. The above was and example of a forward memory move. But what if we need to move, say, 6000-8000 onto 7000-9000? Obviously if we move this as normally we will be writing over 7000-8000 before we have a chance to move the code that was to be moved later. For this type of problem, we can write a backwards move routine. For example:

```
5D00:   LDA  #00
5D02:   LDA  $7F00,Y
5D05:   STA  $8F00,Y
5D08:   INY
5D09:   BNE  $5D02
5D0B:   DEC  $5D04
5D0E:   DEC  $5D07
5D11:   LDA  $5D07
5D14:   CMP  #$5F
```

5D17: BNE $5D02

Note - again this routine is taken from Krackowitz but it does have a couple of changes.

Well this brings my discussion of single load deprotection to a close. I hope I have explained this subject well enough to help you rather than just confusing you.

Deprotection of Modified-DOS schemes

Note: the scope of this article is only to cover DOS 3.3 schemes. ProDOS is not discussed herein.

This has been one of the most widely covered schemes in Computist and this is because the modified DOS scheme has to be the most popular protection scheme ever devised by software companies to deter the copying of software. It is also, usually, one of the easiest schemes to krack, with the correct hardware, software, and knowledge. Once you know how to krack this type of protection, you will be able to krack the majority of todays software, and you will know how to break the first line of protection on the schemes that use a combination of protection methods. Secondary methods of protection will be discussed in detail in a later section.

Before we discuss what needs to be done to defeat these protection schemes, we should first discuss the organization of DOS 3.3 disks. (Note: if I miss anything here, you can be sure to find it in a Computist back issue somewhere, or in your Beneath Apple DOS manual). Normal DOS 3.3 sectors are divided into two parts, an address field, and a data field. The address field serves as an identification marker for each sector. The data field contains the actual data. An example of an address field might be:

```
FF FF FF FF D5 AA 96 FF FE
AA AA AA AA FF FE DE AA EB
FF FF FF FF
```

The FF's at the beginning and end are sync bytes. The D5 AA 96 sequence is the beginning address marker (also called prologue bytes). Then the volume number, track number, sector number, and checksum are listed. Then towards the end of this field are the ending address marks (also known as epilogue bytes). They are usually DE AA EB

The data field is formatted in a similar manner:

```
FF FF FF D5 AA AD -342 bytes
of data+checksum- DE AA EB
FF FF FF
```

The FF's again are sync bytes. The D5 AA AD are known as start of data markers. The DE AA EB are the ending data markers. You will note that they are the same as the ending address bytes.

If any of these important bytes are changed (except for the EB's at the end of both the address and data field markers) then a disk cannot be read (or copied) under normal DOS. Copiers will not know the correct information about each sector and will not find the beginning or end of the data on that sector. In order for a protected disk to use this nonstandard format, it must have a modified RWTS (read write/track sector) or its own loader that can read this format.

If the program has an RWTS then we can reset into the monitor, after it loads and examine the B700-BFFF range of memory. You can then compare the protected RWTS to a normal one by checking these important RWTS locations:

	Read	Write
start of address:	B955:D5	BC7A:D5
	B95F:AA	BC7F:AA
	B96A:96	BC84:96
end of address:	B991:DE	BCAE:DE
	B99B:AA	BCB3:AA
start of data	B8E7:D5	B853:D5
	B8F1:AA	B858:AA
	B8FC:AD	B85D:AD
end of data:	B935:DE	B89E:DE
	B93F:AA	B8A3:AA
address marker sync byte		BC60:FF
data marker sync byte		B83E:FF

If the program does not use an RWTS, but a loader (or if you cannot reset in) you can examine a track with a track bit editor, such as the one in Nibbles Away, and try to find the beginning of the sector. This can be difficult unless you are familiar with the organization of a DOS 3.3 sector. Then you can figure out what the prologue bytes, epilog bytes, and the data markers are. An example of this method will be given later in this article. Several utilities are available to aid in the conversion of modified DOS disks. Advanced Demuffin, Demuffin Plus, COPYB, Super IOB, and Advanced Demuffin Toolkit are just some of the utilities available for deprotectionists. I will go into detail on the use of two of these, although all are very useful.

If the program you are working on has a catalog, then you should try using Demuffin Plus, for the conversion to DOS 3.3. To see if there is a catalog (uncommon these days), boot the disk and reset into the monitor. From the monitor type A56EG (assuming the disk is operating with a close to normal disk). If you get a catalog then use Demuffin Plus. If not, try Advanced Demuffin. To use Demuffin Plus do the following:

1) boot protected disk
2) reset into the monitor
3) move RWTS to safe location (6000< B700.C000M)
4) boot slave (C600G)
5) BRUN Demuffin Plus
6) reset into the monitor
7) move RWTS back (B700< 6000.6900M)
8) 803G
9) convert files to DOS 3.3 diskette

To use Advanced Demuffin do the following:

1) boot disk
2) reset into the monitor
3) 6000<B700.C000M
4) boot slave
5) BRUN Advanced Demuffin
6) exit to the monitor
7) B700<6000.6900M
8) 801G
9) convert disk

If you cannot get the protected RWTS intact but you do know what has been changed (from examining a track with the track bit editor such as Copy][+ or Locksmith), it is possible to modify standard RWTS and use Advanced Demuffin to convert. Doing this would put the disk into COPYA format, but you would need to modify the protected RWTS where it resides on the disk (track 0-2) or the loader (if applicable) to allow it to read from the now DOS 3.3 compatible disk.

However there is sometimes an easier approach to deprotecting slightly modified DOS protection schemes. (This one appears ALOT in Computist) You can simply change the I.O. flag in standard RWTS and copy with COPYA. Then it may be necessary to make a few changes on your new copy. Try this if you have a disk with a modified DOS:

1) RUN COPYA
2) ctrl C
3) 70 (to keep the program from reloading copy.obj0)
4) CALL-151
5) B942:18 (clear the i.o. flag)
6) 3D0G (basic entry point)
7) RUN
8) after the copy process is complete, correct any RWTS changes on your new copy. For example, if the protectors changed B955 in the RWTS to D4 instead of its normal D5, you would modify track 0, sector 3, byte 55 to D5 on the new copy (where the D5 normally resides on disk).
9) instead of modifying those locations in step 9 you might try putting an 18 on track 0, sector 3, byte 42. This permanently clears the I.O. error flag in the RWTS on your new copy.

If this method does not work, then you might try adding the following with B942:18 when you try COPYA:

B925:18 60 *kills end of address*
B988:18 60 *kills end of data marks*

BE48:18 *clears errors*

The B942:18 change can be utilized with FID (for files), Demuffin Plus, and Advanced Demuffin. Using Advanced Demuffin in the following way is even better than COPYA with the changes:

1) BRUN Advanced Demuffin
2) exit to the monitor
3) B942:18
4) 801G
5) convert

You may not even need to make further changes once the disk is converted. But making a COPYA copy of a protected disk may not be even enough. Sometimes tracks are left completely unformatted (Advanced Demuffin will copy around these but COPYA will not) and there are often secondary protection schemes. Sometimes software companies use half and quarter tracks or even non standard encoding schemes to screw up copiers. Some of these things will be dealt with in next section on nibble counts and secondary protection schemes.

The latest in modified DOS routines seems to be one used recently by Broderbund, the one of '18-sector' tracks which are actually 6 sector tracks, each sector containing 3 pages. The reason that 18 pages of information can be stored on a track (instead of DOS 3.3's 16 sectors) is that their are only 6 sectors. The lowered amount of sectors eliminates much of the sector overhead such as sync bytes and data markers. It seems that the data and address markers have been compacted and most of the sync bytes have been removed, making it much easier to store the 18 pages of data. Watch for more on this kind of protection in upcoming Computists as it is becoming more popular with Broderbund and possibly Electronic Arts.

Nibble Counts and Checksums

For some reason the terms nibblecount and checksum strike fear into the hearts of beginning deprotectors. This is as it should be, as the use of such methods of protection, generally indicates that the protectors are much more clever than the those who just modify the DOS a little bit. A nibblecount is a protection routine that reads in a track, or maybe just a sector, or more likely, the sensitive area between the tracks and compares the bytes read in to their original values. For some reason (usually due to the relative unreliability of disk drives in general) these bytes have been changed in the copying. If bytes have been changed, then the program crashes or reboots. Checksum is a term that is used to refer to a routine that makes sure the nibblecount is not tampered with. These frequently read in every byte in a sector, add up the total value of the bytes and if it is not what it should be, it will crash.

In order to read bytes in off of a disk, it is necessary to access certain softswitches that control the disk drive. These are our link to finding nibblecounting routines and (hopefully) removing them. To access the disk drive a program needs to access one of the following:

C08C
C089

or

C0EC (C08C+60 for slot 6)
C0E9 (C089+60 for slot 6)

Now most protectors do not bother to write self modifying code and this is often their downfall. You can often just scan for these bytes on your COPYA version of the program with a disk search utility. Make a note of each place where these are found. Remember, you are going to have to look through many disk accesses on each disk to find the protection code. When you do your searching, remember to search in backwards order. For example, if you want to find occurrences of C08C you would search for 8C C0! Addresses are stored backwards in machine language. Then, once you find your occurrences, you need to get out a sector editor and examine each, to figure out which is the protection routine, and which ones are not.

On tracks 0-2 you will probably find the most disk accesses. This is because normal DOS is located there, and most modified ones are too. It is possible (and quite often done) to hide protection code here. Look for routines there. Look for routines that directly compare values, and branch upon the results. However keep in mind that DOS routines will compare D5 then AA then 96 (also it compares DE AA and D5 AA AD). If a routine does this and then compares several others too, be suspicious!

For example, look at the following code on track $01, sector $0F of Seven Cities of Gold (by Electronic Arts).

```
0000:4C 69 05   JMP  $0569   this sets up EOA's
                             protection
0003:A0 20      LDY  #$20
0005:88         DEY
0006:F0 58      BEQ  $0060
0008:AD EC C0   LDA  $C0EC   read a byte
                             (C08C+60=C0EC)
000B:10 FB      BPI  $0008
000D:49 D5      EOR  #$D5    check for D5 (first byte
                             of the
000F:D0 F4      BNE  $0005   start of data mark)
0011:AD EC C0   LDA  $C0EC   read a byte
0014:10 FB      BPL  $0011
0016:C9 AA      CMP  #$AA    check for second byte of
                             start
0018:D0 F3      BNE  $000D   of data marker
001A:AD EC C0   LDA  $C0EC   read a byte
001D:10 FB      BPL  $001A
001F:C9 AD      CMP  #$AD    check for third byte of
                             marker
0021:D0 EA      BNE  $000D
0023:48         PHA
0024:68         PLA
0025:A0 56      LDY  #$56
0027:AD EC C0   LDA  $C0EC   read a byte
002A:10 FB      BPL  $0027
002C:2C 00 C0   BIT  $C000
002F:C9 B4      CMP  #$B4    look for $B4
0031:D0 31      BNE  $0064   if not where it should be,
                             crash
0033:88         DEY
0034:D0 F1      BNE  $0027
0036:A0 00      LDY  ##00
0038:AD EC C0   LDA  $C0EC   read a byte
003B:10 FB      BPL  $0038
003D:2C 00 C0   BIT  $C000
0040:C9 B4      CMP  #$B4    look for $B4
0042:D0 20      BNE  $0064   crash if not found
0044:88         DEY
0045:D0 F1      BNE  $0038
0047:AC EC C0   LDY  $C0EC   read a byte
004A:10 FB      BPL  $0047
004C:48         PHA
004D:68         PLA
004E:AD EC C0   LDA  $C0EC   read a byte
0051:10 FB      BPL  $004E
0053:C9 DE      CMP  #$DE    look for first byte of end
                             of the
0055:D0 09      BNE  $0060   data marker
0057:AD EC C0   LDA  $C0EC   read a byte
005A:10 FB      BPL  $0057
005C:C9 AA      CMP  #$AA    look for second byte of
                             ending
005E:F0 02      BEQ  $0062   marker
0060:38         SEC          set carry flag
0061:60         RTS          return to the caller
                             routine
```

(Note-this is a disassembly from Nibbles Away II sector editor).

Notice that in the above example, D5 AA AD are checked, as in a normal RWTS, and then it checks for $B4 in two places. If these are not found, then the routine branches to $0064 (where it will clear memory, and reboot the disk). Afterwards, the closing bytes of DE AA are checked, the carry flag is set, and an RTS ends the routine.

Now that we know what to look for, what do we do with them? The usual approach, is to end the routine prematurely, and continuing with the program. Look through the suspicious routine, and see how it ends. If it ends with a JMP, then try moving that JMP to the beginning of the routine, and see if it will now boot properly. If it ends in an RTS, then move it up to the beginning. Sometimes it will be necessary to clear the carry flag before doing this because it may be set

during the count. If this is the case, then put an 18 60 at the beginning.

In the example above you would put an 18 60 at $0000 (this is where the routine begins). But some nibblecounts have checksum routines to see that you do not mess with them. You will remember that we put an 18 60 at $0000. We replaced a 4C and a 69. If you add 4C and 69 and the next byte of A0 and subtract the total from 18 + 60 you will get the value of DD. This is the value that we need to replace the third byte on the sector with (A0). Once you do this, the checksum will find the correct total for the sector, and that nibblecount will be taken care of!

But what if we came up empty searching for disk access bytes? Perhaps there is self modifying code. One method of finding a nibblecount routine, is to boot up your original and when the count is being executed, do an NMI. You can then examine the stack, and see where in memory the count is at. You can then look at the routine in memory, search for the bytes that begin the routine, and then disable it.

The above method is occasionally useful, but not always. The best way is to boot trace the game and find the JSR or JMP to the read routine. You could trace on your copy (non-functioning copy, because of the check) and the last couple of JSR's and JMP's before the program crashes, would be suspicious. Try replacing these with EA EA EA (separately) and boot the game. If it works, then you are done, if not, keep trying.

If you have the game into files, then find out which is the boot file and examine it. Copy //+ will tell you which one it is, if you do not already know. Then look for routines in it that are suspicious (with disk access).

Up until now (in this article) I have referred to any disk checking protection, as being a nibblecount. In reality there are many types of disk checking methods. The best known, of course, is the nibblecount. But often times the protection routine is not a nibblecount. A true nibblecount, will read in and count the number of bytes on a given track. The protector will know how many were on the original's track. This method works, because the speed of disk drives vary, and bytes between sectors are lost and gained, through the copying process.

Another method is synchronized tracks. This is also based upon the timing of your disk drive. Many fast copiers do not worry about timing between tracks, and this is what is their undoing. However, most good bit copiers out today have a synchronize tracks option to deal with this.

Another method of disk checking, is to make special sync bytes with a couple of extra bits. This can only be reproduced with a special disk drive. The routine will obviously find that these two bits have been chopped off by a copier.

The next method is called Track Imaging. Basically, it reads data that is placed between tracks and if it is not where it should be, it will not work. This is based upon the fact that Apple disk drives are severely limited, and that data written less than one full track away from another, will bleed onto the adjacent track (when a copy is made).

Finding disk checking routines can be difficult at times, and easy at others. These methods above are definitely not the only ways to find them, and will not even work on every routine (I can guarantee that they will not always work), but they should provide you with enough background that you can figure out ways to find just about any disk check routine. The EOA game that I discussed earlier is an example of the Track Imaging protection scheme. Notice, however, that it was still found in the same manner as a normal nibblecount would be. You will just need to get some experience at cracking these kinds of protection schemes. With time, you can learn what appears to be something suspicious, and remove it.

Deprotection with a IIe/IIc

Having a IIe has traditionally been an obstacle to beginners at this game. It is a little more difficult, but not impossible, to get a good Krackrom for the IIe or IIc. Now there are several good ones on the market. This section is basically, just going to point out a few of the alternatives people without Krackroms, can use to their benefit.

The easiest way to do single load kracking single load games without a Krackrom is to NMI them (wildcard, replay, etc...), but as I stated earlier, the resulting files often take up too much space and they aren't always reliable. Your next alternative, is to learn how to boot trace. This technique will be discussed in a later section, but it can be done on any Apple (II+, IIe, IIc).

The modified dos protection schemes can be attacked by using COPYA with the I.O. flag changed, often. If this does not work you can try using the anti-reboot disk (from Computist). Some, older programs, let you get into basic with just a CTRL-C (unheard of these days). If none of these are useful, you can do a nibble read of a track and figure out the address and data markers and then do a Demuffin of the disk. Secondary protection schemes can be found in the usual ways.

Basically, IIe and IIc kracking just takes a little more creativity and improvisation on the part of the kracker. You can only use those tools available to you. If you do have a way to get a Krackrom, then do it, or else you will just have to learn to do without.

Boot Tracing

Boot tracing is a difficult art to master, but everyone will (most likely) find a problem that only boot tracing can solve. So it is essential that you know at least something about this art, if you want to be able to crack some of the most difficult protection schemes.

Basically, boot tracing can take several forms. There are many different "styles" of the technique. But each style is an extreme test of your knowledge of assembly language and no method is especially easier than another. I will show you a few of the different styles, and I will give you one short example of a crack using one of these styles.

When you turn on your Apple (with the Autostart ROM) your disk drive controller reads in track 0 sector 0 into 800-8FF of RAM. Basically, your controller ROM is from C600-C700. This is usually referred to as boot 0. 800-8FF is the boot 1 or the first bootstrap loader. From here the path divides. On normal DOS 3.3 disks, the first stage will jump to the B700 (RWTS) range and protected ones may jump elsewhere. The object of boot tracing is follow the stages of the boot process, stopping after each is loaded in, until the whole program is loaded (if it is a single load), or RWTS is loaded (if it is a modified DOS), or until you find a nibble count.

So how do we do this tracing? We must make modifications to each boot stage (in RAM) so that each will stop after reading in from the disk, and end up in the monitor. This is called setting break points. This is where the styles differ. Each has its own way of doing this. The first method to be described was well documented by Mr. Xerox in his articles on Pirates Harbor. This (and all of the methods that I will discuss) starts with the following:

Turn on your Apple and reset. Then type:

```
CALL -151
9600<C600.C700M   moves controller ROM code into RAM, to be modified
```

Then you must set your first break point:

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

And start the program's boot:

```
9600G
```

It should boot, read in track 0, sector 0 (into 800-8FF), and break into the monitor. The drive will still be spinning, so to turn it off, type:

```
C0E8
```

Now you have the first stage loaded in to examine. Your first task is to find out how the stage jumps to the next (It will most likely jump to at least one more stage. Normal DOS jumps indirectly to B700 from here). You must then move 801-8FF somewhere safe, so it is not overwritten by the next boot (remember that when you boot a disk it will read in track 0, sector 0 onto 800-8FF, and since we have changes to make there, it is not desirable to have these overwritten). It is a good idea to move it to a page of memory ending with an 8 such as 8800 or 9800. So move 801-900 to 8800 by typing:

```
8801<801.900M
```

We then need to change 96F8 to jump to 9801 instead of breaking into the monitor, so it will read the next stage. After we read in the next stage, we will again want to get into the monitor. When you examined 801-900 you should have found the jump to the next stage. You then need to set a break point here as well. Then you again would type 9600G to boot the disk. The program should load a little more and then pop into the monitor. If it does not, then you did not find the correct jump. Once you believe that you have the program correctly loaded, or have the RWTS intact, or have found the nibblecount, then you can save it out as a file, Demuffin the game, or disable the count.

Another style of boot tracing just allows you to change one byte to keep the next boot from overwriting the last. You would do the following commands as normal:

```
9600<C600.C700M
96F8:4C 59 FF
9600G
```

and

```
C0E8
```

You then examine the 801-900 range as normal, and attempt to figure out the jump to the next stage. Once satisfied, you do the following (lets assume you found that the jump out was a 4C 01 03 at 841):

```
841:4C 59 ff
9659:28   This is where we differ from other methods. It sees to it that our mods will not be written over
96F8:4C 01 08
9600G
```

and

```
C0E8
```

You then continue as normal, but you also need to figure out what byte will tell the boot stage not to overwrite your previous changes.

The last method of boot tracing that I will mention was documented by Mycroft in one of the original issues of Hardcore Computist. The article is still available in the Best of Hardcore.

Again you must copy C600 into 9600:

```
9600<C600.C700M
```

Now you will set 96FA so that the stage will jump to 9801:

```
96FA:98
```

You then need to put a break point at 9801:

```
9801:4C 59 ff
```

and

```
9600G
```

and

```
C0E8
```

Then you will need to examine boot 1. Let's use the example above and assume that there is a jump at 841 to 301. You would need to do the following:

```
9801<801.900M
9843:93   set the jump to 9301
9301:4C 59 FF
```

And then start up with the 9600G.

Now that we have looked at each of the styles of setting break points, I want to give you a short example of boot tracing an actual game. A while back I softkeyed a lame adventure game called Lava Pits of Asnar by Sanctum Software. Basically it is just a modified DOS. Here is the description of the krack:

Softkey for...

Lava Pits of Asnar

Sanctum Software

1. Turn on your Apple and RESET.
2. Enter the monitor if you're not already there.

```
CALL -151
9600<C600.C700M
96F8:4C 59 FF
9600G
C0E8
```

3. If you list $801 (800L) you will see some abnormal code (that does not appear to be anywhere similar to the normal code at track 0, sector 0 of a normal DOS 3.3 disk). It however has a jmp to B700 (direct!) at $83C. Then do the following:

```
9801<801.900M
983C:4C 59 FF
96F8:4C 01 98   jump to our modified stage 1
9600G
C0E8
```

4. And list B700. You now have their RWTS intact in memory. Now just do the following, to complete the krack.

```
6000<B700.C000M   move RWTS to a safe place in memory
```

5. Boot DOS and:

```
BRUN ADVANCED DEMUFFIN
```

6. Exit to the monitor.
7. Move RWTS back.

```
B700<6000.6900M
801G   restart Demuffin
```

8. Convert tracks 3-22
9. Copy a normal DOS onto tracks 0-2 of your new copy.

It is now softkeyed. There is obviously a more to boot tracing than i have discussed so far. It is not always so easy to figure out the jump to the next stage of the boot. Sometimes there are indirect jumps which need to be figured out. Occasionally you will need to modify a few bytes to keep a stage from crashing. This is where your knowledge of assembly comes in. You must be able to figure out the code in each stage to continue the trace. If you want to learn more on the subject of boot tracing, then I suggest that you read the excellent articles on the subject, written by Mr. Xerox, and if possible, the one by Mycroft.

Pascal and ProDOS

With the recent wave of ProDOS based programs, comes another aspect of deprotection. Also along a similar line, is the protection scheme of programs that are written in Pascal (P-code or Runtime). For the most part, ProDOS and Pascal programs cannot use wildly modified formats. Therefore we generally do not have to worry about making these programs into COPYA format. We also do not need to worry about strange tracking increments (although every rule has its exceptions).

In Pascal and ProDOS, nibblecounts and other disk checking routines are the popular methods of protection. You need only search for it in the same manner described earlier. Besides using self-modifying code, there is not much else to do to protect Pascal programs. ProDOS can also have protection routines hidden inside of itself. You can find out whether the ProDOS has been changed, by doing the following:

1. Boot a standard ProDOS disk
2. get into Basic
3. catalog protected disk
4. type prefix,d1
5. run the first system file in the catalog

If it runs, then the protected program, has its protection hidden inside of ProDOS. This can be fixed by simply copying the files over to normal ProDOS. If the nibblecount is not in ProDOS, then it is on one of the other files on the disk. It can be found and removed as normal. However, if the program reboots almost immediately after loading ProDOS, you might check out the first system file. This will load in at $2000.

Basically that is about it for ProDOS and Pascal protection schemes. Not too many other methods of protection are used. Pascal games (such as Wizardry and Sundog) all do some sort of disk check. But beware, software companies will surely come up with new methods of protection for these, in the future.

Note: lately, several companies have begun using an 18-sector format of ProDOS which has proven difficult to crack. Broderbund is now using this format on their releases.

Using The NMI to Deprotect

If you have owned your Apple very long, then you surely have heard about (or own one of) those miraculous cracking cards, that do everything for you at the press of a button. These are based upon a function of the Apple, called a non maskable interrupt. This means that nothing (almost) can prevent it from stopping the program operating on your Apple.

To generate one, all you need to do is cross pin 26 and 29 with a 100 ohm register. This is not extremely practical to try to do manually, because they occur so quickly, that they can even interrupt each other. This will dump garbage all over the stack, and make this device worth very little.

When you use an NMI the 6502 pushes the location of where the program is running onto the stack and then jumps to where ever pointed to at $FFFA and $FFFB. To restart the program we need to restore the registers and the stack, and do an RTI (return from interrupt). Most NMI cards are designed so that the stack is preserved on interrupt.

There are very few uses for the NMI that I condone. One, is for finding nibblecounts. When one is being executed, you can do an NMI and find out where it is located in memory by checking the bytes on the stack. You should never need to use one for cracking single loading games. This makes the file way bigger than it needs to be, and the program may not work every time you run it. However, if it is your last resort (and nothing else works) it is (in rare circumstances) acceptable. For instance, if you cannot find a nibblecount, you can do an NMI krack of the program once loaded and after the count. This will often allow you to BRUN your Bfile, and stick your COPYA copy of the game in to work. However this will make your game much longer than it needs to be (maybe even a file and a side,instead of just one side). But this should only be used as a last resort.

Encoding Data through 4 & 4 Nibbilizing

Before we delve into the deep dark secrets of 4 & 4 encoding, we should take time out to discuss all of the encoding methods in general. Standard DOS 3.3 and ProDOS both use what is called 6 & 2 encoding, and DOS 3.2 uses 5 & 3 encoding.

So what does all of this mean? When the Apple disk drive was first introduced, it used a different method of recording data than other disk drives. This helped the Apple become popular, but it limits the Apple's ability to read all of the possible values for bytes (there are 256 of them) from the disk. Only bytes that meet certain restriction can be stored on a disk. These requirements are:

1) The high bit must be a one (1)
2) The byte can only contain one pair of adjacent zeros
3) There can only be two consecutive zero bits

There are only 64 bytes that meet these requirements. In order for the Apple disk drive to find a value that does not fall between one of these 64 bytes (but that is one of the possible 256 bytes that the CPU can understand), some sort of nibblizing has to be done by the loader or RWTS. This nibblizing is a process by where two bytes can be used to come up with one value acceptable to the Apple CPU.

DOS 3.3 and ProDOS both use a nibblizing process known as 6 & 2 encoding. Basically, the first byte contains 6 bits and the second contains two bits. In DOS 3.3, this nibblizing process is done in two tables. The first of these is located at BB00. The extra two bits are chopped off and stored in the second table. They are replaced by zeros in the first table. For example:

Original byte	New one
(01111111) 7F	3F (00111111)

But the extra two bits are stored in the second table, at BC00. Basically, the other methods of nibblizing, do what their titles indicate. DOS 3.2 uses 5 & 3 nibblizing. One table stores 5 bits, the other stores 3 bits. 4 & 4 encoding stores bytes in two halves.

Note, however, that 6 & 2 encoding allows you to have 16 sectors per track, because it only takes 342 disk bytes to reconstruct one sector. 5 & 3 encoding allows you to have 13 sectors per disk, because it requires 410 disk bytes to reconstruct a sector. 4 & 4 encoding uses 512 bytes to reconstruct a sector, therefore you can have 10 sectors (or less) per track.

So why would anyone want to use any disk encoding scheme other than 6 & 2? First of all, 4 & 4 is an easier method of encoding, if you are into writing your own loaders. Also, it allows you to load things at a much faster rate. Of course, if you are worried about storage space, it is definitely not a desirable encoding scheme, but for most games, space is not usually important (especially arcade games). Also, if you do not have your program in files (i.e. you make your game have no catalog, but instead have your loader read directly) then cracking it is extremely difficult. There are not (to my knowledge) any conversion programs, that work with this kind of encoding. One reason for this, may be the fact that 10 sectors or less may be used per track. Because we do not know in advance, how many will be used, how can we write a converter? Also, we do not know how the protector has written his loader (or RWTS). Most converter programs assume that the RWTS will be close to normal, you cannot assume this with 4 & 4 encoding, because if the author went to the trouble of writing his loader, chances are that he will make sure it is completely different.

So, how do we go about kracking these types of programs? Well as I stated earlier, usually only games are involved with this kind of encoding scheme. Many of these are just single load cracks. The tough ones, are the ones that have other disk access (for levels). To crack them you must:

1) Find and isolate the loader and figure out the logic of it.

2) Modify the loader to let you read in one level at a time, and save it out with Inspector (or the equivalent) under normal 16 sector format.

3) Rebuild the program to work under normal dos. This will consist of modifying the loader and the main program.

That is really all there is to say about cracking these types of programs, except to mention that these are often the hardest ones to krack, although this will become painfully obvious to anyone who tangles with this little beastie. Unfortunately, this is one scheme that is definitely NOT for beginners. This one takes a lot of time, effort, skill and experience with breaking copy protection schemes. And some of these also have secondary protection schemes as well (as if the encoding was not enough).

Cracking Protection Schemes in BASIC

Believe it or not, protection schemes in BASIC, can often be difficult to thwart. This article will show you a few of the things to look for in BASIC programs.

The first problem you may encounter, is a difficulty in listing a program. If you try to, and it runs instead of listing, then reset into the monitor. Then check location D6. If it is not zero, then change it to zero. This is the Applesoft auto run flag. If the flag is set (other than zero) any command entered will cause whatever program is in memory to execute. If it lists part of the program, then executes a DOS command instead, then they may have hidden a dos command inside of a normal statement. This can be done by putting a CTRL-D on the 34th character of a line and then putting a DOS command. For example:

```
10 REM YOU CANNOT LIST THIS!!!!!!*
   CATALOG
```

*(The * represents the ctrl D)*

If when you try to list, the computer just lists one line number, over and over again, the authors may have poked a 1 into location 2049. In order to stop this, reset before it runs, poke a zero there, and then list. You can then delete the line in the program that does the poke. Occasionally, protectors will do peeks into DOS, to see if the address marks or data marks have been changed. Just look for peeks over 40192. Sometimes DOS commands are changed, and basic programs may try and use the abnormal ones. When you do a conversion to normal DOS, the normal DOS commands are the only ones available. Thus the program will crash. You may need to go through all of the basic programs on a disk, to make sure that only normal DOS commands are called.

Sometimes, nibblecounting routines may be called from basic. The program may load a binary file, call to it, and if everything is right, it will return control to the basic program. These are just a few of the possibilities of basic protection schemes. To find out what is wrong, you will just have to scan through all of the basic programs, looking for suspicious peeks, pokes, and calls (and maybe weird DOS commands too). One programs that use mostly basic, this can be a headache, but it is just about the only way.

Half- and Quarter-Tracking

Apple disks have data written in 35 concentric circles known as tracks. This is common knowledge. But these tracks are written approximately 1/48th of an inch apart. It is possible to write closer than this, but it will most likely overwrite some of the data on the adjacent track (s). However the apple disk drive's head can be positioned over 1/96th of an inch increments. Because older standard copiers and fast-copiers copy only "Full-tracks", and not these "Half-tracks", it was sometimes used as a protection scheme on older software. However, because most bit copiers come with an option to duplicate half and quarter tracks, this is an obsolete and uncommon scheme these days.

If you have a Track Star, a Know Drive, or if you strip the outer casing off of your disk II, you can see when these Half-tracks are in use.

As I said, this is not really very common anymore. The basic approach to cracking a game with this protection scheme is:

1) Find out which Half-tracks are used (Games do not use every half- track, because at least the first track must be readable under normal DOS. It was also easier to confuse older copiers by using only part of the program on Half-tracks).

2) Read in the Half-tracks with a track bit editor, such as Nibbles Away, and write them to your new copy on the Full-tracks.

3) Fix the RWTS to read only Full-tracks

To find out which tracks are used, you can use the methods listed earlier. To use Nibbles Away for moving the tracks, just read in the Half-tracks type 'z' to allow Nibbles Away to analyze what it has read in, and write out to a Full-track. For example, if track 17.5 is being used, then read it in, analyze, and then write out to track 17.

The final problem, is that the foreign RWTS will be looking for data on Half-tracks, and not finding it there. You need to tell their RWTS that the tracks are now on Full-tracks. To find the code that you need to fix, you need to understand how normal RWTS moves the disk drive head for reading and writing. There are several softswitches that need to be accessed for any use of the disk drive. See table 1.

Normal DOS moves the head from track zero to track 1 in the following manner:

a) turn on phase 1 (track .5) and wait.
b) turn off phase 0 (track 0) and wait.
c) turn on phase 2 (track 1) and wait.
d) turn off phase 1 (track .5) and wait.
e) turn off phase 2 (track 1) and wait.
f) return to the caller.

By examining the above example, you can see what is needed to move the head to Half-tracks. It will be your task, to fix the protected disks RWTS to read the Full-tracks instead of the Half-tracks.

So much for the Half-tracks, lets move on to the second part of this article. Quarter-tracking works on similar principles, but is often used in totally different ways than its counterpart. As the name indicates, Quarter- tracks reside exactly 1/4 of the way to the next Full-track. Because of the limitations of the Apple disk drive, it is not easy to use Quarter-tracks successfully, and extremely difficult to use with the //c. There are a few laws that apply to the use of Quarter-tracks:

a) When tracks are written, they will produce exact images of themselves on adjacent Quarter-tracks but the image on the adjacent Half-tracks will not be an exact image.

b) When a track is written it will not (usually) effect data on tracks at least 3/4 of a track away.

It is possible to write (once) to adjacent Half-tracks. Suppose that we we write out data to track 7, then to 7.75, and to 8.25. Images would appear on 7.25, 7.5, 8.0 and 8.5. Now if you remember rule (a), then you can see, that when someone copies the disk, the images on everything but the adjacent Half-track, will come out perfectly. But the data on that Half-track (7.5) will be questionable (at best). If it is read by the protected disk, the data probably will not be found. Several companies use this scheme, or something similar. Electronic arts and Broderbund, (and others) did this, up until recently. Microprose used something similar (especially on F-15 Strike Eagle; long time Computist readers will recall what a headache that one caused). This scheme is called "Track Imaging".

Another technique of using odd track spacing, is called "Track Arcing" or "Spiral Tracking". This technique consists of putting about 1/3 of a tracks worth of data on one track, and one third on the adjacent Half-track, and so on. If valid data is written to a track, it will zap the adjacent Half-track (therefore killing one third of a tracks data). Note also that it is possible to use quarter track increments for track arcing. The possi-

Table 1

Softswitch	Label	What it does
C080	phase off	stepper motor 0 phase off
C081	phase on	stepper motor 0 phase on
C082	phase 1 off	stepper motor 1 phase off
C083	phase 1 on	stepper motor 1 phase on
C084	phase 2 off	stepper motor 2 phase off
C085	phase 2 on	stepper motor 2 phase on
C086	phase 3 off	stepper motor 3 phase off
C087	phase 3 on	stepper motor 3 phase on
C088	motor off	turns off disk drive
C089	motor on	turns on disk drive
C08A	drive 0 on	engages drive 0
C08B	drive 1 on	engages drive 1
C08C	Q6L	strobe data latch for I/O. Reads a byte.
C08D	Q6H	load data latch.senses write protect.
C08E	Q7L	prepare latch for input. writes a byte.
C08F	Q7H	prepare latch for for output.loads write latch.

bilities of anyone doing this are astronomical, but it has been done at least once.

If the apple disk drive can only increment in Half-tracks, how then can it position its head over a Quarter-track? To do this, you must turn on the stepper magnets on both sides of a Half-track. So, to use track 6.25, you would position the magnets to be over 6.0 and 6.5. You would then turn the magnets off and leave the head in between the two Half-tracks. So to use track 6.25 you would (moving from track 6):

a) Turn on phase 0 (remember: there are only four phases) and wait
b) Turn on phase 1 and wait
c) Turn off phase 1
d) Turn off phase 0
e) Return to caller routine

Well I realize that this mess is all very confusing at first, so I would suggest you read chapter 6 of "Beneath Apple DOS", for more information on direct usage of the disk drive. This is an extremely important chapter for learning about copy protection.

Self-Modifying Code, EOR and Disk Encryption

Often times software protectors have something that they do not want you to see. A few of them actually do something about it. There are a few ways to make it more difficult to find something such as a nibble-count. Until recently, very few software companies actually went to the trouble of writing this kind of code (to hide something), but you may encounter it, so I will speak briefly about some of the methods to look for. Most methods of hiding code are called "Self-Modifying code".

Some companies use the Exclusive-Or operator in a diabolical way to hide code (this was discussed in a back issue of Computist in greater detail). Some tracks on a disk, may appear to contain garbage, but after they are read in, and EOR'd, the garbage transforms into real code.

Another method of Self Modifying code, uses the INC and DEC operators to do their dirty work. This method is well documented by Krackowitz a long time ago. The company discussed was Sierra Online and the game was Cannonball Blitz (to give you an idea about how old this was). Rather than reprint all of this material here, I will sum up what he said in a little piece of the code:

```
59E4:CE E7 59   DEC $59E7    notice that this
                             decrements the
59E7:CF         ???          CE at $59E7, which is
                             the next
59E8:EA         NOP          instruction!
59E9:59 EF EA   EOR $EAEF,Y
59EC:59 AD 51   EOR $51AD,Y
59EF:C0 AD      CPY #$AD
59F1:54         ???
```

If you execute the first instruction and relist the code, you will notice a slight change:

```
59E4:CE E7 59   DEC $59E7
59E7:CE EA 59   DEC $59EA    the next instruction!
59EA:EF         ???
59EB:EA         NOP
59EC:59 AD 51   EOR $51AD,Y
59EF:C0 AD      CPY#$AD
```

If you execute the second instruction is executed, you will see the whole picture. You can see exactly what it is that they do not want you to see.

```
59E4:CE E7 59   DEC $59E7
59E7:CE EA 59   DEC $59EA
59EA:EE EA 59   INC $59EA
59ED:AD 51 C0   LDA $C051
59F0:AD 54 C0   LDA $C054
59F3:AD 57 C0   LDA $C057
```

You can plainly see now that they did not want you to know where the graphics were being set up. This is because you can then determine where the starting location is (remember back to the single load section?)

The key to figuring out this kind of code hiding, is to know your assembly inside out, and to have some patience.

VTOC and the Catalog Track

In cracking protection schemes, you will sometimes need to know something about the catalog and VTOC of a disk. If you type A56EG from the monitor, it will show you the contents of the catalog (unless DOS has been tampered with). Normally, the catalog track is located on track $11 (17 in decimal). VTOC is the volume table of contents. On normal disks, this is located on track 11, sector 0. Some protected disks will relocate this. It is through this table, that DOS knows where to find files. DOS must also read from the VTOC, in order to know where the first sector of the catalog track is located. On normal (and many protected) disks the location of the catalog track will be stored at AC01. The value for the current track in use will be stored (usually) at $2E. The value for the current sector in use will (normally) be stored at $2D.

Now that we have some of the basic information about the catalog track, we can start discussing its part in disk protection schemes. Naturally, if you were a software protector, you would want to keep those "nasty pirates" from screwing around with the files on your disk. There are a few ways that you can keep some people out. The first is to do away with files altogether, and just write a loader to read data straight from the tracks. This is not always within the realm of possibility (it may just not be worth the trouble). You can also, just relocate the catalog track. This will not keep out the "real" softkey-ers, but it may keep out some people. Obviously, to fix this kind of protection you would need to:

a) find out where the catalog is
b) relocate what is stored on track 11 of your new copy
c) move the catalog to track 11
d) fix the DOS/loader to know that the data is no longer on track 11 and that the catalog is now there

Because moving the catalog, is not really a true protection scheme, I will not go over all of the details of the above steps. All of the necessary information is available in Beneath Apple Dos and Computist back issues. I will, however, point out a few other tricks that protectors use with catalogs. One of these tricks, is putting CTRL-H's after the filenames, so that when you catalog, no names are printed. Sometimes (old) software protectors will hide other control characters in a file name although this is no longer an effective scheme, to keep you from loading a file. Sometimes games even have hidden files in the catalog such as Broderbund's Lode Runner.

It is also possible to use deleted files. If you recall, Copy II+ has a function that undeletes files. This is because deleted files are still stored somewhere on the disk, and can be brought back from the grave. The catalog track is not usually important for the actual process of cracking a game. Microprose's F-15 Strike Eagle had a moved catalog track with a false catalog on track 11.

Happy Cracking!!!

FINDER

A Nibble Count Finder & Gray Byte Searcher

The nibble count finder/grey byte searcher is a rather convenient utility that will search a disk for drive access (the telltale mark of a nibble count) and/or search a disk for grey bytes, one of the more recent protection schemes for apple. For those who don't know, grey bytes are bytes that are different every time the sector(s) are read. A normal copier will of course read this byte a certain way and write as a "concrete" byte to the disk, thus turning it into a normal byte. The programs are self explanatory, prompting one for the tracks and sectors (in decimal) that one wishes to search.

Note that the program is not set up to discriminate between normal disk read routines (such as those that check for data and address markers) and actual nibble counts that are part of a protection scheme. Nor is the program designed to read a protected disk. This has been designed with the intention that the protected disk has already been converted to a normal-DOS format. If, for example, the protected disk (or copy) has altered markers, you will need to convert them to normal for this program to work. When this utility was written it was designed so that if you had a disk where all the other protections had been removed and you still weren't getting a working copy (suspected nibble count or signature check routine) then you could run this program and hopefully find suspicious disk-access routines.

The Finder also cannot find encrypted routines on disk. The only way around this is to decrypt the information first and then run the Finder.

One last note: if you abort the operation before the program is finished (with ctrl C), you may have problems with your operating system, so be careful. The operation will take some time so be patient. Happy hunting!

NIBBLE COUNT FINDER

```
0 ONERR GOTO 10
10 PRINT : PRINT CHR$ (4); "BLOAD◊
   UTILITY.OBJ,A$8000"
20 HOME : PRINT : PRINT
30 PRINT "◊◊◊◊◊◊◊NIBBLECOUNT/GREY◊
   BYTE◊FIND"
50 PRINT : PRINT "◊◊◊◊◊◊◊◊◊◊SOFT
   KEY◊PUBLISHING"
55 PRINT : PRINT : PRINT
60 PRINT : PRINT "◊◊◊FIND◊A:◊(N)
   IBBLE◊COUNT" : PRINT
70 PRINT "◊◊◊◊◊◊◊◊◊◊(G)REY◊BYTE"
90 PRINT : PRINT
100 PRINT "◊◊◊◊◊◊◊◊<◊>CHOICE"
110 VTAB 15: HTAB 11: GET A$:
   PRINT
120 IF A$ = "N" THEN HOME : PRINT
   : GOTO 150
130 IF A$ = "G" THEN HOME : PRINT
   : GOTO 400
140 GOTO 10
150 HOME : PRINT "◊◊◊◊◊◊◊◊◊NIBBLE◊
   COUNT◊FINDER"
160 PRINT : INPUT "◊STARTING◊TRA
   CK:" ;ST
180 INPUT "◊STARTING◊SECTOR:" ;SS
190 INPUT "◊ENDING◊TRACK:" ;ET
200 INPUT "◊ENDING◊SECTOR:" ;ES
210 FOR TR = ST TO ET
220 FOR SE = SS TO ES
230 POKE 2,TR: POKE 3,SE: POKE 4,1
240 CALL 32768: REM UTILITY.OBJ
260 FOR L = 36864 TO 37119
270 VA = PEEK (L)
280 IF FL = 1 AND VA = 192 THEN
   360
290 FL = 0
300 IF VA = 233 THEN FL = 1: NEXT
   L
305 IF VA = 137 THEN FL = 1: NEXT
   L
310 NEXT L
320 FL = 0
330 NEXT SE
340 NEXT TR
350 PRINT : PRINT "NIBBLE◊SEARCH◊
   DONE." ;: FOR I = 1 TO 2000:
   NEXT I: GOTO 10
360 PRINT : PRINT : PRINT : PRINT
   : PRINT
370 PRINT "◊SEQUENCE◊AT◊TRACK◊"
   ;TR; ",◊SECTOR◊" ;SE; "◊BYTE◊"
   ;(L - 36865)
380 NEXT L: GOTO 270
390 END
400 HOME : PRINT "◊◊◊◊◊◊◊◊◊GREY◊
   BYTE◊SEARCHER"
410 DIM X(255)
420 PRINT : PRINT
421 PRINT : INPUT "◊STARTING◊
   TRACK:" ;ST
422 INPUT "◊STARTING◊SECTOR:" ;SS
423 INPUT "◊ENDING◊TRACK:" ;ET
424 INPUT "◊ENDING◊SECTOR:" ;ES
426 FOR TR = ST TO ET
427 FOR SE = SS TO ES
428 POKE 2,TR: POKE 3,SE: POKE 4,1
429 CALL 32768
440 FOR B = 0 TO 254:X(B) = 36863
   + B: NEXT B
450 CALL 32768
460 FOR B = 0 TO 254: IF X(B) < >
   36863 + B THEN 480
470 NEXT B
471 NEXT SE
472 NEXT TR
475 PRINT : PRINT "GREY◊BYTE◊SE
   ARCH◊DONE." ;: FOR I = 1 TO
   2000: NEXT I: GOTO 10
480 PRINT "◊SEQUENCE◊AT◊TRACK◊"
   ;TR; ",◊SECTOR◊" ;SE; ",◊BYTE◊"
   ;(B)
```

Checksums

```
  0-$9435   210-$0350   400-$7E07
 10-$ED91   220-$659E   410-$94E4
 20-$202C   230-$5205   420-$4BD5
 30-$9789   240-$8960   421-$DE43
 50-$7280   260-$895C   422-$48E8
 55-$C944   270-$80ED   423-$EB68
 60-$F9D0   280-$A117   424-$97BF
 70-$193B   290-$BC3D   426-$0E84
 90-$B432   300-$CF39   427-$264B
100-$3181   305-$69D1   428-$071C
110-$A80F   310-$1EA3   429-$0E11
120-$D36E   320-$BAC5   440-$EE14
130-$2778   330-$950A   450-$30C8
140-$1D1F   340-$CCDE   460-$8FF1
150-$F0A3   350-$D1EA   470-$BEB4
160-$4C56   360-$EA21   471-$F148
180-$9BD2   370-$BA0F   472-$A244
190-$0759   380-$3FCE   475-$6625
200-$D4F4   390-$9451   480-$1273
```

UTILITY.OBJ

```
8000:20 E3 03 84 00 85 01 A5  $C4C5
8008:02 A0 04 91 00 A5 03 C9  $29DE
8010:10 90 04 A9 00 85 03 A0  $5CE5
8018:05 91 00 A0 08 A9 00 91  $AF30
8020:00 C8 A9 90 91 00 A5 04  $C5A3
8028:A0 0C 91 00 A9 00 A0 03  $B0B4
8030:91 00 20 E3 03 20 D9 03  $7FAD
8038:A9 00 85 48 90 1B A9 87  $5758
8040:20 ED FD A9 C5 20 ED FD  $B45F
8048:A9 D2 20 ED FD A9 D2 20  $6F50
8050:ED FD A0 0D B1 00 20 DA  $ABE3
8058:FD 60 A9 D5 8D 53 B8 8D  $A8E1
8060:E7 B8 8D 55 B9 8D 7A BC  $8A12
8068:A9 AA 8D 58 B8 8D F1 B8  $19B2
8070:8D 5F B9 8D 7F BC A9 96  $82DC
8078:8D 6A B9 8D 85 BC A9 DE  $2986
8080:8D 9E B8 8D 35 B9 8D 91  $68F8
8088:B9 8D AE BC A9 AA 8D A3  $255D
8090:B8 8D 3F B9 8D 9B B9 8D  $A057
8098:B3 BC 60 A9 D5 8D 53 B8  $4C22
80A0:8D E7 B8 8D 55 B9 8D 7A  $57B7
80A8:BC A9 AA 8D 58 B8 8D F1  $0534
80B0:B8 8D 5F B9 8D 7F BC A9  $8C23
80B8:96 8D 6A B9 8D 85 BC A9  $026C
80C0:DE 8D 9E B8 8D 35 B9 8D  $AB80
80C8:91 B9 8D AE BC A9 AA 8D  $E7D7
80D0:A3 B8 8D 3F B9 8D 9B B9  $0A16
80D8:8D B3 BC 60              $8F98
```

BSAVE UTILITY.OBJ, A$8000, L$DC

C J Gammage — Australia

Ⓠ I'm having trouble with the softkey for "Explore Australia" (issue #76, pg. 17). I can CALL-151, no worries, I can enter 2421:60 and 2000G. But form there, all I get is a blue frame around a black screen and the machine refuses to accept any further keyboard entry. Is there another way to deprotect this program? Can anyone explain why my machine locks up after I do a "2000G"?

Tony Sandoval — TX

Ⓠ Does anyone have some tips for "Beyond Zork"? I'm having some problems and could use a little help.

Martin P. Longenberger — PA

Softkey for...

Robocop

Data East

Requirements:
COPYA
sector editor (with search)
blank disk

After reading William Rice's softkey for Batman by Data East (issue #74, p 16), I decided to see if the same protection scheme was used on Robocop by Data East; to my

surprise, it was. The protection was identical only shifted slightly in location but still on track $99, sector $01.

Step-by-step

1. Boot a DOS 3.3 system disk.
2. Tell DOS to ignore errors and use COPYA to copy both sides of the disk.

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

3. Search side one (1) of the copy for:

A0 00 BD 8C C0 10 FB 88 F0 C6 C9 E7 D0 F4 BD 8C C0 10 FB C9 E7 D0 B9 BD 8C C0 10 FB C9 E7 D0 B0 BD 8D C0

and replace it with:

A9 FC 85 F0 A9 EE 85 F1 A9 EE 85 F2 A9 FC 85 F3 A9 E7 85 F4 A9 EE 85 F5 A9 FC 85 F6 A9 E7 85 F7 4C 85 60.

Side two (2) requires no modification.

Playing Tip for...

Robocop

Data East

If you want to play the game without running out of time and also getting a periodic recharge while playing, then proceed as follows:

1. When the first game screen appears put the game in pause.

```
ctrl [
```

2. Tell the game to enter unlimited play mode.

```
ctrl ]
```

Initially your power level will drop, but as you play it will fluctuate up and down without ever reaching zero.

Scott Jelsma — IA

Does anyone know how to deprotect "The Teacher's Tool Kit" (v3.1 by Hi-Tech of Santa Cruz) on a 3.5" or 5.25" disk.

Does anyone know how to alter Apple's Fast Copy program (v1.0) so it will automatically use all available memory without having to setup a RAM disk?

Does anyone know how to deprotect "Miners Cave" (v1.0) by MECC on 3.5" disk?

Has anyone heard any rumors when Claris is going to fix the bugs in Appleworks 3.0? I have found a bug and even Beagle Bros patch program doesn't fix it. I contacted Claris and they would not comment as to when or if a fix will be available. If anyone is interested in making a patch to fix this bug I would greatly appreciate it.

To encounter the bug follow these steps:

1. Load the file "MISSPELLED" from the subdirectory "SAMPLE.FILES" on your master disk.
2. Put the first top margin as 2.0" and the right, left and bottom margins as 1.0".
3. Change the characters per inch to 10.
4. Calculate the page break. You will notice that the page break is between:

Pneumonia
by Sarah
———End of Page 1———
I had the flu and I woke up screaming with a nightmare. I

Now add the second top margin of 0.2 inches. I did right after the second line which says "Teachers' Idea & Information Exchange". Notice that the page break is at:

were on the other side of the curtin. My temperature was
———End of Page 1——
taken and I got a shot. The nurse said I had pneumonia. I

The page break for end of page 1 has moved down 9 lines since the top margin of 0.2 inches was added!

I use a IIgs with 1.25M of memory, one 3.5" and one 5.25" drive. My printer is an Imagewriter II.

Brian Plautz — IL

Help! I would like to see APT's for QIX and Gauntlet and a softkey for Tetris IIe. I currently have two levels of Gauntlet mapped (in Appleworks format). When I have completed five consecutive levels I will send then to the RDEXed.

C J Blanchard — CA

I have two programs (Paintworks Gold & Appleworks GS) that do lovely graphics. Alas, neither support my printer — an Okidata 92. I would appreciate any information on locating a suitable driver.

Raymond Ross — NV

To Paul A. Johnson: Your program worked on my ROM 01 IIgs. It printed "MOUSETEXT COMPATIBLE (MS=1)". Your program did have a couple of missing parenthesis in line 20. The third PEEK should have a left parenthesis to the left of it and the number 65055 needs a right parenthesis to the right of it.

To J. C.: I was able to run Battlechess from my hard drive using GS System 5.0.2. There is a program on America Online called Chess.Finder that (when double-clicked) will run Battlechess. But it doesn't work with GS System 5.0.3. Maybe someone will update it.

Chris Oliverson — CA

I'm having trouble with "Centauri Alliance" by Broderbund. My trouble is on Kevner's World, specifically Zentek's Tower. How do I get the password? I would also like to know who's name is supposed to be used at Wizard Tonka OG's?

To Robin Locksley: There are no lockpicks in Dungeon Master GS, although they are on the list of items on the disk. According to the publisher, the lockpicks in the Atari ST version worked erratically so they weren't implemented in the code for the Apple IIgs version.

Fred Martz — WA

I'd like to get a modified ROM for a IIc that would let me jump to the monitor. If a COMPUTIST reader has a ROM burner and can help, I would be willing to compensate them for their trouble.

W R Eade — CA

I wrote to Origin and Strategic Simulations regarding their support for Apple II software.

I mentioned in my letter that they do not have to support the minimum Apple II+ with 64K as almost no one is using that configuration anymore.

Strategic Simulations didn't bother to reply but Origin did. They stated their case for their programming decisions. They also indicated that they were interested in hearing from Apple II users and that they would take the input into account in future decisions.

With the rapidly declining amount of good (new) software available, I think we should all sit down and pen a letter to Origin telling them what machine we use and what software we would be interested in buying. Maybe we can influence them to keep writing new wares for the Apple II.

Phil Garrison — MI

I'm having great difficulty making the Garfield Companion disk work with a deprotected copy of Garfield Deluxe. When I try to access the disk thru the Library Disk Option on the menu I get an error message. Any suggestions?

Edward L. Eastman — NE

I purchased Megaworks by Megahaus recently. It LOOKS like a neat spell checker and mail merge supplement for AppleWorks. It was on sale with the now standard 'no return' policy. Suffice it to say that the original program disk won't work. I tried to call the number on the package but the phone is no longer in service. My letters have not been answered either. I suspect they are out of business.

I am almost certain the copy protection is the cause of my woes. Does anyone out there have a deprotect for this? Failing that, would someone send me a bitcopy of their program disk? I would be willing to send you my non-working original to prove ownership.

B. Dudley Brett — Canada

FIND.CAT Enhancements

"Scotty" B.M.E. Upp, in COMPUTIST #72, p.24, added a few lines of code to my DOS 3.3 Cataloging utility (v.67, pp.8,9, allowing it to print to screen as well as printer. Fine work! I never thought of this, as I just wanted a hard copy of a disk.

In response to Scotty's question, regarding problems in reading from a disk that has no Binary files, the correction is quite easy. Just add one line to the FIND.CAT programme:

```
225 IF N2 = 0 THEN TEXT : HOME :
  GOTO 560
```

For those interested programmers, the utility initially reads all Catalog sectors on Track $11 that contain information about files present. Information about file name, type and track/sector lists is stored for each file. Immediately following this, for each binary file the T/S list is read to find the initial track and sector of the file. This sector is now read and the first four bytes containing the file address and length are recorded. My mistake was to unconditionally search for T/S lists of binary files, even if there were none present. The offending line is line 230 which is accessed when track $11 reading is completed. This line GOTOs Line 500, where the Binary file sector search is done immediately before obtaining a printout.

Note, however, that a flag (N2) is incremented each time a binary file is found on track $11 (Line 430). As one can see, immediately preceding the offending line 230, I just added a check for this flag, and allowed a GOTO to the printing routine at Line 560, bypassing the T/S search.

As a final note, when I added Scotty's printer/screen enhancement to my programme, I found that neither the printer or the 80 column screen functions could be accessed. Reading through a listing of his routines provided me with the bug. Apparently just before turning on the 80 column screen there is a GET statement (in line 570). Without a prior PRINT statement in lines 580 and 590 to terminate the line after the GET input, a DOS command such as PR#1 or PR#3 will be passed over and not fulfilled. If you have added Scotty's enhancement alter these two lines (but don't do it if you haven't!):

```
580 IF PR THEN PRINT: PRINT D$;
  "PR#1": PRINT CHR$(9);"80N":
  GOTO 600
590 PRINT: PRINT D$; "PR#3"
```

Super 6.0 FastCopyA Enhancements

Dan Reid, in COMPUTIST, v.68, p.20, showed how to patch Locksmith 6.0 Fastcopy to copy selected tracks. As my version was the same as the one Dan described, I decided to include this enhancement in my programme, Super 6.0 FastCopyA (See COMPUTIST, v.72, pp.20-22). To do this simply patch in lines 284, 288, 322-328. This will now allow one to copy selected tracks the easy way, instead of making these changes when the Locksmith menu is shown. By the way, if you have a different version of Locksmith 6.0 Fastcopy (mine loads at $2000), remember to alter the pokes in lines 324 and 327 as Dan suggested (add $232 and $236 to the loading address). The pokes in the data table at the end of Super 6.0 FastCopyA will also have to be extensively altered. If someone does have a different version, please, if you can, make these alterations, but write in to COMPUTIST to let everybody know.

When I received volume #72 some time ago, I noticed 2 wee typing errors that crept in as I was creating the FastCopyA programme. These occur in lines 500 and 530. Change 111 to 11 in line 500 and change 10 to 11 in line 530. Without these changes, altering Data field read bytes becomes impossible (you will get a Dimension error and crash, boom, heck!). As an added enlightenment, receiving issue #77 today (I composed most of this article several months ago, and neglected to send it in!), I read Mr. Edison's remarks on p.4, and discovered an almost identical alteration in the same two lines. I admire anyone who can fix bugs in someone else's undocumented programme. However, Mr. Edison made a minor error. In line 500 he changed the 111 to 12, and in line 530 the 10 was changed to 12. I surmise that he misread the number of dimensioned data statements for changing read bytes. There are really exactly 11, whereas the changing write bytes number 12. I do not see that the programme will crash or do amazing things if the magic number 12 is used, but in the spirit of programming perfection, they should be left at the number 11.

Donald Oliveau — CA

IBM Softkey for...

Where in Time is Carmen Sandiego

Broderbund

I tried the softkey in issue #77 without success. The string of bytes mentioned did not exist on my version. However, I was able to defeat the request for the original disk.

Step-by-step

1. Rename the file and startup DEBUG.

```
ren carmen.exe car
debug car
```

2. Search for C2 08 00.

```
S0000 FFFF C2 08 00
```

The search for C2 08 00 finds the only instance of RET 0008 in my copy of WITICS. It occurs at the end of the subroutine that checks for the original floppy disk. Putting these three bytes at the beginning of the routine effectively cancels it.

```
xxxx.yyyy
```

Note the value of yyyy. The beginning of the subroutine is at xxxx.yyyy minus hex 80 (decimal 128). For example, on my copy C2 08 00 begins at xxxx.BAE5 and the subroutine starts at xxxx.BA65. Check that your copy is the same by disassembling the beginning of the subroutine. If zzzz is yyyy - $80 then:

```
u xxxx.zzzz
55          PUSH  BP
8BEC        MOV   BP,SP
06          PUSH  ES
1E          PUSH  DS
BF0700      MOV   DL0007
```

3. Change the 55 8B EC to C2 08 00.

```
e xxxx.zzzz
xxxx.zzzz 55.C2 8B.08 EC.00
```

Follow each correction with a space; end the line with return

Initially, I thought that this would do it but the program quit cold after displaying the initial graphic. Since there had been no check of the original disk at the point that the program quit, I assumed that a checksum was being done and was coming up wrong. Replacing the original C2 08 00 with the changed bytes at the beginning of the subroutine did the trick.

```
e xxxx.yyyy
xxxx.yyyy 55.C2 08.8B 00.EC
w                    write the changes to disk
q                    quit debug
ren car carmen.exe   rename the file
```

Although I haven't run the revised program far enough to get into the Hall of Fame, it carries out promotions without complaint.

unClassifieds

How to place an UnClassified Ad

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

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

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

★★★★ New Rates (per line) ★★★★

Computist club member
25¢
All others
35¢

The minimum order is $5.

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

COMPUTIST unCLASSIFIEDS
33821 East Orville Road
Eatonville, WA 98328

TRADE YOUR APPLE SOFTWARE

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

Byron Blystone
PO Box 1313
Snohomish, WA 98290

BOOK SALE

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

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

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

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

Add $2.50 for postage & Insurance

Software — Books — Magazines

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

Frank Polosky
PO Box 9542
Pgh, PA 15223

CRACKING TOOLS

For sale:
Computist Back Issues
Core Vol #1,2,3
Computist 1,3,7,16,19,20,22-78
Each back issue $2.50 or all for $150
No "Zox" issues here, all originals
Book of Softkeys Vol 1 for $5
Apple Machine Lanuage for Beginners by Richard Mansfield for $12
Crackniques a complete cracking tutorial includes 2 disks for $20
OR Buy everything for only $168
Save up to $350

Contact: Sven Swanson
R1 Box 143
Kensington, MN 56343

Apple Peripherals

1 Meg GS RAM $125
Serial Grappler printer interface $20
Apple Joystick $10

The Nautilus Computer
27 Nautilus Drive
Hampton Bays, NY 11946
(516) 728-3435

$10 Ten Buck Sale $10

All Software Only $10 Each

If you don't believe it, you're missing out on the excellent bargains! Several programs bought for over $30, sold for a mere $10. Get 'em while they last!!!!

Black Cauldron - IIe
Copy II Plus v8.3 (disk only) - IIe
Destroyer - IIe
F-15 Strike Eagle - IIe
Karate Champ - IIe
Marble Madness - IIe
Print Shop (enhanced) - IIe
Silicone Dreams - IIe
Strike Fleet - IIe
Softdisk Sampler - IIgs

Send check/money order for $10 each plus $3.00 shipping to:

Alan T. Zak
9724 N.W. 59th Place
Gainesville, FL 32606-2837

If you enclose 1-2 blank 5.25" disk w/order, I'll copy some assorted public domain programs on them. If that isn't a good incentive, I don't know what is!

Wanted

Most Wanted List Software

? Need assistance or solution 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. System: Apple II+, 64K. Send disk to:

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

SOFTWARE ORIGINAL SALE

Airheart, Alter Ego, Archon II, Arctic Fox, Damiano, Gamma Force, Lane Mastadon, Magic Memory, Mind Forever V., SAT Prep, Seven Cities, Wasteland, Zork Quest .....$6 each

2400 AD, Magic Window IIe, MagiCalc, Space Quest 2 .....$12 each

Send check or money order to:

David Stewart
1324 Ava Road
Severn, MD 21144

Senior PROM For Sale — $70

Complete with documentation, installation instructions, Deprotecting manual, and Senior PROM utilities disk. Hardware requirements: Apple Enhanced IIe with CD/EF monitor ROM chips. This is a low-profile version of Senior PROM designed to fit under any long peripheral cards in slot 6 or 7.

Barry Sliwinski
8424 Braddock Way
Columbia, MD 21046

MOVING AUCTION!!!

I am moving, and have lots of software, hardware, and computer books to move. I do not want to bring some of it with me, so I am offering it to the highest bidder. SUPPLIES ARE LIMITED!

Some things for sale include:

DOS 3.3 System Master (1 copy)
Copy 2+ v8.4 w/manual (2 copies)
dBASE II/CP/M Package

& Lots more.

Send a SASE for a list of stuff, and current bids to:

MOVING AUCTION
c/o Kevin C. Redden
P.O. Box 487
Vanceburg, Ky 41179 USA

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

Anthony M. Smith
PO Box 1131
Kulpsville PA 19443

RDEX Contributors:

CPR	Agent	15
C J	Blanchard	22
B. Dudley	Brett	22
W R	Eade	22
Edward	Eastman	12, 22
Parity	Error	16
Rob	Fiduccia	15
C J	Gammage	21
Phil	Garrison	22
James A.	Hodge	9
Jeff	Hurlburt	4
Scott	Jelsma	22
	Krakowicz	12
Martin P.	Longenberger	21
Fred	Martz	22
Donald	Oliveau	22
Chris	Oliverson	22
The	Overlord	15
Brian	Plautz	22
Raymond	Ross	22
Tony	Sandoval	21

Most Wanted

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

IBM Most Wanted

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

Eamon Adventure Games for only $1 each

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.*

- ☐ 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 Eamon Bluff
- ☐ 52 The Devil's Dungeon
- ☐ 53 Feast of Carroll
- ☐ 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 Crab Key
- ☐ 72 House on Eamon 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

- ☐ Dungeon Designer Diskette v7.0
- ☐ Multi-Disk Supplement DDD7.0
- ☐ Dungeon Designer Diskette v6.2
- ☐ Eamon Utilities Diskette
- ☐ Graphics Main Hall

Free Eamon Adventure disks

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

# 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

Complete set of Eamon

All 223 disks (includes all adventures plus designer and utility disks.) $120

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

❑ Send me the Complete set of Eamon for: $120.00 _____

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

If total # of disks ordered is less than 10, add $4 for postage & handling. _____

Washington residents add 7.8% sales tax. _____

Name _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone _____

❑ Visa

❑ MC _____ - _____ - _____ - _____ Exp _____

Find it Fast! Make it Easy!

Want a **quick and easy** way to find your favorite softkey, article or play tip without having to re-read your entire COMPUTIST library? This is what you are looking for:

The Computist* Super Index!

The Computist Super Index is a data base listing of all the Softkeys, Articles, Play Tips, APT's, and RDEX inputs printed in Computist magazine. Three colossal indexes with over 6,000 records and 31,000 entries! (Over 100 records and 500 entries added each issue!)

With The Computist Super Index, you instantly have the title, issue, page, special requirements, bugs, and "type" of softkey! At the touch of a key, you can search, sort, and print softkeys by program name, publisher, or de-protection method! By entering a key word or characters, the location of every article, tip, and entry about your topic of interest, is immediately sent to the screen or printer.

Extra Bonus! The Computist Super Index includes nearly 3,000 softkeys and tips (hidden within the RDEX), not printed in the contents or back-issue listings of Computist! If it is printed in Computist, it is listed in The Computist Super Index! (IBM and MacIntosh entries included.)

***David Hopkins is not connected with COMPUTIST magazine and The Computist Super Index is not a product of COMPUTIST or SoftKey Publishing.**

Features

• Three (index) data bases: Softkeys, Inputs-Articles-Programs, and Play Tips-Advanced Playing Techniques-Reviews.

• Over ten fields to search and sort, including: Title, Issue, Page, Distributor, Input Location, Softkey Type, Bugs, Author, and more!

• All RDEX entries are summarized and categorized! Categories include: Articles, Programs, Editorials, Queries, Inputs, Listings, Help Wanted's, and others.

• The Computist Super Index lists virtually everything printed in Computist, from Hardcore Volume 1, Number 1, to the present! IBM entries in separate indexes.

• Softkey "Type" identifies the exact deprotection techniques used for each softkey! Includes: Sector & Block Edits, Modified Copya, Controller, DOS Patch, File Copy, Bootcode Trace, RWTS Capture, Hex Dumps, Hardware Copy, etc.

• Format: Appleworks, DIF, or Text (ASCII). Please note: Since Appleworks screen and printer formats are lost when the files are converted to DIF or Text (ASCII), Appleworks is the recommended format.

Computist Super Index

The Computist Super Index is updated each issue. All orders sent Air Mail. Price includes shipping. Hardware requirements: Apple II, II+, IIe, IIc, IIgs, or compatible, with at least 128K RAM for the Appleworks version, 48K for the DIF and Text versions. Not available in MS-DOS format. The indexes are broken into 40K segments for the 5.25" version and RAM size less than 256K. Order The Computist Super Index today!

Update Subscriptions (must pre-order Index)

An Update is a wise investment! An Update Subscription is even better!
(1) Each update contains the entire data base, plus the current issue, and all corrections and revisions.
(2) You get a new, fully updated, copy of The Computist Super Index with each update, and at half of the original index price!
(3) Each update totally replaces all previous versions!
(4) Save money, time, and effort with update subscriptions. The more updates ordered, the greater the savings!

An update subscription is four to eight updates, sent every one to four issues. **Please note: You must purchase the INDEX before you can get an update subscription.**

Order Form

How to order:

(1) Copy order form.
(2) Mark desired options.
(3) Complete questions.
(4) Send with check or money order to:

David R. Hopkins
3495 W. Hoye Place
Denver, CO 80219

What COMPUTERS are you using? _____

What is your RAM SIZE? _____

What DATA BASES are you using? _____

What WORD PROCESSORS are you using? _____

What ISSUE # do you want your Update Subscription to start with? _____

Prices (effective January 1, 1991)

	U.S.	Foreign
Computist Super INDEX (5.25" or 3.5")	❑ $19.95	❑ $22.50

	Every issue	One every 2 issues	One every 3 issues
U.S. (5.25" or 3.5"):	❑ $48.80	❑ $42.30	❑ $31.80
Foreign (5.25" or 3.5"):	❑ $67.20	❑ $56.40	❑ $41.20
Total Updates:	8	6	4

Individual Updates are available for ❑ $9.95 (US) or ❑ $12.50 (foreign)

Please specify format, DOS, and disk size by checking **one** box in each **row**.

Format:	❑ Appleworks	❑ Text (ASCII)	❑ DIF
DOS:	❑ ProDOS	❑ DOS 3.3	❑ GS/OS
Disk size:	❑ 5.25"	❑ 3.5"	

Name _____

Address _____

City _____ State _____ Zip _____

• Send U.S. funds drawn on U.S. bank. Not available by charge card or COD. Prices subject to change without notice. Please make checks payable to: David R. Hopkins

Thank you for your order and support!