85 COMPUTIST

Canada & Mexico $7

U.S. $3.75

Contents

Subscribe to COMPUTIST

We give you More!

Only $25 for 8 issues

I am:

- ○ Renewing my current subscription
- ○ Changing my address (Please include last label)
- ○ A new subscriber.

Subscription rates:

- ☐ **$27** **For new subscribers** (and late renewals that missed 1 issue. U.S. regular/3rd Class Mail) Includes an extra $2 for postage and handling. We will send your **Starter Kit** and 1st issue by 1st Class mail.
- ☐ **$25** U.S. regular renewal (3rd Class mail)
- ☐ **$35** 1st Class (U.S. / Canada / Mexico)
- ☐ **$54** All other Foreign (Air mail)
- ☐ **$68** COMBO (1st Class plus Disk)
- ☐ **$95** Foreign COMBO (Air mail plus Disk)

VISA **Charge It!** **MC**

Name ____________________

Address ____________________

____

City ______________ State ______ Zip ______

Country ______________ Phone ______________

Visa/MC ______ - ______ - ______ - ______ Exp. ___ - ___

Signature ____________________ #85

•Send US funds drawn on US bank. • For regular subscriptions, please allow 4-8 weeks for 1st issue or add $2 for postage and we will send your 1st issue by 1st Class mail. • Send check/money order to:

COMPUTIST 33821 E Orville Rd Eatonville WA 98328-9590 (206) 832-3055

COMPUTIST(85)
PO Box 242
Kapowsin WA 98344-0242

Address correction requested
Forwarding & return postage guaranteed

BULK RATE
U.S Postage
PAID
Kapowsin WA
Permit No. 6

COMPUTIST

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

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

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

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

• Unsolicited material (manuscripts, letters to the editor, softkeys, A.P.T.s, playing tips, questions, etc.) are assumed to be submitted as letters-to-the-RDEX-editor for publication with all and exclusive rights belonging to COMPUTIST.

• Entire contents copyright 1990 by SoftKey Publishing. All rights reserved. Copying done for other than personal or internal reference (without express written permission from the publisher) is prohibited.

• The SoftKey Publishing assumes no liability or responsibility for the products advertised in this newsletter. Although we are usually pretty much in agreement, any opinions expressed by the authors are not necessarily those of COMPUTIST magazine or SoftKey Publishing.

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.

Apple® is a trademark of Apple Computers. IBM® is the IBM trademark.

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: Reader's Data EXchange. We print what you write. When you send in articles, softkeys, APTs, etc., you are submitting them for *free* publication in this magazine. RDEX does *not* purchase submissions nor do we verify data submitted by readers. If you discover any errors, please let us know so that we may inform our other readers.

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

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

How to get a free library disk

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

When to include a printed letter

Don't include hardcopy (printout) unless:

a. You are writing about a bug or other printing error.

b. You are writing to ask for help.

c. You are answering another readers help request.

d. You are writing about your subscription or sending an order for back issues or software.

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

Writing to get help

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

How to get mail

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

How to write to RDEX authors

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

Help Line

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

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

Marc Batchelor, 6025 Coker St., Cocoa, FL 32927

Rich Etarip, 824 William Charles Ct. #2, Green Bay, WI 54304-4018

The BBS
(Bulletin Board System)

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

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

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

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

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

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

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

United States Code title 17, §117

85 COMPUTIST
U.S. $3.75
Canada & Mexico $7

Table of Contents

Editorial Notes

It's a double issue! That's right. Things got a little heavy here. Our laser printer went south (something about bubbles on the fuser roller) and we couldn't print the final layout. So issue #84 didn't make it to the printer on time.

But the computer still worked so we continued to layout issue #85. It took a few weeks to get the printer fixed. The local Apple dealer wanted $600 to replace the fuser roller assembly. We decided to do some shopping around and found a Computerland store that did the repairs for $342. So we save $250 and lost several weeks.

Which brings us to now. Issue #84 was almost finished when the laser printer went kaput. By the time we got the repaired printer back we had finished issue #84 and issue #85.

We decided to print (and mail) them both at the same time.

We have a lot of new material so we may just do this again to try to get caught up with our schedule.

Ha! Ha! I know, you don't believe Computist has any schedule (judging by past performance) but we do. We just don't seem to be able to stick to it very well.

So anyway, expect another double issue soon.

Jeff Hurlburt TX

The PRODUCT MONITOR

RATINGS

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

GD 301: Spring Seminar (1PM session)

This afternoon we shall discuss a problem which continues to bedevil designers and frustrate players of swords & sorcery adventures: Why, despite the promise of easy character transferability, do many such series dry up after just one or two runs?

In the long history of swords-and-sorcery computer adventuring, only a few epics allow taking a party of characters through second and third installments of the same core scenario. Mainly, the explanation is the 'Superman Syndrome'. Monster extermination is more fun when there's an escalating challenge and the payoff of steadily increasing character powers. Yet, under the direction of a skilled adventure gamer, the party's fighters, clerics, mages, etc. can become too powerful! Like Superman, they are virtually invulnerable. For designers, coming up with worthy, believable opposition and still greater character powers to match is a lot more bother than simply wrapping things up and starting a new series.

Most adventurers, I'm sure, judged the "Pool of Radiance" series successfully concluded with its first sequel. In "Curse of the Azure Bonds" we demolished powerful forces and put the lid on Tyranthraxus, POR's arch villain. It was not an easy quest; but, in the end, followers of the evil god Bane were blasted to grease or left scattered and fearful. Phlan and surrounding Moon Sea lands could prosper in peace.

It was not to be. Trouble was brewing in nearby Verdigris Valley; so, "Secret of the Silver Blades" teleports your battle-honed party to New Verdigris, with attributes intact, to battle a powerful lich. Despite the need to acquire new weapons, armor, etc., this is much the smallest quest of the series both in terms of gamescape and duration. Small wonder! With two major quests under their belts, your heroes are soon munching small armies like M&M's. Evil's monsters, guards, and mages just don't have the stuff to mount a credible threat in any single combat. You suspect something is awry when you notice your mages are carrying crates brimming with fireball scrolls. You know the tactical challenge is gone when a game's climactic encounter comes down to winning the same battle five or six times in succession.

"Secret of the Silver Blades" shares the locale, mythos, and characters of earlier quests, but, adds very little to the story. Given the 'teleport without stuff' gimmick, and with scenario development on 'hold', the "Pool of Radiance" saga seemed poised for an endless stream of low-challenge tack-on releases: "Puzzle of the Platinum Pikes", "Pool Heroes Meet the Mummy", etc., etc.. To the credit of SSI's design team, one such diversion was enough. They saw that there is no way around the Superman problem. The only way to handle Superman is to up the stakes and pose a Super Challenge! Your dauntless band did not know it at the time, but the "'Silver Blades" quest was their vacation— a bit of R&R before "The Final Challenge" facing "The Ultimate Enemy"!!!

Pools of Darkness

★★★

$49.95 for EGA-VGA 640K PC

Strategic Simulations/EA

Optional cluebook: $12.95

(From the journal of Froolin the Ubiquitous)

It seems like only yesterday that I met with Mothnose, Goo-Goo, Rubywand, and several other heroes for lunch in Phlan's newly restored Valhegen Park. Rubywand was slowly turning her crystal goblet to catch the sun's glint in Phlan's best golden wine. "You know", she mused, "if we continue to knock off big-time minions of Evil like Tyranthraxus, sooner or later we could run into somebody who is really bad news." 'Most everyone laughed, since the dragon-mage had proved entirely adequate in the "bad news" department. Batfoot just continued to stare dourly at his empty goblet. "Well, I say bring 'em on," he rumbled. "Another week of patrols like the last, and we'll be down to commissions for park guards. Whoever thought Phlan would come to this!"

It is fortunate that the next day's duty took us far from town, though no one felt especially lucky when the weather changed. After hours holed up in an abandoned shack to escape the wrath of an absolutely incredible storm, we returned to find the town... missing! Phlan was gone. Where? How? We could scarcely guess; but, all that remained was a gigantic crater. Mothnose huffed up to the lip, peered down, and shook his head: "There goes the neighborhood. This has to be the work of a major league meanie." Rubywand nodded, "Yes, but not just a big guy. This time I'm afraid we've hit the jackpot." ...

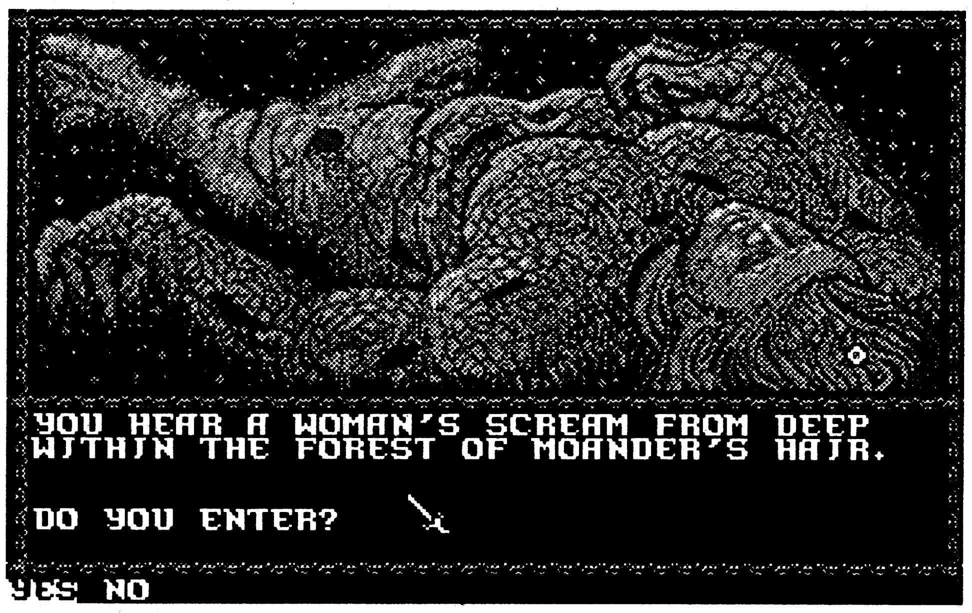
Too right! Bane was VERY annoyed by your victory over a favorite minion in Phlan; being foiled in New Verdigris was the last straw! It's "no more Mr. Nice Guy"; and the cataclysmic ripping out of cities around the Moon Sea is just for openers. Bane, courtesy of SSI's scenario writers, has pulled out the stops: "So, an arch-mage lich backed by legions of guards, spell casters, and monsters was 'too easy'? Well, just go ahead and transfer your characters in tact from 'Silver Blades— OR, start with new guys; they'll still come in around Level 6. KEEP your rings, wands, +5 plate armor, silver long swords, ... whatever. You think you're such a hot bunch of super heroes?! Well, try THIS!"

If SSI published comic books this would be the "Major Minions Team-UP" issue. Your party comes up against 1. Thorne, an ancient red dragon who guards the Horn of Doom, 2. Modthryh, a wizard creating undead Dracolich spell-casters, 3. Marcus, Adept Cleric of Bane who animates chunks of flesh from the comatose god Moander, 4. Tanetal, demon lord of the Moander Dimension, in charge of Moander 'mining operations', and holder of the Talisman of Bane, 5. Kalistes, half-serpent mistress of the Web Dimension, keeper of the Crystal Ring, and inspiration for creation of giant spider mages by Drow elf cultists, 6. Arcam, an Elder Beholder who rules Mulmaster and guards portals to the Lands of Bane, where you encounter 7. Gothemene, Balor Arch Demon, 1st lieutenant and chosen one of Bane!

Dealing with each entails a mini-quest, some of which are not so "mini". One does not, for instance, just walk up to Thorne's cave and knock. You begin at the Hill Giant Steading (an Evil Forces recruitment center), make your way through traps and guards in the Fire Giant's cave; and, then, in the Aerie, fight flocks of dragons to collect the four keys you need to reach the portal leading to Thorne! (Not as easy as it sounds.) In "mapese", this one questlet translates to a 32 x 32 region crammed with rooms and corridors. Your campaigns against Tanetal and Kalistes are much tougher. The Kalistes quest alone is nearly equivalent to a complete adventure.

Bane has planned his big grab for power well. In the Real Realms, dragons, vampire mages, giants, etc. scour the lands while his followers continue to organize in places like the 'Steading, Zhentil Keep, and Mulmaster. (The bad towns did not get scooped.) However, the real centers of power are in the Dark Realms dimensions, reachable only via the Pools (portals); and these are all well defended. PLUS, when your party moves into a dark dimension, practically all weapons, armor, and other Realm Realms equipment must be left behind; otherwise, it's destroyed!

Fortunately, between the realms you find Limbo, a handy stopping-off place where you may place items in storage, Encamp (to rest, heal, and restore spells), obtain any Healing your clerics cannot handle, and Train to advance in Level. Here, as well, you find Elminster, a good arch-mage who regularly supplies helpful advice and encouragement. It is from Elminster that you learn something about the forces at work and the personages charged with implementing the evil god's grand design. Perhaps most important of all, you discover hope!


YOU HEAR A WOMAN'S SCREAM FROM DEEP
WITHIN THE FOREST OF MOANDER'S HAIR.

DO YOU ENTER?

YES NO

Elminster believes Bane's works may yet be undone IF you can reclaim certain powerful artifacts. Naturally, these are held by the dark god's greatest minions.

Like earlier POR releases, Pools of Darkness wraps your adventuring into the story via on-screen text, 'cut scenes' for special situations, and well-written "Adventurer's Journal" passages. You will overhear conversations, find maps and messages, and encounter numerous personages with rumors, history, and important clues to impart. Adding to the fun and realism of each meeting, there is always an attractive, partially animated picture and, occasionally, sound effects and music.

Not every encounter is packed with clues. Shopkeepers, Trainers, etc. are concerned with the business at hand. Sometimes, as when coming face to face with an arch villain, a lot of what you get amounts to pre-battle hype. Other times, there's humor and irritating duplicity, as when you're dealing with Phlan's new mayor, Sasha. (She was out of town during the 'big scoop'; and, of course, manages to get into more trouble than ever!) Several characters will offer to join the party for their own reasons— a dwarf who wants to rescue his sister, etc.. Even your old comrade from "'Silver Blades", the talented Vala, needs some help to end an invasion threat from the east. (Yet another mini-quest!)

With so many in-game resources, will you need the (72-page!) "Clue Book" too? To crack puzzles, probably not. There are just a few; and, only your stint in Moander's heart qualifies as a "tricky situation". Most of the heavy duty challenge comes in combat encounters. The CB's numbered map references can help you avoid unnecessary battles, steer you to weapons caches, and, in general, reduce the need for exploration. On the other hand, it is very easy to over use such a powerful reference and miss the enjoyment of genuine discovery and problem solving. Should you pick up the CB when you get the game? If you don't enjoy mapping, definitely! Even if map making constitutes a special delight, having ready access to the ultimate un-sticker is nice, just in case. But; you do not absolutely need it. Both in 'holding the story together' and supplying what you must know to succeed, the program and manuals get the job done.

While the promise of another TSR swords & sorcery scenario is the Pools of Darkness 'up front' attraction, SSI's Advanced D&D gaming system is what makes everything work. This means you can quickly check any character's possessions and status (e.g. attributes, hit points, armor class, etc.). Spell casting, equipping items, trading, buying and selling, etc. are equally speedy. The idea is simple: if a player will want to check it, do it, or change it, then stick IT in a clickable menu he or she can get to with minimum hassle. True, current spell effects (e.g. "Blessed", "Hasted", etc.) should be shown in each character's normal "View" display— not just available during "Encamp". And, yes, it would be very handy to have armor class and "damage" (hit power) numbers on the "Items" display, where you equip armor and weapons. There IS room for improvement, but, not a whole lot. No one offers a more user-friendly interface. Indeed, it remains a mystery why one still encounters so many computer S&S adventures with cumbersome, user-UNfriendly interfaces. All any designer need do is boot one of the current SSI AD&D releases to see how to 'do it right'!

Just after the cataclysm strikes, your party appears as a dot on the crisply detailed (256-color VGA) single screen map of the Moon Sea region. (The Web Dimension and Moander each has its own "big map" as well.) Guiding the dot can take your party to the Temple of Tyr, Zhentil Keep, and many other interesting places. During explorations of towns, towers, dungeons, etc. you will often have a choice of two displays (placed in the upper left portion of the screen). One, a 3-D perspective forward view of nearby walls, doorways, etc. is always available. As in earlier POR releases, level of detail here remains 'just fair'. You can readily distinguish town buildings from temple interiors or the insides of Moander; but, basically, this is EGA-class stuff. Another deficit: the view still fails to show approaching monsters or personages.

Selecting "Area" substitutes a bare bones top-down diagram showing walls and corridors (but, not doors) for several 'squares' in each direction. Since you can move the party (an arrow symbol) on the map as easily as in the 'normal' 3-D forward view, "Area" is a very handy navigation aid. Probably, it's too handy. That, sometimes, the designers feel compelled to turn 'off' the feature— you get a "Not Here" message and must stick with the 3-D view— only underscores the problem. "Area" gives away too much information. To enjoy the more realistic forward view requires a conscious decision not to explore in the efficient, but boring, "Area" mode. A better setup would make the "Area" display self-mapping— that is, "What you see is where you've been" (pronounced "WISIWIB"!)— with movement allowed only in 3-D mode.

Tactical combat remains the highlight of AD&D gaming; and, in the smooth-scrolling, multi-screen battlescapes of Pools', you will face some of the toughest, most demanding challenges ever. Partly, it's the quality of your opponents: several, like the giant spider "Pets of Kalistes", Black Circle mages, and Moander Fanatics are dangerous magic users. Beholders combine deadly multi-spell barrages with near total immunity to any magical attack! Many enemies, such as the dragons, Minions of Bane, Bits O' Moander, and the Giant Cockatrice employ lightning, fire, frost, acid, poison and other powerful 'natural' weapons against which the Globe of Invulnerability is useless. Add the usual supporting cast of warriors, bowmen, assassins, etc. and you get the picture: some of the encounters would be "rather difficult" even if the bad guys wore red coats and marched in a straight line!

They don't; and, neither do you. The most entertaining and challenging feature of several engagements is the terrain. In Pools', the top-down, partial-perspective-view battlescape accurately reflects your current location in a maze, a building, or in the countryside. This means you and the enemy have several 'screens worth' of rooms, corridors, alcoves, trees, rocks, streams, etc. in which to maneuver!— AND, it's all in nicely detailed, partial-animation VGA with AdLib/SB sound effects. (Adversaries move and slash, arrows zip, lightning bolts ripple, fireballs mushroom, ...) In many combats, using walls, doorways, etc. effectively may simply avoid serious injury and having to risk encampment in a dangerous area. The REALLY tough battles all require some 'solution' which takes advantage of one or more terrain features. A 'wrong answer' here is the last answer (at least until you Restore from a saved position). Easy or tough, from the first encounter to the final showdown, you can count upon flexible, easy to use KB and mouse controls to get the most from each character. For good or ill, your strategems will, virtually, come to life.

For sure, all games have bugs! Pools' version 1.00 had more than its share; but, with version 1.10, almost all notable problems seem to have been exterminated. The exceptions include one oversight and two rather low-probability bugs. Bug #1 can 'hang' the game when monsters are gated-in via a Summon Monsters spell. In many many SM castings, this happened just a few times. A second bug crops up when you are in the Clerk's office in Phlan. If you request a commission and none are available, you may find that you cannot leave the office, ever! Evidently, the only commission is the assignment to help Vala defeat the Vaasans. Once you've been paid off for the Vala mission, DON'T ASK for another!

You can easily spot the "oversight" on page 48 of the "'Journal". Here you find that advancement for non-human characters is severely limited in all occupations except "Thief". A dwarf Fighter, for example, cannot advance beyond Level 9! (The highest non-human, non-Thief Level is 11, for an Elf Magic User.) Since a same-experience human Fighter, Cleric, etc. can easily advance beyond Level 30 by the end of the game, this means non-human, non-Thief characters go through most of the adventure with NO tangible payoff for their achievements. Needless to say, it did not take long for all of my affected dwarfs, etc. to undergo Humanization! (How? Example: To Humanize the 'first character' in your party saved in Game "J", edit CHARDATJ1.GAM using "Xtree Gold" or similar utility. Just set the 'race byte' at location $00AE to $05.)

Pools of Darkness tackles the Superman problem head-on and, despite the odds, is singularly successful. Your party starts as a very rugged, hard-hitting bunch and soon meets very tough adversaries. As Level advances and new spells augment your powers, the challenge escalates again and again. It's a dual for power spanning the biggest AD&D gamescape, with more major personages, more monsters, and tougher combat than ever before. When, at last, you come to the face-off with Gothemene, there's no doubt: each side KNOWS it faces "the ultimate enemy" in The Final Confrontation! Carve out three or four weeks of game time. Prepare for the 'Ultimate Wrap-up' to a classic adventuring saga when you enter the Pools of Darkness!

Fast Frames, Updates, etc.

In the Lore Library: Pools Finale

They were there all right, in the "New Arcane Scripts" bin. The sheets were headed "Pools: Notes of Gorbash and Turdnil on the Big Showdown":

The final 3-part battle with Gothemene is VERY rough. Thus, for the first time in the Pools of Darkness quest, we have resorted to the item transfer and duplication powers available via the Remove Character option. Two examples should suffice to show how these higher-level magics may be employed to good advantage:

Transferum del Ultra

You have just returned to Limbo after a hard day in Dark Phlan (in Bane's domain). It would, you now realize, be very nice to have some of the Real Realms items you had to leave in Storage. (Except for rings, Drow equipment, and the Vorpal Sword, all such items in the possession of party members would have been destroyed at the moment of crossing into the Dark Realms.) The solution is to bring goody-laden temporary characters into the party after the regular characters have made the crossing.

First, "Remove", (not "Drop"!) all but one character. Next, create a new character named "Agent", add him to the party, and "Remove" your last regular character. Now, "Move On" (with Agent) to "Real Realms" and create two or three new characters named "Holder1", "Holder2", etc.. Add them to the party, go to Storage, load them with desired items, and "Remove" them. Agent, still empty-handed, crosses back into Banesville, arrives in Dark Phlan, turns around, and reenters Limbo. Now, you can bring back your regular characters, "Drop" Agent, and bring in your "Holder"'s long enough to transfer items.

Dupliccacio Mondo Grosso

Due, largely, to a barrage of lightning bolts, the party has just been extinguished in your first try at the Final Battle. GooGoo has a Ring of Electrical Immunity; but, you need one for each character. Fortunately, upon startup, or, by selecting "Train" in Limbo, you have the option to "Remove", "Create", etc. characters. So, first, "Remove" GooGoo. Respond "No" to the "Overwrite GooGoo Yes/No?" prompt; and enter a new name, "Goo2". (You may not get the prompt the first time. Just "Add" GooGoo and "Remove" him again.) Now both GooGoo and Goo2 are available for addition to the party and each has the desired Ring! Etc., etc. ...

The Battle

Not having read the Clue Book description, my first six or seven tries at ultimate victory produced an astonishing picture of the challenge. After defeating a large force of Spider mages, Bane Minions, Moander mounds, AND Dracolich dragon mages, one must, without benefit of "Encamp" to Rest and Fix, tackle a major force of magic-proof Beholders! Having managed, several tries later, to get through that combat with a few live characters, I called in Gorbash to witness the expected Victory display, pressed RETURN, and... found my party matched against Gothemene and a swarm of Minions supported by Moander mounds and Black Dragons! PLUS, Bane decides his guys need some help(?); so, he eliminates my magic powers!!! The battle did not last long. I decided to Restore to an earlier Save in Limbo and do some serious re-equipping.

Recommended Equipment

Aside from the usual armor and weapons, every party member should wear a Ring of Electrical Immunity, Boots of Speed, Girdle of Giant Strength, and carry a bow & arrows, staff sling, or other ranged weapon. Of these, only the latter are absolutely essential— you MUST survive the first two combats with at least one character able to attack Minions at range. (These monsters return twice the damage for any blow landed by a sword or other close combat weapon.) The Girdles save you the bother of depending upon Enlarge, while the Boots guarantee good movement range whether or not you've cast a Haste.

Having the Rings lowers the chance of a premature ending to the first combat. Since you CAN, with luck, bring every character through the initial flurry of lightning strikes without the Rings— and, since you should, probably, restart and reload the game if you don't— the Rings amount to another bother reducer. (IMPORTANT: The magic effect of some rings 'times out'. Be sure to flick all spell rings off and on every few rounds.)

Final Showdown Strategems

You will have to fashion tactics best suited to your own party. Evidently, there are many routes to victory; so, you may wish to put off reading any further and compare notes after winning. On the other hand, you may not.

1. General: With no between-combat Encamp's allowed, you must prepare for the next combat (flick rings and cast Dispel Magic, Heal's, Bless, Haste, etc.) during the one you've just won. (i.e. Say "Yes" to "Continue Battle?" until all preparations are completed.) Use Dispel Magic to counter effects of Slow. Beware of placing characters on "Guard" with Minions around. Finally, try not to walk into your own Blade Barrier or Target yourself with a Fireball (sigh).

2. Cast Haste before each combat and Mind Blank's before each of the first two. Haste gives your fighters multiple hits. MB protects from Fear and Charm.

3. Combat 1: Get out of the center of the room! Try using the North chamber as a fortress and cast Blade Barrier to block half of the entrance. If your Mages are Lightning-protected, place them in the NW corner of the main room and blast the Spiders and Minions with DB Fireballs. Summon Monsters brings in some helpful allies once the Spiders gone. Use Ice Storm, etc. and arrows (fired from North chamber) to finish the rest.

4. Combat 2: Spread out! Immediately send Fighters against Beholders. Cast Summon Monsters to supply diversionary targets. High Level Clerics should try Turn Undead against Death Tyrants; otherwise, assault with flails. Attack, attack, attack!

5. Final Combat: Forget "Cast" and "Use"; but do flick Electrical Immunity Rings each round. (They still work, most of the time.) Immediately send Fighters

against Gothemene. Next, send them against Moanders or Dragons threatening other party members. Switch to bow & arrow. Use Clerics and Mages to attract Minions and keep them away from bowmen. Use arrows to eliminate Minions and mop-up any remaining monsters.

Cleaning Tip

You've just peeled off an over-sized diskette label and need to get rid of the adhesive residue which your new label will not cover. Before risking the application of some cleaning solution, try lifting off the patch with Scotch "Magic Tape"!

Jimmy Connors Pro Tennis Tour ★★★★

No doubt, after the highly favorable review of "Pro Tennis Tour II", flocks of fans scoured local shops looking for the best in computer tennis. With luck, each encountered a sales person who steered them to Jimmy Connors' Tennis' ($39.95, for 640K PC). UBI Soft signed up "the greatest tennis champion ever" and changed the name in order to add a touch of pizazz. (Good idea! Look for a bright green box with an action shot of Connors on the front.) Everything else—crisp VGA displays, AdLib/SB sound, computer players, 1-4 human players using joystick(s), tournament setups, multiple court surfaces, practice with programmable ball machine, ...—is the same, including the rating!

More Links ★★★★

Evidently, at Access the camcorders and digitizers never rest. The result is two new super-realistic Links courses: Hyatt Dorado Beach East and Barton Creek ($24.95 each, for VGA 640K PC). Set in Puerto Rico, Robert Trent's 6985 yard Par 72 'Dorado Beach' takes you from palm-lined jungle tees to the seaside. Treadlike fairways, creative bunker placement, and tricky greens all reward accuracy first, power second. Tom Fazio's 6956 yard par 72 Barton Creek design takes advantage of the natural mix of 'wide-open spaces', tree islands, and water flows you expect to find in the central Texas hill country. Featuring uniquely tricky slopes, this is a balanced power/accuracy challenge which often starts you on spacious hilltop tees shooting at the fairway.

With Bay Hill, Access began including slide show tours of the course and facilities on separate diskettes in each package—colorful, attractive, but, as of the latest release, there's still no sound! (By the way, to see the Barton Creek tour enter "BARTOUR", not "BCTOUR" as directed on the diskette label.) Course installation also remains more cumbersome than it should be. As long as Access continues to include updates of the main program— the latest version of GOLF.EXE is 1.52— why not round off the few remaining rough edges? Simplify course installation and version updating; plus, utilize the AdLib/SB sound interface already part of Links to get music (and Sound Blaster speech!) for the tours. Both "Add Course" and "Take Tour" belong on the Links Main Menu.

Next

Expect Accolade's sexy Lost in LA and Elvira II, a goody or two from Electronic Arts, some new Disney stuff, IIgs books from Addision-Wesley, a fix that just may cure that glitchy PC keyboard, and .... more!

Vendors

Access Software
4910 W. Amelia Earhart Drive
Salt Lake City, Utah 84116
Attn: Susan Dunn/ Steve Witzel (800-800-4880/ 801-359-2900)

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

Brown & Wagh
130D Knowles Drive
Los Gatos, CA 95030
Attn: LouAnn Meir (800-451-0900) ref. Sound Blaster

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

Strategic Simulations Inc.
675 Almandor Ave
Sunnyvale, CA 94086
Attn: Kathleen Watson (408-737-6800) dist: Electronic Arts

Ubi Soft
1505 Bridgeway, Suite 105
Sausalito, CA 94965
Attn: Leila Emadin (415-332-8749) dist: Electronic Arts

M.M. McFadden CA

PDOS (ProDOS RDOS) v1.1

(This is an updated version of an article which originally appeared in Computist #52, pages 24-29.)

Requirements:
ProDOS

A few years back, the folks over at Strategic Simulations, Inc. came up with a modified DOS called RDOS. It used DOS 3.2 and had a double boot loader so that it would work on DOS 3.2 and 3.3 compatible drives.

A little while later, a pirate named Krakowicz came up with RDOS 3.3, a version of RDOS that would work with DOS 3.3 disks instead of the older DOS 3.2. A modified version of COPYA, called COPYB, made the transfer easier.

Then, in 1986, SSI finally decided to update their disks to 16-sector format. This was a "real" version of RDOS 3.3, and had some major internal alterations... but the ampersand interface was left unchanged.

In every case, the files were trapped in the RDOS format. My first attempt at cracking RDOS programs was to try to move programs to DOS 3.3. However, since RDOS only uses the memory from $B100 to $BFFF, whereas DOS 3.3 uses everything from $9600 up, a program copied from RDOS to standard DOS wouldn't have enough memory to work.

One solution was to use a DOS that resided in the RAM card. But RDOS doesn't use normal commands; it works completely with ampersand (&) commands from Applesoft BASIC. The commands could be changed within the program, but other problems arise.

DOS 3.3 uses the output hooks ($36-37) to intercept commands (print ctrl-D). Since RDOS doesn't use them, DOS is often disconnected, leaving commands ignored.

Other problems arise when transferring programs. It isn't possible to transfer over a large Applesoft file to DOS 3.3 without considerable difficulty. Any transfer program would have to run in machine language, and would have to be careful since the DOS save and load commands tend to wrench things around.

Then, a few years back, Apple introduced ProDOS. ProDOS runs almost entirely in the upper 16K of memory, allowing it to reside with an implementation of RDOS. Applesoft files can be BSAVED by using the T parameter, so a transfer program is simple. But one problem remained: what to do about the ampersand interface used by RDOS?

RDOS 2.1 Under the Microscope

To solve this problem, the only thing required was someone crazy enough to tear apart RDOS and re-write it under ProDOS. So, one fine morning I donned by straight jacket and went to work.

Eight hours later, I had torn apart the RDOS code (thanks to the method given in Enhancing Your Apple II by Don Lancaster and the info in What's Where in the Apple II by William F. Luebbert). Here is a general description of what I found:

Memory Map

Location	Purpose
$B100-B2FF	file buffers
$B300-B679	code for RDOS commands
$B67A-B9FF	RDOS subroutines, error messages, etc.
$BA00-BFFF	DOS 3.2 RWTS (almost unchanged)

The ampersand interpreter begins at $B303. There are 17 commands available under RDOS 2.1:

&CAT : catalogs a disk. The actual code is read from block 25 (track 1, sector 12) of the RDOS disk (so attempting to catalog a non-RDOS disk could conceivably crash the program!). Note that the PDOS version of &CAT shows the ProDOS block count, not the DOS 3.3 sector count (I figured this would be less confusing).

&LOAD "filename" {,addr} : loads a BASIC program. You may specify a different load location for it (automates "poke 103,lo: poke 104,hi").

&RUN "filename" {,addr} : executes a BASIC program.

&GOTO "filename" {,addr} : used to "chain" programs. Variables are saved, the new program is loaded, variables are restored, and the program is executed.

&SAVE "filename": saves the current BASIC program.

&STORE "filename", addr, len : BSAVES a file. The DOS 3.3 command BSAVE SPUDS, A$300, L$200 would be &STORE"SPUDS", 768, 512.

&RECALL "filename" {,addr} : BLOADS a file.

&DEF "filename", size : creates a text file SIZE blocks long. Because RDOS uses a contiguous file system (like UCSD Pascal), it is necessary to determine the size of the file before it is written.

&PRINT "filename" : writes a text file. Note that the file is truncated first (NOTE: PDOS v1.0 appended to the file; this has been fixed in v1.1).

&READ "filename": reads a text file.

&END : terminates reading or writing of text file by printing ASCII character $00 (nul).

&DEL "filename": deletes a file.

&LEN : prints the start location and size of the BASIC program in memory, and prints the current lomem value. This command was removed from 48K PDOS to make room for some important features.

&D#, nxtcom : changes the drive number. Must be used in conjunction with another command (i.e., &D2, CAT)

&S#, nxtcom : changes the slot number. See above.

&NEW : erase the current program, reset himem, and coldstart basic (like DOS 3.3 "FP" command).

&USR addr : If a command is not intercepted by RDOS, it is passed on to the routine at ADDR. Do not use a comma. This provides a way to chain to other ampersand routines.

Filenames, addresses, and slot/drive numbers can be variables. It is perfectly legal to write statements like:

```
150 &RECALL "SEGMENT" +
  STR$(SG) + "A", LOC + 5
```

PDOS : RDOS under ProDOS

The problem at hand was to duplicate RDOS using ProDOS MLI calls instead of the RWTS routine. This effort took four days and required almost 1500 lines of code (special thanks go to Beneath Apple ProDOS by Don Worth and Pieter Lechner).

The result is an RDOS work-a-like which is compatible with RDOS files and 100% compatible with standard ProDOS file types. After conversion, RDOS files can be loaded with BASIC.SYSTEM, and ProDOS BASIC files can be loaded with RDOS.

Some minor problems had to be taken care of, and were resolved as follows:

• A 1024 byte file buffer was required by ProDOS. This was put at $BB00-BEFF, where the RWTS routines used to be.

• Text files under RDOS are never really "open"; they are read or written as long as the input/output hooks ($36-39) point to the text file routines (this is why the &DEF command is necessary; the &READ and &PRINT commands assume that a text file is there). Technically speaking, you could "close" a file with "PR#0:IN#0". To be certain that files are not just left hanging, a generic CLOSE call is made every time an ampersand command is accepted; this keeps PDOS from hanging (only one file can be open at a time; if a text file was left open, then trying to execute almost any other PDOS command would cause a "too many files open" error).

• The &DEF command is still required to create text files, but pre-sizing is not required under ProDOS. The size parameter is simply ignored (I could SET_EOF if it makes anybody feel better).

• While RDOS is device oriented (slots & drives), ProDOS is volume oriented (prefixes & volume names). Code was added to ensure that a prefix would be set, and the slot/drive change commands were drastically altered. Whenever a slot or drive is changed, a ProDOS MLI ONLINE call is made to find out the prefix of the destination drive.

• To take advantage of prefixes, a new command similar to BASIC. SYSTEM's "PREFIX" command has been added: &P"pathname". Use this to change directories (examples are &P"/hd1/rdos/fmc" or &P"Ringside").

• To allow easy exits from RDOS, &STOP will call the ProDOS QUIT code.

• While RDOS catalog sectors are only 256 bytes, ProDOS directory blocks are 512 bytes. This meant using the entire buffer space from $B100-B2FF would have to be used, half of which was formerly occupied by the disk catalog code (recall that it was read from track 1 sector 12). The catalog code had to be part of the main program, not read in from disk; this cramped things a bit.

• Even though I had an extra page ($BA00-BAFF) of memory, the catalog code took up twice as much room as the original (ProDOS is a bit more complicated than RDOS!). The BASIC chaining code had to go somewhere, and is now kept in a file called "CHAINSTUFF" (it used to reside on track 1 of the RDOS disk). If &GOTO is failing, make sure that there is a copy of CHAINSTUFF in the current directory.

• RDOS filenames use DOS 3.3 syntax (i.e., spaces and punctuation are allowed), and may be 24 characters long. ProDOS only allows numbers, letters, and a period, and has a 15 character maximum. The filename interpreter automatically converts illegal characters to a period. PDOS v1.1 includes a length truncator, which silently truncates every name to 15 characters. Unfortunately, inclusion of this forced me to remove the &LEN command to make space (PDOS v1.0 had a program to do this for you, but I decided it was better to make it standard).

Entering and Using PDOS

(This section is for people entering PDOS from Computist.)

If you want to type in the hex dump, type it in at $2000 and:

CREATE PDOS.SYSTEM,TSYS
BSAVE PDOS.SYSTEM, A$2000, L2390, TSYS

Type in the "CHAINSTUFF" file at $1000 and

BSAVE CHAINSTUFF, A$1000, L205

Note to people with source code: the old EDASM source code used to create three files, which had to be glued together by hand. The v1.1 Merlin source code handles everything nicely, automatically creating PDOS.SYSTEM and CHAIN STUFF.

To use PDOS, execute it as you would any other system program. After running, it will try to execute the BASIC commands in a text file called "SYSTEMBOOT" (this is the standard SSI method). If it isn't found, a message will be displayed telling you so. Note that PDOS now supports the ProDOS "startup protocol"; if you use a program launcher like ProSel, you can tell PDOS to execute a text file other than "SYSTEMBOOT."

The ideal situation is a disk with ProDOS, PDOS, and CHAINSTUFF as the first three files, followed by the program files. If the files must be kept in a subdirectory, put PDOS and SYSTEMBOOT in the volume directory and CHAINSTUFF in the subdirectory. Put a HELLO program in the volume directory that changes the prefix (&P"...") and &RUNs the true HELLO program (it can serve as a menu on high-volume media).

PDOS Updates

Changes for v1.1 of PDOS:

• The &len command was removed to make room for other features.

• A filename truncator was added (so filenames longer than 15 characters are silently chopped).

• The ProDOS "startup" feature is supported, allowing text files other than "SYSTEMBOOT" to be executed on initial startup.

• Some bugs involving text files were squashed.

• The initial text message was expanded to show version information. Note that there are two important version numbers, the version of PDOS and the version of RDOS that is being emulated.

Warning: Because of the modifications, all internal locations are different. This means that the original deprotection for Computer Ambush (which stuffed filenames directly into the filename buffer) will not work with this version. You can either update your copy of Computer Ambush with the information in the "Examples" file, or just keep using PDOS v1.0 with CAM.

Credit goes to Evin Mulron for finding and reporting the bugs.

Changes for v1.1 of RDOS Transfer:

• Important: I renamed RDOS 3.3 to RDOS 3, and RDOS 3 to RDOS 3.3. The "manual" has been updated to reflect this. It's easy to remember: RDOS3.2 works with the DOS 3.2 version of RDOS; RDOS3.3 works with the DOS 3.3 version of RDOS; RDOS3 works with the Krakowicz cracked version.

• Added INPUT statements to prompt the user for the version and the prefix (it is no longer necessary to modify Transfer). Automatically selects 13 or 16 sectors based on which version is specified.

• Altered the messages and comments to be slightly clearer. I can't add much more; there's only about 30 bytes of space left before it starts being stepping on.

• Changed the "press return to begin" prompt to an INPUT statement, so now you can stop the program with ctrl-C at that point.

• Rewrote TRANSUBS because a change to Basic.System caused all of the auxtype fields to be set to $2100. Also dramatically improved error reporting (errors are now reported by Basic.System instead of TRANSUBS, so it prints a text message and stops the program instead of printing a hex number and dropping into the monitor).

• Fixed a bug which caused Transfer to crash on files larger than about 100 DOS 3.3 blocks (25K). It no longer crashes, and it will correctly handle files up to about 200 blocks (50K). Since RDOS can't read pieces of files, this limit should be sufficient.

• Fixed it to handle lower case filenames (used to turn all lower case letters to '.').

• Fixed it to display names when it truncates them.

• Patched "RDOS3.3" so that it is no longer necessary to use the softkey from Computist #51 (which used COPYA to make a readable copy) before transferring RDOS 3.3 disks. Now just use "3.3+" format.

Credit goes to Evin Mulron for finding and reporting most of these bugs, and for testing the new version.

RDOS Transfer Utility v1.1

(This is an updated copy of the article, which originally appeared in Computist #52, pages 24-29.)

Requirements:
PDOS
Old System Master
ProDOS

A few years back, the folks over at Strategic Simulations, Inc. came up with a modified DOS called RDOS. It used DOS 3.2 and had a double boot loader so that it would work on DOS 3.2 and 3.3 compatible drives.

This DOS can be found on a few dozen of SSI's products. Because it is a very terse, limited operating system, file examination and modification can be difficult. It would be much easier to edit the files under ProDOS.

How RDOS Transfer Works

This section is rather technical, and is not necessary to understand in order to use Transfer.

Files under RDOS are stored in sequential blocks (256 byte, not ProDOS 512 byte blocks); the catalog entry contains the first block and the number of blocks in the file. This is similar to Apple's UCSD Pascal operating system.

Each entry in the catalog is 32 bytes long, and has the following format:

00-23	The filename, padded with trailing spaces
24	File type (ASCII character A, B, or T)
25	Number of blocks used by this file
26-27	Load location (lo, hi format)
28-29	File length (lo, hi format)
30-31	First block (lo, hi format)

The first file on every disk (for RDOS 2.1) is "RDOS 2.1 COPYRIGHT 1981". This occupies the first 26 blocks (tracks 0 and 1 of a 13-sector disk), and includes the catalog track and RDOS itself. The catalog occupies track 1, sectors 0 to 10. Sector 12 contains the code that actually catalogs the disk, and sector 11 contains the code used to chain Applesoft programs.

Since files are contiguous (unlike ProDOS, you can't have parts of a file scattered about the disk), a deleted file is marked as unused space. The first byte of the name is set to $80, and the type is set to $A0 (a blank space). The next file which uses a deleted directory slot also inherits the entire region that the deleted file occupied.

Transfer starts by asking for a version number and a destination prefix. These are explained later. RDOS uses the version number to set various parameters, and load the appropriate RWTS file.

Next, Transfer reads the entire catalog into a buffer from $E00 to $18FF (line 140). This buffer is immediately after the program and immediately before the RWTS (DO NOT add much to the program, or the end of it will be overwritten by the catalog buffer!).

Line 150 sets the current entry pointer (CE) to the second entry in the catalog (no need to transfer over the DOS and disk catalog). Line 155 looks for a blank entry, and if it finds one, it skips it.

Line 160 calls the subroutine at 1000, which gets the file information:

N$ = "raw" filename
NAME$ = ProDOS - compatible file name
T$ = type
BL = number of blocks
LD = load location
LN = file length
SB = start block

NAME$ is derived by taking N$, stripping the trailing blanks, and converting all illegal characters to ".". If there are no more entries (NAME$ = ""), the program ends at line 300. Line 162 prints information about the current file, and line 166 prints the old file name if it doesn't match the new one.

Lines 170-200 create a file of the appropriate type. Line 200 also sets up the track and sector numbers, and initializes the buffer pointer.

Lines 210-225 are the heart of the program: they translate the block numbers into tracks and sectors, and read in the entire file. The call to RW is a short machine language routine (in TRANSUBS) which calls the RWTS routine. Line 220 handles files that are exceptionally large; when the buffer is completely full, it saves the portion of the file that is in memory, and resets the buffer pointer.

Lines 230 calls another part of TRANSUBS which sets the auxiliary type of the file. It was necessary to use a special program because it is impossible to set the AUX_TYPE bytes from Basic. System (as of Basic.System v1.4 or so, the BSAVE command resets the AUX_TYPE every time a file is saved; this caused Transfer to give all transferred files an AUX_TYPE of $2100. Credit goes to Evin Mulron for finding this bug).

(Note that the AUX_TYPE holds the load location of programs, and is used by Basic.System when altering certain absolute pointers in Applesoft programs. Although RDOS would be able to read the files without difficulty if these bytes weren't set, ProDOS's Basic.System wouldn't be able to).

Because most of memory is needed to copy files, line 235 garbage-collects the variables after each pass. Line 240 moves the pointer to the next file, and loops back.

Entering Transfer

(This section is for people entering the programs from Computist magazine.)

Type in the Transfer program (under ProDOS), and SAVE it. Next, enter the TRANSUBS program and BSAVE TRANSUBS, A$300, L174. Now you must get an old DOS RWTS routine.

If you intend to transfer DOS 3.2 programs, break out the old MUFFIN program on the system master (hope you have one... sigh). You need to BLOAD MUFFIN, and then BSAVE RDOS3.2, A$1900, L$800. This should be transferred to the same directory as Transfer (use the ProDOS utilities or Copy II Plus).

If you want to convert programs cracked with "RDOS 3.3" (courtesy Krakowicz or COMPUTIST issue #30; I'll call it "RDOS 3" from here on), or wish to use the newer 16-sector SSI version of RDOS, boot your system master, and allow it to load integer

BASIC. Then relocate the DOS 3.3 RWTS as follows:

INT
CALL-151
D4D5G *Initialize the relocator*
1900<B800.BFFF ctrl Y* *Define the source block*
1900<B800.BA10 ctrl Y *Relocate some code*
.BC55M *Move some stuff*
.BFA7 ctrl Y
.BFC7M
.BFFF ctrl Y
20B8:0 2 4 6 8 A C E 1 3 5 7 9 B D F
BSAVE RDOS3.3, A$1900, L$800 *For the "real" RDOS 3.3*
1F2A:EA EA EA EA *Cancel sector interleaving*
BSAVE RDOS3, A$1900, L$800 *For the Krakowicz version*

Then, transfer the RWTSs over to the ProDOS Transfer disk.

Transferring Programs

To transfer files, run the Transfer program. You will be prompted for a version number, which tells Transfer how the disk is formatted. The possible choices are:

3.2	Standard DOS 3.2 (13 sectors). This is used occasionally.
3.2+	Modified DOS 3.2 format. This is the most common format, and was the default format used by the old version of Transfer.
3.3	Standard DOS 3.3 (16 sectors).
3.3+	Modified DOS 3.3 format. This is commonly used on newer games. If the softkey from Computist #51 works on your disk, then the disk is in 3.3+ format.
3	Krakowicz (already cracked) format. This is actually a 13-sector format on a 16-sector disk.

Transfer will automatically select 13 or 16 sectors based on the version number, and will patch the RWTS routines as necessary.

Next, you will be asked for the destination prefix. Enter the name of a ProDOS directory (you must have created it already; Transfer does not create subdirectories). All of the transferred files will be placed there.

I suggest that you transfer them to a newly formatted disk, into an empty subdirectory (volume directories hold only 51 files). High-capacity RAM disks work beautifully.

As the files are copied, information about them will be printed. If Transfer must change a filename, the old name will be printed on the line below in parenthesis. In 80 columns, it looks something like:

```
"RSS.RING.TEXT"            TYPE B, 10 BLOCKS, START = 102
(WAS: 'RSS RING/TEXT')
```

You should write down any filenames with a slash ('/') - references to these MUST be changed within the programs. Filenames with blank spaces or other characters which are illegal under ProDOS will be changed, and these changes are automatically recognized by PDOS. Filenames longer than 15 characters are silently truncated by PDOS. Filenames with a "/" in them CANNOT be fixed, because PDOS has no way of knowing if it's a legal filename with a slash or a legitimate attempt to access a file in a subdirectory.

(If it seems reasonable, I may make '/' illegal for everything except the &P (prefix) command in a future version.)

After it finishes, you can see how much space is left. If you are transferring it to a 5.25 inch disk, you will need room for the ProDOS image (32 blocks) and the PDOS.SYSTEM file (6 blocks), and a copy of CHAINSTUFF (1 block).

You should then set up the destination disk, placing a copy of CHAINSTUFF in the directory. Copy the files over, and make modifications to the following:

HELLO program : add the following line. SYSTEMBOOT changes several page 3 vectors; this changes them to something more appropriate. It changes the DOS warmstart vector ($3D0) to $B300, the RWTS vector to the monitor, and the reset vector to basic ($E003). It also clears the run flag (214).

```
POKE 977,0: POKE 978,179:
  POKE 986,89: POKE 987,255:
  POKE 1010,3: POKE
  1011,224: POKE 1012,69:
  POKE 214,0
```

It is usually best to leave SYSTEMBOOT unchanged; sometimes it has important stuff in it.

QWERTY (@WERTY, QWERTY.4) : determine the start address, BLOAD the program, store A9 00 85 00 60 at the very start, and BSAVE it (for budding assembly programmers, this stores a 0 in address zero, and returns). This removes the secondary copy protection. It is also usually possible to just delete the lines in the HELLO program which call QWERTY, but some SSI programs call it again later on, so it is probably best to alter QWERTY itself.

Note: the traditional methods for killing QWERTY (allowing it to execute, but always returning a valid result) will not work. These are unacceptable, because QWERTY makes direct calls to RWTS routines which don't exist under PDOS. Thus, it is important that the call never be made in the first place.

SSI.INIT : this formats disks to RDOS format. Since it requires the RDOS RWTS, it won't work; even if it did, the disks couldn't be used from ProDOS. BLOAD SSI.INIT, store a $60 (RTS) at the program start (usually $800), and BSAVE SSI.INIT. This will prevent you from accidentally reformatting your disks.

Since you can't initialize save game disks, make sure that you have a formatted ProDOS volume before you start. You must either use a disk with the same volume name as the program, or save games with names like "/PROG/GAME".

If you noticed that some of the original file names contained a slash, you must check the Applesoft files for the places where they are referenced. If not fixed, the errors could cause the program to crash or hang.

See "RDOS Examples" for a list of RDOS-compatible programs and detailed explanations for transferring several of them.

Possible Problems While Copying

Sometimes during a transfer, the program will print "PRODOS ERROR:" followed by a two-digit hex number, and fall into the monitor. The error is likely one of the following:

$27 - i/o error. Something is wrong with the destination volume.

$2B - write protected.

$2E - volume switched.

$40 - invalid pathname syntax.

$44 - nonexistent path.

$47 - duplicate file name. Do not try to copy files into the same directory as a bunch of other RDOS files. Could be caused by having two similar RDOS files (like "GAME-A" and "GAME+A") which get converted to the same name.

$48 - disk full. Not enough space - make sure disk is empty (no ProDOS file or other system files).

$49 - volume directory full. Too many files; use a subdirectory.

$52 - not a ProDOS disk. What were you thinking?

$57 - duplicate volume online.

Version 1.1 of Transfer was altered slightly, so now most errors will be reported by Basic.System (so you'll see "DUPLICATE FILE NAME BREAK IN 230" instead of "$47" followed by a crash into the monitor).

Common Problems

If you think you've done everything right, and the program still won't work, make sure there's a copy of CHAINSTUFF in the directory. If you get a "FILE NOT FOUND" error from an &GOTO command, this is probably why.

Closing Notes

PDOS and Transfer allow you to put SSI programs on virtually any type of storage, from 5.25" floppies to 100MB hard disks. I was able to put Ringside Seat, Computer Ambush, Operation Market Garden, and Phantasie all onto a single 3.5 inch disk, and still had 280 blocks free - enough for a single-sided game.

I was also able to move the programs over to a RAM disk. Combined with an accelerator card or //gs fast mode, even the slowest of SSI's programs take on a new life.

Related product: find a copy of RKCrack (from Computist #70), and you can put Germany 1985, RDF 1985, Baltic 1985, Norway 1985, and the original Reach for the Stars on a ProDOS disk with all the rest of your SSI games. It's the same concept as PDOS, but MUCH simpler (took a day to write).

PDOS Examples

Requirements:
PDOS
RDOS Transfer

PDOS-Compatible Software

The following programs can be transferred:

Battle Cruiser
Battle For Normandy
Battle Group
Battle of Antietam
Battle of Shiloh
Breakthrough in the Ardennes
Bomb Alley
Carrier Force
Cartels and Cutthroats
Computer Air Combat 1.1
Computer Air Combat Data Disk
Computer Ambush 2.0
Computer Baseball
Computer Bismarck 1.1
Cosmic Balance
Cosmic Balance II
Cytron Masters
Eagles
Epidemic
Fifty Mission Crush
Fighter Command
Galactic Adventures
Galactic Gladiators
Geopolitique 1990
Guadalcanal Campaign
Imperium Galactum
Kampfgruppe
Kampfgruppe Scenario Disk One
Knights of the Desert
Mech Brigade
Napoleon's Campaigns:1813 & 1815
North Atlantic '86
Objective: Kursk
Operation: Market Garden
Phantasie
President Elect
Pursuit of the Graf Spee
Reforger '88
Ringside Seat
Road to Ghettysburg
Roadwar 2000
Roadwar Europa
Six-Gun Shoot Out
Tigers In The Snow
Torpedo Fire
U.S.A.A.F.
War In Russia
Warp Factor
Warship

The following have problems:

Broadsides (###)
Computer Conflict
Field of Fire (###)
Fortress (###)
Nam (###)
Operation Apocalypse
Panzer Grenadier (###)
Rails West (this one is tricky)

(### means that Transfer doesn't work at all, possibly because a different disk format is used. The others are just stubborn.)

The following use 64K of memory, and will not work under 48K PDOS (look for a 64K version of PDOS soon):

Battles of Napoleon
B-24
Colonial Conquest
Ghettysburg: The Turning Point
Overrun
Panzer Strike
President Elect '88
Rebel Charge at Chickamagua
Sons of Liberty
Typhoon of Steel

The following use 128K of memory, and simply aren't going to work (some of the above may also require 128K; I haven't checked them all):

War in the South Pacific

A Few Examples

In the following, the program name is followed by the disk format type (3.2, 3.3, 3.2+, 3.3+), and then the initials of the person who got the information. (MMM) means that I did the deprotection (see Computist #52), and (EM) means that the information came from Evin Mulron's article (see Computist #76).

(I was unable to verify the format on many of these, so I tried to guess; the ones I'm uncertain about are followed with a '?', as in (3.2+?).)

Note: all of these assume that you have followed the procedure in the Transfer document. Most of these games have secondary copy protection (i.e. QWERTY) which must be deactivated.

Battle Cruiser

(3.3+) (EM) Runs without further modification.

Battle of Antietam

(3.3+) (EM) In this program, you need the following lines added, in order to catalog your save game disk:
In "G", add `& CAT: GOSUB 5000`to the beginning of line 2000
In "LOADER", add `&CAT : GOSUB 49` to the beginning of line 2000

Battlegroup

(3.2+?) (EM) Change line 200 in VECTOR.P to read `200 A$ = "COMBAT": GOTO 95`(deleting `&RECALL "ARSENAL": CALL 516` you may delete the file ARSENAL).

Bomb Alley

(3.2+?) (EM) In order to run a saved game, you must change line 45020 in HELLO to read: `45020 &RECALL "P." + N$,640: FOR XX = 0 TO 7: POKE 105 + XX, PEEK(640+XX) : NEXT: &RECALL N$`

Breakthrough in the Ardennes

(3.2+?) (EM) In order to catalog your save game disk change the following: In "LOADER", change line 130 to `GOSUB 12000 : IF A = 204 THEN 2030`. In "H", change line 957 to `GOSUB 12000 : IF A = 204 THEN 2030`

Carrier Force

(3.2+?) (EM) In order to run a saved game, you must change line 45020 in VSTART to read 45020 &RECALL "P." + N$,640: FOR XX = 0 TO 7: POKE 105+XX, PEEK(640+XX): NEXT: &RECALL N$

Computer Ambush v2.0

(3.2+) (MMM) When this was rewritten in machine language, it was interfaced directly with RDOS. Since it bypasses the string entry routines, filenames cannot be screened for illegalities. Take care when saving games (use short names and characters that are legal under ProDOS). This program requires a few patches.

Note: these values are slightly different from those for version 1.0 of PDOS, since some parts of PDOS have been rearranged. If you are using a copy of Computer Ambush modified to work with PDOS v1.0, you will need to make these changes again.

Rename HT2, HT *Higher Text 2*
HELLO : change "HT2" and "INIT PHASE" to "HT" and "INIT.PHASE" in line 100. "HT2" had to be abbreviated to leave space in the programs for storing the length byte.
SWITCH : this is used to switch between the main programs. Do the following:

BLOAD SWITCH
CALL-151
84C:B6 BA *change the filename buffer*
854:B6 BA
857:8C B5 BA EA EA EA EA EA *set the name length*
861:10 B5 *jump to the PDOS readfile routine*
BSAVE SWITCH

OIP : (Order Input Phase) this tries to load the file "HT2" after loading.

BLOAD OIP
CALL-151
4003:B6 BA *Change the filename buffer*
4008:B7 BA
400B:02 *Set the name length to 2*
400D:B5 BA
4011:10 B5 *Read the file*
BSAVE OIP

RP : (Report Phase) not only does this try to load "HT2", but it also tries to save the game.

BLOAD RP
CALL-151
4017:B6 BA *Same as OIP*
401C:B7 BA
401F:02
4021:B5 BA
4025:10 B5

(The following is necessary only to save games...)

6194:13 B9 *re-route onerr*
6199:14 B9
641E:00 *Change "delete" code to zero*
6552:86 00 EA *Store save/delete code*
6581:B6 BA *Filename buffer*
658D:B6 BA
6595:8C B5 BA A5 00 D0 06 20 39 B6 4C B5 65 A9 00 85 50 8D BA B9 A9 9C 8D BB B9 A9 14 85 51 20 FE B4 A9 00 8D 72 63 A5 00 *I actually didn't modify much, but everything had to be shifted by three bytes.*
BSAVE RP

One final note: make sure that you use the original SYSTEMBOOT file with this and any other program that uses Higher Text 2; it calls $3EA and possibly some other page 3 vectors that are initialized by SYSTEMBOOT.

Fighter Command

(3.2+?) (EM) In order to run a saved game, you must change line 500 in HELLO to read, `500 &RECALL N$ + ".F", 640: FOR XX = 0 TO 7: POKE 105 + XX, PEEK (640 + XX): NEXT: &RECALL N$`

In order to use the catalog function in the save game menu, I had to change line 10 to read `10 IF GM = 67 THEN PRINT CHR$(12): GOSUB 40: HOME: & CAT : POKE KC,0: GET A$: GOTO 3` (replacing `"CALL 2800: PRINT PC$"` with `"&CAT"`).

Guadalcanal Campaign

(3.2+?) (EM) In order to run a saved game, you must change line 45020 in HELLO to read: `45020 &RECALL "P." + N$,640: FOR XX = 0 TO 7: POKE 105 + XX, PEEK(640+XX) : NEXT: &RECALL N$` Does this look familiar yet...?

Imperium Galactum

(3.2+?) (EM) In order to save games, change line 12005 in IMCOM to read **`12005 &RUN "GSAVE"`** (deleting "POKE DA+98, TU").

North Atlantic 86

(3.2+?) (EM) In order to run a saved game, you must change line 45020 in HELLO to read: **`45020 &RECALL "P." + N$,640: FOR XX = 0 TO 7: POKE 105 + XX, PEEK(640+XX) : NEXT: &RECALL N$`**

To save a game to the same subdirectory, delete "`GOTO 30000`" from the end of line 30000.

Operation Market Garden

(3.2+) (MMM + EM) Runs without modification (or so I thought... +mmm). If moved to high-capacity storage, you may want to eliminate lines 160-162 in the file "LOADER". Make sure both sides have the same volume name.

In order to catalog your save game disk add the following:
In "I", add **`&CAT : GOSUB 5000`**to the beginning of line 2040. In "LOADER", add **`&CAT : GOSUB 1190`**to the beginning of line 2040.

Phantasie

(3.3+?) (MMM) Runs without modification. If moved to high-capacity storage, you only need one set of the "MNn" files.

It may be a good idea to use two subdirectories (or two disks), putting all of the scrolls, dungeon, and town data in one, and the main programs and monster files in the volume directory of the other. There are a large number of files, and ProDOS takes its own sweet time searching through large directories (while this would require modifications to the programs, it would allow two-drive play).

Questron

(3.2+) (MMM) I only took a brief look, but it appears to require several modifications. The main hitch is that the program tries to load the "DISK-n" files at $00FE - illegal under ProDOS, even if you adjust the "memory in use" map. The load address must be changed on these (see line 101 of the HELLO program).

If you plan to move it to high-capacity storage, it would be nice to put each disk in its own subdirectory; replace the drive change commands with &P commands. Try moving disks 1 and 2 to a RAM disk, while leaving disk 0 on a floppy to keep your saved games.

President Elect

(3.2+) (EM) Runs without any modification.

Ringside Seat

(3.2+) (MMM) Requires a change in line 11200 of "PRERSS". Change "RSS.RING/TEXT" to "RSS.RING.TEXT". Boxer data disks must have the same volume name as the boot volume. There will be only 8 free blocks on a 5.25" disk after the transfer is complete.

Roadwar 2000 & Roadwar Europa

(3.3+) (EM) Runs without further modification.

U.S.S.A.F.

(3.2+) (EM) Change line 9930 in COMBAT to read `9930 GOSUB 199: PRINT "ALL RAIDS COMPLETED": GOTO 390`(deleting &RECALL "PH1": CALL 516; you may delete the file PH1).

War in Russia

(onto a 3.5" or hard disk) (3.2+?) (EM) Transfer the files from each side of the disk into different directories. Rename the following files from side 2:

RENAME BRAIN.1 to BRAIN.5
RENAME BRAIN.2 to BRAIN.6
RENAME BRAIN.3 to BRAIN.7
RENAME BRAIN.4 to BRAIN.8

Change the following lines in VECTOR to read:

```
600 A$ = "BRAIN.5" : POKE
  8,0 : GOTO 95
700 A$ = "BRAIN.6" : GOTO 95
800 A$ = "BRAIN.7" : GOTO 95
900 A$ = "BRAIN.8" : GOTO 95
```

Then, copy the contents from side 2 to the directory with all the files from side 1. You can now run the game entirely from one disk or subdirectory.

Warship

(3.3+) (EM) Runs without further modification.

Note to the curious: the reason for the repeated [FOR XX = 0 TO 7] stuff is that the game tried to &RECALL the saved game directly onto page 0 (at location 105). ProDOS refuses to read onto page 0, so it was necessary to read the file onto page 2 (the input buffer) and then copy the data over.

TRANSFER

```
10 LOMEM: 34304: REM $E00-
  $85FF
100 TEXT : NORMAL : HOME
110 PRINT "RDOS◊TRANSFER◊
  V1.1◊-◊BY◊M.M.◊MCFADDEN" :
  PRINT : INPUT "FORMAT◊
  (3.2[+],◊3.3[+],◊3)?◊"
  ;V$:V = VAL (V$):F =
  RIGHT$ (V$,1) = "+"
114 INPUT "DESTINATION◊
  PREFIX?◊" ;P$: IF RIGHT$
  (P$,1) < > "/" THEN P$ =
  P$ + "/"
116 D$ = CHR$ (4):SE = 13:
  IF V = 3.3 THEN SE = 16:
  REM #OF SECTORS
120 PRINT D$ "BLOADTRANSUBS"
  : PRINT D$ "BLOADRDOS"
  V:RW = 771:IOB = 782:TR =
  IOB + 4:SC = IOB + 5:BF =
  IOB + 9:LP = 14:MP = 134 -
  33: REM $8600-$2100
123 IF V = 3.2 AND F THEN
  POKE 6774,212: POKE
  6795,183
125 IF V = 3.3 AND F THEN
  POKE 6722,24
130 PRINT : INPUT "PUT◊RDOS◊
  DISK◊IN◊S6,D1◊AND◊HIT◊
  RETURN" ;A$: PRINT
140 POKE TR,1: FOR A = 0 TO
  10: POKE SC,A: POKE BF,LP
  + A: CALL RW: NEXT : REM
  READ CAT AT $E00-$18FF
150 BS = 33:CE = 3584 + 32:
  REM DATA @$2100
155 IF PEEK (CE) = 128 OR
  PEEK (CE + 24) = 160 THEN
  240: REM DELETED
160 GOSUB 1000: IF NAME$ =
  "" THEN 300
162 PRINT CHR$ (34)NAME$
  CHR$ (34);: HTAB 41: PRINT
  "TYPE◊" T$ ",◊" BL "◊BLOCKS
  ,◊START◊=◊" SB
166 IF OL$ < > NAME$ THEN
  PRINT "(WAS:◊'" OL$ "')"
170 IF T$ = "A" THEN TY$ =
  "BAS"
180 IF T$ = "B" THEN TY$ =
  "BIN"
190 IF T$ = "T" THEN TY$ =
  "TXT"
200 PRINT D$ "CREATE"
  P$NAME$ ",T" TY$:T = INT
  (SB / SE):S = SB - T *
  SE:B = BS:BB = 0: FOR A =
  1 TO BL
210 POKE TR,T: POKE SC,S:
  POKE BF,B: CALL RW:B = B +
  1:S = S + 1: IF S > (SE -
  1) THEN S = 0:T = T + 1
220 IF A = MP THEN BB = MP *
  256: PRINT D$ "BSAVE"
  P$NAME$ ",A" BS * 256 ",L"
  BB ",T" TY$:B = BS:LN = LN
  - BB
225 NEXT
230 PRINT D$ "BSAVE" P$NAME$
  ",A" BS * 256 ",L" LN ",T"
  TY$ ",B" BB: CALL 768,P$ +
  NAME$,LD
235 PRINT D$ "FRE"
240 CE = CE + 32: GOTO 155
300 PRINT "DONE!" CHR$ (7):
  END
```

```
1000 N$ = "" : FOR A = 0 TO
  23:N = PEEK (CE + A): IF N
  THEN N$ = N$ + CHR$ (N -
  128): NEXT
1010 IF A = 0 THEN NAME$ =
  "" : GOTO 1060
1020 A = 24: FOR A = 24 TO 1
  STEP - 1: IF MID$ (N$,A,1)
  = "◊" THEN N$ = LEFT$
  (N$,A - 1): NEXT
1025 OL$ = N$:N$ = LEFT$
  (N$,15)
1030 NAME$ = "" : FOR I = 1
  TO LEN (N$):A$ = MID$
  (N$,I,1):A = ASC (A$): IF
  (A < 65 OR A > 90) AND (A
  < 48 OR A > 57) AND (A <
  97 OR A > 122) THEN A$ =
  "." : REM [A-Z][0-9][a-z]
1040 NAME$ = NAME$ + A$:
  NEXT
1050 T$ = CHR$ ( PEEK (CE +
  24) - 128):BL = PEEK (CE +
  25):LD = PEEK (CE + 26) +
  PEEK (CE + 27) * 256:LN =
  PEEK (CE + 28) + PEEK (CE
  + 29) * 256:SB = PEEK (CE
  + 30) + PEEK (CE + 31) *
  256
1060 RETURN
```

Checksums

10-$A92B	160-$5C3F	240-$BB9A
100-$F42C	162-$2AEC	300-$5044
110-$78D2	166-$EA2D	1000-$8BCF
114-$BDEA	170-$4998	1010-$3ADE
116-$510F	180-$51FA	1020-$E0D1
120-$EA19	190-$9F67	1025-$1A4B
123-$3EA0	200-$822B	1030-$FA3C
125-$8E3F	210-$832E	1040-$FBC0
130-$2B29	220-$94B3	1050-$18E5
140-$4F33	225-$D851	1060-$9AB4
150-$EF43	230-$AAAE	
155-$2391	235-$4D23	

TRANSUBS

```
0300:4C 23 03 A9 03 A0 0E 08 $57FC
0308:78 20 00 1E 28 60 01 60 $030E
0310:01 00 01 00 1F 03 00 20 $B7C0
0318:00 00 01 00 00 60 01 00 $A7A4
0320:01 EF D8 20 BE DE A9 55 $3270
0328:85 52 20 7B DD 20 6C DD $22CE
0330:A0 02 B1 A0 99 55 00 88 $61EB
0338:10 F8 C8 B1 56 99 81 02 $AAF9
0340:C8 C4 55 90 F6 A9 0A 8D $8890
0348:8E 03 20 00 BF C4 8E 03 $ADDB
0350:B0 24 20 BE DE 20 67 DD $A84B
0358:20 52 E7 A5 50 8D 93 03 $39ED
0360:A5 51 8D 94 03 A9 07 8D $22F7
0368:8E 03 20 00 BF C3 8E 03 $BA01
0370:B0 04 60 4C 79 E1 48 A0 $EB02
0378:00 B9 A0 03 20 ED FD C8 $0494
0380:C0 0E 90 F5 68 20 DA FD $CD9D
0388:20 DD FB 4C 59 FF 0A 80 $E79E
0390:02 00 00 00 00 00 00 00 $06FE
0398:00 00 00 00 00 00 00 00 $E69E
03A0:D0 D2 CF C4 CF D3 A0 C5 $0486
03A8:D2 D2 CF D2 A0 A4       $3C5B
```

CHAINSTUFF

```
1000:20 84 E4 A9 07 85 8F A5 $2192
1008:69 A6 6A 85 9D 86 9E E4 $E03E
1010:6C D0 04 C5 6B F0 05 20 $5410
1018:71 B1 F0 F3 85 9F 86 A0 $B2FC
1020:A9 03 85 8F A5 9F A6 A0 $1CFE
1028:E4 6E D0 05 C5 6D D0 01 $89C2
1030:60 85 9D 86 9E A0 00 B1 $6024
1038:9D AA C8 B1 9D 08 C8 B1 $3479
1040:9D 65 9F 85 9F C8 B1 9D $8B37
1048:65 A0 85 A0 28 10 D5 8A $07DF
1050:30 D2 C8 B1 9D A0 00 0A $BA4E
1058:69 05 65 9D 85 9D 90 02 $BBA0
1060:E6 9E A6 9E E4 A0 D0 04 $C3C8
1068:C5 9F F0 BC 20 7B B1 F0 $5370
1070:F3 B1 9D 30 46 C8 B1 9D $FF79
1078:10 41 C8 B1 9D F0 3C C8 $E766
1080:B1 9D AA C8 B1 9D 85 9C $429F
1088:86 9B C5 B0 F0 02 B0 2B $5399
1090:88 88 B1 9D 48 38 A5 6F $F728
1098:85 94 F1 9D C8 91 9D 85 $B6FD
10A0:6F C8 A5 70 85 95 E9 00 $E0AD
10A8:91 9D 85 70 68 18 65 9B $6E5D
10B0:85 96 A5 9C 69 00 85 97 $93AB
10B8:20 9A D3 A5 8F 18 65 9D $3F79
10C0:85 9D 90 02 E6 9E A5 9D $24F1
10C8:A6 9E A0 00 60          $488B
```

PDOS.SYSTEM

```
2000:4C 47 20 EE EE 41 0A 53 $9513
2008:59 53 54 45 4D 42 4F 4F $8B87
2010:54 00 00 00 00 00 00 00 $71B7
2018:00 00 00 00 00 00 00 00 $A187
2020:00 00 00 00 00 00 00 00 $71B7
2028:00 00 00 00 00 00 00 00 $A187
2030:00 00 00 00 00 00 00 00 $71B7
2038:00 00 00 00 00 00 00 00 $A187
2040:00 00 00 00 00 00 00 A9 $D8E3
2048:5F 85 FA A9 21 85 FB A0 $A072
2050:00 84 FC A9 B3 85 FD A2 $1569
2058:08 B1 FA 91 FC C8 D0 F9 $8FD1
2060:E6 FB E6 FD CA D0 F2 A2 $19E5
2068:17 A9 C1 9D 58 BF CA A9 $1106
2070:1F 9D 58 BF CA A9 00 9D $91D8
2078:58 BF CA 10 FA A9 00 8D $E8FF
2080:58 BF AD 82 C0 AD 10 C0 $5ED2
2088:A9 03 8D F2 03 A9 E0 8D $A8A9
2090:F3 03 20 6F FB 20 00 B3 $B98C
2098:A0 00 B9 FF 20 F0 06 99 $8AC8
20A0:A8 04 C8 D0 F5 A0 00 B9 $4973
20A8:28 21 F0 06 99 AF 05 C8 $4BF3
20B0:D0 F5 A0 00 B9 43 21 F0 $1940
20B8:06 99 2E 07 C8 D0 F5 AD $765C
20C0:30 BF 8D B7 B9 AD 9A BF $0F2B
20C8:D0 03 20 10 B8 AE 06 20 $FDDD
20D0:BD 06 20 9D B5 BA CA 10 $19EF
20D8:F7 A2 03 B5 36 9D C2 B9 $3E10
20E0:CA 10 F8 A9 F2 A0 20 85 $F296
20E8:38 84 39 A9 0F 85 25 4C $2F03
20F0:82 B6 A2 03 BD C2 B9 95 $45A4
20F8:36 CA 10 F8 4C C5 B5 B4 $01FA
2100:B8 CB A0 D0 C4 CF D3 A0 $048F
2108:A8 D0 F2 EF C4 CF D3 A0 $FA3C
2110:D2 C4 CF D3 A9 A0 A0 C2 $6FA4
2118:F9 A0 CD AE CD AE A0 CD $7AFF
2120:E3 C6 E1 E4 E4 E5 EE 00 $A924
2128:D6 E5 F2 F3 E9 EF EE A0 $07F1
2130:B1 AE B1 A0 A0 CE EF F6 $C2CD
2138:E5 ED E2 E5 F2 A0 B1 B9 $7FDD
2140:B9 B1 00 A8 D3 E9 ED F5 $035E
2148:EC E1 F4 E5 F3 A0 D3 D3 $46AD
2150:C9 A7 F3 A0 D2 C4 CF D3 $1F3E
2158:A0 F6 B2 AE B1 A9 00 4C $61D9
2160:59 B9 A0 00 D9 7E B9 F0 $EEA1
2168:08 C8 C0 13 D0 F6 4C 58 $7B17
2170:FE 98 0A A8 B9 92 B9 48 $B381
2178:B9 91 B9 48 AD CA B9 F0 $68B6
2180:0B 20 4E B9 20 7F B8 A9 $EF70
2188:00 8D CA B9 4C B1 00 C9 $0494
2190:C5 F0 03 4C FA B8 20 00 $1070
2198:BF C7 F0 B9 AD B5 BA D0 $B153
21A0:03 4C 06 B9 A0 00 B9 10 $4087
21A8:BA 20 5C DB C8 C0 24 90 $5296
21B0:F5 20 00 BF C8 F3 B9 AD $99BC
21B8:F8 B9 8D FA B9 8D 0C BA $0FAA
21C0:A9 00 8D FB B9 A9 B1 8D $9647
21C8:FC B9 A9 00 8D FD B9 A9 $CB28
21D0:02 8D FE B9 20 00 BF CA $628C
21D8:F9 B9 90 03 4C C0 B8 A9 $0B12
21E0:14 8D C0 B9 AD 25 B1 85 $018C
21E8:FF AD 23 B1 8D B8 B9 AD $5CD8
21F0:24 B1 85 FE 8D B9 B9 A9 $F5F6
21F8:04 85 FA A9 B1 85 FB 20 $C826
2200:BD B3 20 F0 B3 CE C0 B9 $03AE
2208:D0 08 20 0C FD A9 14 8D $A3F5
2210:C0 B9 C6 FF D0 E9 20 7F $B8FF
2218:B8 4C B1 00 C6 FE F0 15 $74CD
2220:A5 FA 18 6D B8 B9 85 FA $88C9
2228:A5 FB 69 00 85 FB A0 00 $E244
2230:B1 FA F0 E8 60 20 00 BF $C523
2238:CA F9 B9 90 03 4C C0 B8 $2AA7
2240:A9 04 85 FA A9 B1 85 FB $BE0B
2248:AD B9 B9 85 FE D0 DF A0 $1464
2250:10 B1 FA C9 04 F0 13 C9 $31C7
2258:06 F0 12 C9 0F F0 11 C9 $D6D1
2260:FC F0 10 C9 FF F0 0F A9 $76EB
2268:BF 2C A9 D4 2C A9 C2 2C $1981
2270:A9 C4 2C A9 C1 2C A9 D3 $9BE9
2278:20 5C DB 20 57 DB A0 13 $D5E7
2280:B1 FA AA C8 B1 FA A8 D0 $1C59
2288:12 A9 B0 E0 0A B0 03 20 $4A3C
2290:5C DB E0 64 B0 03 20 5C $18D0
2298:DB A9 00 20 24 ED 20 57 $5C50
22A0:DB A0 00 B1 FA 29 0F 85 $2482
22A8:FC E6 FC C8 B1 FA 09 80 $622C
22B0:20 ED FD C8 C4 FC 90 F4 $51B9
22B8:A9 1E 85 24 A0 15 B1 FA $B440
22C0:AA C8 B1 FA 20 24 ED 4C $3109
22C8:FB DA 20 CA B6 4C 03 E0 $26E1
22D0:20 CA B6 4C 66 D5 AC AA $4F6D
22D8:BA 8C B5 BA B9 AB BA 99 $8953
22E0:B6 BA 88 10 F7 A9 00 8D $6153
22E8:BA B9 A9 B1 8D BB B9 20 $6B27
22F0:45 B8 20 00 B1 A0 07 B9 $0D74
22F8:69 00 99 C2 B9 88 10 F7 $3EC7
2300:20 CA B6 A5 6A CD C3 B9 $4E2D
2308:90 0C A5 69 CD C2 B9 90 $6B27
2310:05 F0 03 4C FA B8 A0 07 $E830
2318:B9 C2 B9 99 69 00 88 10 $AFEC
2320:F7 20 97 D6 4C D2 D7 20 $C8C3
2328:4E B7 A5 67 A4 68 8D BA $3128
2330:B9 8C BB B9 38 A5 AF E5 $6A11
2338:67 8D 05 BA A5 B0 E5 68 $60C0
2340:8D 06 BA A9 FC 4C 86 B8 $97A0
2348:20 4E B7 B0 03 4C FA B8 $12D9
2350:A5 50 8D BA B9 A5 51 8D $A8BA
2358:BB B9 20 CA B7 A5 50 8D $51A7
2360:05 BA A5 51 8D 06 BA A9 $182E
2368:06 4C 86 B8 20 4E B7 08 $B58A
2370:A9 06 20 D7 B7 28 90 0A $4FDB
2378:A5 50 A4 51 8D BA B9 8C $46D5
2380:BB B9 4C 45 B8 20 4E B7 $8D01
2388:B0 03 4C FA B8 A9 04 8D $C6D3
2390:CF B9 A9 00 8D D0 B9 A9 $6761
2398:B1 8D D1 B9 20 00 BF 82 $4A3B
23A0:00 00 20 00 BF C0 CB B9 $6BE1
23A8:90 03 4C C0 B8 60 20 4E $F639
23B0:B7 20 43 B9 A9 04 20 D7 $985F
23B8:B7 20 00 BF C8 F3 B9 EE $C104
23C0:CA B9 AD F8 B9 8D 02 BA $3729
23C8:8D 0C BA A9 00 8D 03 BA $599A
23D0:A9 B1 8D 04 BA A9 00 8D $5D4A
23D8:0D BA 8D 0E BA 20 00 BF $87A3
23E0:D0 0B BA A9 95 85 36 A9 $7FCD
23E8:B5 85 37 A9 B4 85 38 A9 $52B7
23F0:B5 85 39 60 48 29 7F 8D $5A94
23F8:00 B1 98 48 8A 48 A0 01 $5FFE
2400:8C 05 BA 88 8C 06 BA 20 $7205
2408:00 BF CB 01 BA 68 AA 68 $F1BD
2410:A8 68 60 48 20 7F B8 20 $6CA4
2418:4E B9 A9 00 8D CA B9 68 $04AE
2420:60 20 4E B7 20 43 B9 A9 $FA35
2428:04 20 D7 B7 20 00 BF C8 $BB42
2430:F3 B9 AD F8 B9 8D FA B9 $BE27
2438:EE CA B9 A9 00 8D FB B9 $9DF8
2440:A9 B1 8D FC B9 A9 F7 85 $213A
2448:38 A9 B5 85 39 A9 2A 85 $B0A4
2450:36 A9 B6 85 37 60 A9 A0 $9932
2458:91 28 A9 01 8D FD B9 A9 $667E
2460:00 8D FE B9 20 00 BF CA $0E3A
2468:F9 B9 90 09 20 B4 B5 20 $C62E
2470:DA FD 4C FA B8 AD 00 B1 $FE93
2478:D0 0C 20 B4 B5 20 DA FD $2C16
2480:20 DD FB 4C 03 B9 09 80 $8021
2488:60 C9 00 D0 06 20 B4 B5 $8999
2490:4C 4E B9 60 60 20 4E B7 $3D46
2498:20 00 BF C1 D7 B9 90 03 $D467
24A0:4C C0 B8 60 60 A2 03 20 $71F9
24A8:FD B7 CA AD B7 B9 29 70 $BE1F
24B0:8D B7 B9 8A 18 6A 6A 0D $85B6
24B8:B7 B9 8D B7 B9 20 10 B8 $DC96
24C0:20 BE DE 4C 03 B3 A2 08 $926D
24C8:20 FD B7 AD B7 B9 29 80 $2FFE
24D0:8D B7 B9 8A 0A 0A 0A 0A $0E5C
24D8:0D B7 B9 8D B7 B9 4C 5E $B44F
24E0:B6 20 43 B9 A9 90 A0 B6 $8B4A
24E8:85 38 84 39 4C 00 E0 91 $299F
24F0:28 A9 00 A0 B1 85 73 84 $22B5
24F8:74 20 4E B9 20 4B D6 4C $543C
2500:03 E0 20 CD B7 8C 0F B3 $D93E
2508:8D 0F B3 60 20 4E B7 20 $48F1
2510:00 BF C6 F0 B9 90 03 4C $15FA
2518:C0 B8 60 20 00 BF 65 C3 $B3D3
2520:B6 00 04 00 00 00 00 00 $4A67
2528:00 A5 67 A4 68 85 50 84 $03D4
2530:51 20 4E B7 A9 FC 20 D7 $B41C
2538:B7 A5 50 38 ED BA B9 85 $9B46
2540:5E A5 51 ED BB B9 85 5F $159C
2548:A5 50 A4 51 8D BA B9 8C $BC22
2550:BB B9 C0 08 B0 03 4C FA $798F
2558:B8 46 D8 20 45 B8 18 A5 $4297
2560:50 85 67 6D 0D BA 85 69 $7CDF
2568:A5 51 85 68 6D 0E BA 85 $2517
2570:6A A9 00 A0 FF C6 68 91 $77B3
2578:67 E6 68 A5 36 A4 37 8D $AEF8
2580:BC B9 8C BD B9 A9 39 A0 $E1CE
2588:B7 85 36 84 37 68 8D C0 $17D7
2590:B9 68 8D C1 B9 4C F2 D4 $F7B0
2598:AD C1 B9 48 AD C0 B9 48 $5632
25A0:AD BC B9 AC BD B9 85 36 $07FB
25A8:84 37 4C 6C D6 A9 55 85 $B909
25B0:52 20 7B DD 20 6C DD A0 $02DB
25B8:02 B1 A0 99 55 00 88 10 $1110
25C0:F8 C8 B1 56 29 7F C9 60 $56BB
25C8:90 03 38 E9 20 C9 20 B0 $1BE2
25D0:03 4C 99 E1 C9 2F B0 02 $DABD
25D8:90 0E C9 3A 90 0C C9 41 $C1C9
25E0:B0 02 90 04 C9 5B 90 02 $7E5D
25E8:A9 AE 99 B6 BA C8 C0 3F $F432
25F0:F0 2E C4 55 90 CC A5 55 $9AAA
25F8:8D B5 BA A8 AA A9 00 85 $5931
2600:71 B9 B5 BA C9 2F F0 06 $BD89
2608:88 D0 F6 8A D0 06 84 71 $E355
2610:8A 38 E5 71 C9 10 90 08 $CFE1
2618:A9 0F 18 65 71 8D B5 BA $28CE
2620:20 B7 00 C9 2C F0 02 D0 $0F59
2628:0B 20 BE DE 20 67 DD 20 $4205
2630:52 E7 38 24 18 60 8D C0 $128B
2638:B9 20 00 BF C4 DE B9 90 $6953
2640:03 4C C0 B8 AD E2 B9 CD $BDC7
2648:C0 B9 F0 03 4C 09 B9 AD $2B6D
2650:E3 B9 8D BA B9 AD E4 B9 $E7E3
2658:8D BB B9 60 8E C0 B9 20 $DB81
2660:F8 E6 E0 01 90 06 EC C0 $ECF9
2668:B9 B0 01 60 4C 99 E1 AD $56BC
2670:B7 B9 8D DB B9 20 00 BF $7CA1
2678:C5 DA B9 90 0B C9 28 F0 $0813
2680:22 C9 27 F0 1E 4C C0 B8 $8274
2688:AD B6 BA 29 0F 8D B5 BA $38D1
2690:EE B5 BA A9 2F 8D B6 BA $D505
2698:20 00 BF C6 F0 B9 90 03 $287E
26A0:4C C0 B8 60 20 00 BF C8 $2BED
26A8:F3 B9 B0 2D AD F8 B9 8D $A7C2
26B0:FA B9 8D 0C BA 20 00 BF $251C
26B8:D1 0B BA AD 0D BA 8D FD $07A9
26C0:B9 AD 0E BA 8D FE B9 AD $9A72
26C8:BA B9 8D FB B9 AD BB B9 $2EEA
26D0:8D FC B9 20 00 BF CA F9 $036F
26D8:B9 90 03 4C C0 B8 20 00 $C0A3
26E0:BF CC 09 BA 60 8D CF B9 $562F
26E8:AD BA B9 8D D0 B9 8D 03 $957A
26F0:BA AD BB B9 8D D1 B9 8D $0B25
26F8:04 BA 20 00 BF C0 CB B9 $58D2
2700:90 03 4C C0 B8 20 00 BF $1482
2708:C8 F3 B9 90 03 4C C0 B8 $FA43
2710:AD F8 B9 8D 02 BA 20 00 $2644
2718:BF CB 01 BA 4C 7A B8 48 $920B
2720:20 7F B8 68 C9 2B F0 43 $CD7D
2728:C9 40 F0 2D C9 44 F0 26 $6AA6
2730:C9 45 F0 22 C9 46 F0 1E $6777
2738:C9 47 F0 20 C9 48 F0 1F $E129
2740:C9 49 F0 1B C9 4C F0 1A $9D19
2748:C9 4D F0 16 C9 55 F0 0F $88C9
2750:C9 57 F0 08 D0 0F A2 01 $EC6E
2758:2C A2 02 2C A2 03 2C A2 $03F1
2760:04 2C A2 05 2C A2 06 2C $04C2
2768:A2 07 2C A2 08 24 D8 10 $E870
2770:03 4C 12 D4 20 FB DA A0 $30B6
2778:FF C8 B9 4A BA 10 FA CA $5130
```

```
2780:D0 F7 20 5A DB C8 B9 4A $97F5
2788:BA 08 20 5C DB 28 30 03 $3CEF
2790:4C 26 B9 A5 76 C9 FF F0 $468A
2798:03 20 19 ED 20 DD FB 4C $42A9
27A0:03 E0 A2 03 B5 36 9D BC $0F26
27A8:B9 CA 10 F8 60 A2 03 BD $91F6
27B0:BC B9 95 36 CA 10 F8 60 $BDE6
27B8:20 89 FE 20 93 FE 20 58 $062C
27C0:FC A9 4C 8D F5 03 A9 03 $041A
27C8:8D F6 03 A9 B3 8D F7 03 $AA7C
27D0:A9 00 85 73 A9 B1 85 74 $2C59
27D8:A9 00 85 F2 60 43 B6 AC $8CC3
27E0:AB B7 A8 A7 B8 BA 87 80 $D6EA
27E8:85 E3 44 53 BF D5 50 B3 $D5BD
27F0:2F B3 6A B4 70 B4 76 B4 $8951
27F8:C7 B4 E8 B4 0C B5 25 B5 $4CAF
2800:4E B5 C1 B5 34 B6 35 B6 $A581
2808:44 B6 45 B6 66 B6 81 B6 $A299
2810:A2 B6 AC B6 BB B6 60 00 $BAB3
2818:00 00 00 00 00 00 00 00 $1A43
2820:00 00 00 00 00 00 00 00 $BAB3
2828:00 00 07 B5 BA C3 00 00 $DEA5
2830:00 01 00 00 00 00 01 B5 $D8CC
2838:BA 02 60 B6 BA 0A B5 BA $A82B
2840:00 00 00 00 00 00 00 00 $A85B
2848:00 00 00 00 00 00 00 01 $A92B
2850:B5 BA 03 B5 BA 00 BB 00 $C3B9
2858:04 00 00 B1 00 00 00 00 $B459
2860:04 00 00 B1 00 00 00 00 $7339
2868:01 00 02 00 00 00 00 A0 $724A
2870:A0 CC C5 CE A0 A0 A0 A0 $6C96
2878:A0 A0 A0 A0 A0 AD BC CE $41C1
2880:C1 CD C5 BE AD A0 A0 A0 $C8F3
2888:A0 A0 A0 A0 CC C5 CE C7 $DF96
2890:D4 C8 8D D3 D4 C1 D2 D4 $B8D4
2898:BA 00 CC C5 CE C7 D4 C8 $7EBF
28A0:BA 00 CC CF CD C5 CD BA $2BCD
28A8:00 80 46 49 4C 45 20 4E $00D8
28B0:4F 54 20 46 4F 55 4E C4 $C8AD
28B8:44 4F 53 20 53 59 4E 54 $FF08
28C0:41 58 20 45 52 D2 44 55 $C4FC
28C8:50 4C 49 43 20 45 4E 54 $B7D6
28D0:52 D9 44 49 53 4B 20 46 $DEBD
28D8:55 4C CC 4F 55 54 20 4F $ECA5
28E0:46 20 44 41 54 C1 49 2F $1665
28E8:4F 20 45 52 D2 46 49 4C $0C30
28F0:45 20 54 59 50 45 20 45 $01DB
28F8:52 D2 57 52 49 54 45 20 $87DE
2900:50 52 4F 54 45 43 54 45 $F69A
2908:C4 0A 43 48 41 49 4E 53 $AAF8
2910:54 55 46 46 00 00 00 00 $24E9
2918:00 00 00 00 00 00 00 00 $E439
2920:00 00 00 00 00 00 00 00 $24E9
2928:00 00 00 00 00 00 00 00 $E439
2930:00 00 00 00 00 00 00 00 $24E9
2938:00 00 00 00 00 00 00 00 $E439
2940:00 00 00 00 00 00 00 00 $24E9
2948:00 00 00 00 00 00 00 00 $E439
2950:00 00 00 00 00 EA       $9CF0
```

TRANSUBS Source

```
* RDOS Transfer subroutines *
* By M.M. McFadden
* v1.1 November 1991
* Merlin assembler format
* Adapted from:
* RDOS Transfer subroutines
* v1.0 August 1987
         lst    off
* Subr #1: ProDOS "AUX_TYPE" command
* Calling conventions (from Applesoft):
* call 768, "filename", aux_type
* (due to a change in BASIC.SYSTEM, the original
* subroutine no longer works):
* Subr #1: augmented ProDOS "CREATE" cmnd
* Calling conventions (from Applesoft):
* call 768,"filename",{A,B,T},start addr
* Subr #2: DOS 3.3 RWTS call
* Calling conventions (from Applesoft):
* Poke track, sector, buffer (determine addr)
* call 771
         org    $300
linnum   equ    $50
lastpt   equ    $52
strscr   equ    $55
facmo    equ    $a0
getchr   equ    $b1
get_file_info   equ   $c4
set_file_info   equ   $c3
namebuf  equ  $280
rwts     equ  $1e00  ;relocated RWTS routine
mli      equ  $bf00
frmnum   equ  $dd67
chkstr   equ  $dd6c
frmevl   equ  $dd7b
chkcom   equ  $debe
illerr   equ  $e179
getadr   equ  $e752
bell     equ  $fbdd
prbyte   equ  $fdda
cout     equ  $fded
monitor  equ  $ff59
         jmp  begin
         lda  #>iob   ;rwts routine
         ldy  #<iob
         php
         sei
         jsr  rwts
         plp
         rts
* DOS 3.3 RWTS IOB
iob      dfb  $01
         dfb  $60,$01
         dfb  $00
         dfb  $01     ;track
         dfb  $00     ;sector
         dw   devchr
         dw   $2000  ;buffer
         dfb  $00,$00
         dfb  $01     ;read
         dfb  $00
         dfb  $00,$60,$01
devchr   dfb  $00
         dfb  $01
         dfb  $ef,$d8
* ProDOS AUX_TYPE routine
begin    jsr  chkcom
         lda  #$55
         sta  lastpt
         jsr  frmevl  ;evaluate formula
         jsr  chkstr  ;make sure it's a string
         ldy  #$02    ;copy string address & length
subs1    lda  (facmo),y
         sta  strscr,y
         dey
         bpl  subs1
         iny
:copy    lda  (strscr+1),y
         sta  namebuf+1,y
         iny
         cpy  strscr
         bcc  :copy
* (this was the part that processed the file type)
*subs3 lda strscr
* sta namebuf
* jsr chkcom
* cmp #$42
* beq type1
* cmp #$54
* beq type2
* cmp #$41
* beq type3
* bne error
*type1 lda #$06
* dfb $2c
*type2 lda #$04
* dfb $2c
*type3 lda #$fc
* sta Create_prm+4 ;file type
* jsr getchr
         lda  #$0a    ;parameter count for get_file_info
         sta  file_info
         jsr  mli     ;get the file info
         dfb  get_file_info
         dw   file_info
         bcs  proerr
         jsr  chkcom
         jsr  frmnum
         jsr  getadr
         lda  linnum  ;change the aux type
         sta  file_info+5
         lda  linnum+1
         sta  file_info+6
         lda  #$07    ;parameter count for set_file_info
         sta  file_info
         jsr  mli     ;set the file info
         dfb  set_file_info
         dw   file_info
         bcs  proerr
         rts
error    jmp  illerr
* Report a ProDOS error message
proerr   pha
         ldy  #00
:loop    lda  errmsg,y
         jsr  cout
         iny
         cpy  #errmsg_e-errmsg
         blt  :loop
         pla
         jsr  prbyte
         jsr  bell
         jmp  monitor
file_info dfb $0a
         dw   namebuf
         dfb  $00
         dfb  $00
         dw   $0000  ;aux_type
         dfb  $00
         dw   $0000
         dw   $0000
         dw   $0000
         dw   $0000
         dw   $0000
errmsg   asc  "PRODOS ERROR $"
errmsg_e ;locate end of errmsg
         lst  on
         typ  $06     ;use "bin" file type
         sav  TRANSUBS    ;TRANsfer SUBroutineS
         lst  off
```

PDOS SYSTEM Source

```
* 48K PDOS v1.1 November 1991
* (ProDOS RDOS 2.1)
* By M.M. McFadden
* Merlin assembler format
         lst    off
* Adapted from:
* ProDOS RDOS 2.1
* By M.M. McFadden
* v1.0 August 1987
* 48K PDOS memory map:
* $0000-b0ff Program usage
* $b100-b2ff Data buffer
* $b300-baff Main PDOS code
* $bb00-beff ProDOS file buffer (1K)
* $bf00-bfff ProDOS system global page
* This code is VERY cramped; v1.1 is about four
* bytes away from walking on the ProDOS file buffer.
* Note that &LEN was dropped from v1.1.
* Brief note on PDOS text files:
* All text I/O is driven by the cswl/kswl. PDOS
* sets up the keyboard vectors, and returns; if
* the program issues an ampersand command at any
* point, then the chances are good that they forgot
* to close the text file. Thus, the text file
* status is checked every time a command is executed.
* There's really no way around it; if the text file
* is open, then the ProDOS buffer is busy, and we
* can't do anything with other files anyway...
* zero-page equates (mostly Applesoft)
ch       equ  $24
cv       equ  $25
basl     equ  $28
cswl     equ  $36
kswl     equ  $38
linnum   equ  $50
strscr   equ  $55
index    equ  $5e
texttab  equ  $67
vartab   equ  $69
arytab   equ  $6b
strend   equ  $6d
fretop   equ  $6f
frespc   equ  $71      ;used as tmp by getinstr
memsize  equ  $73
curlin   equ  $76
scrub    equ  $8f
highds   equ  $94
hightr   equ  $96
lowtr    equ  $9b
dsctmp   equ  $9d
facmoh   equ  $9f
chrget   equ  $b1
chrgot   equ  $b7
facmo    equ  $a0
prgend   equ  $af
errflg   equ  $d8
traceflg equ  $f2
ptr      equ  $fa      ;(2b)
ptr2     equ  $fc      ;(2b)
ctr      equ  $fe      ;used by c_cat
ctr2     equ  $ff      ;used by c_cat
* ProDOS MLI call numbers
Quit     equ  $65
GetTime  equ  $82
Create   equ  $c0
Destroy  equ  $c1
GetInfo  equ  $c4
Online   equ  $c5
SetPrefix equ $c6
GetPrefix equ $c7
Open     equ  $c8
Read     equ  $ca
Write    equ  $cb
Close    equ  $cc
SetMark  equ  $ce
SetEof   equ  $d0
GetEof   equ  $d1
* Misc constants & buffer regions
scrfiles equ  $14      ;20 files/screen (&cat)
himem    equ  $b100
datbuf   equ  $b100    ;$200 bytes (tmp storage)
filebuf  equ  $bb00    ;$400 bytes (open file)
namelen  equ  datbuf+$23
filesper equ  datbuf+$24
filecount equ datbuf+$25
mli      equ  $bf00
lastdev  equ  $bf30
bitmap   equ  $bf58
preflg   equ  $bf9a
* Firmware equates
kbd      equ  $c000
clrkbd   equ  $c010
rom      equ  $c082
eighty   equ  $c300
* Applesoft equates
softev   equ  $3f2     ;"soft" reset vector
ampvect  equ  $3f5     ;ampersand vector
mbltu    equ  $d39a    ;block transfer
error    equ  $d412    ;onerr handler (code in X)
apconv   equ  $d4f2    ;redo apsoft hooks
scrtch   equ  $d64b    ;apsoft NEW
run      equ  $d566    ;apsoft RUN
clearc   equ  $d66c    ;apsoft CLEAR
stxtpt   equ  $d697    ;set txtptr to prog start
newstt   equ  $d7d2    ;apsoft GOTO
outspc   equ  $db57    ;print a space
outqst   equ  $db5a    ;print a question mark
outdo    equ  $db5c    ;print a character
crdo     equ  $dafb    ;print <CR>
frmnum   equ  $dd67    ;evaluate expression (#'s)
chkstr   equ  $dd6c    ;evaluate expression (str)
frmevl   equ  $dd7b    ;evaluate expression (any)
chkcom   equ  $debe    ;look for an devour comma
basic    equ  $e000    ;basic coldstart
basic2   equ  $e003    ;basic warmstart
illerr   equ  $e199    ;print ILLEGAL QUANTITY
garbag   equ  $e484
getbyt   equ  $e6f8    ;evaluate expression (#'s)
getadr   equ  $e752    ;convert fac to 2-byte int
inprt    equ  $ed19    ;print "BREAK IN xxxx"
linprt   equ  $ed24    ;prt 2-byte # (x & a-regs)
* Monitor equates
setreset equ  $fb6f
bell     equ  $fbdd
home     equ  $fc58
keyin    equ  $fd0c
prbyte   equ  $fdda
cout     equ  $fded
setkbd   equ  $fe89
setvid   equ  $fe93
monitor  equ  $ff59
* Startup code
* Prints title message, and
* relocates PDOS to $b300.
         org  $2000    ;this is a SYS file
         jmp  startup
         dfb  $ee      ;ProDOS startup protocol
         dfb  $ee
         dfb  $41      ;65 bytes of space
stulen   dfb  10
         asc  'SYSTEMBOOT' ;default exec file
         ds   54       ;65 - name - len byte
startup  lda  #<reloc
         sta  ptr
         lda  #>reloc  ;relocate from $20xx
         sta  ptr+1    ; to $b300
         ldy  #$00
         sty  ptr2
         lda  #$b3
         sta  ptr2+1
         ldx  #$08     ;relocate 8 pages
:reloc   lda  (ptr),y
         sta  (ptr2),y
         iny
         bne  :reloc
         inc  ptr+1
         inc  ptr2+1
         dex
         bne  :reloc
* Relocation done, now convince ProDOS to be friendly.
         ldx  #$17     ;clear sys bitmap
         lda  #$c1     ;b8-b9, bf
         sta  bitmap,x
         dex
         lda  #$1f     ;b3-b7
         sta  bitmap,x
         dex
         lda   #$00
:clear   sta   bitmap,x
         dex
         bpl   :clear
         lda   #$00     ;formerly $cf (0-2, 4-7)
         sta   bitmap
         lda   rom
         lda   clrkbd
         lda   #<basic2   ;set reset, rom, kbd
         sta   softev   ;soft reset vector ($3f2)
         lda   #>basic2
         sta   softev+1
         jsr   setreset
         jsr   begin   ;RDOS init code
* Put the title message on the screen
         ldy   #$00
:loop1   lda   titlemsg1,y
         beq   :title2
         sta   $4a8+0,y
         iny
         bne   :loop1
:title2  ldy   #$00
:loop2   lda   titlemsg2,y
         beq   :title3
         sta   $5a8+7,y
         iny
         bne   :loop2
:title3  ldy   #$00
:loop3   lda   titlemsg3,y
         beq   isprefix
         sta   $728+6,y
         iny
         bne   :loop3
isprefix lda   lastdev
         sta   curdev
         lda   preflg   ;is there a prefix?
         bne   namecp ;yes, continue
         jsr   newprefix   ;make sure prefix set
* copy name from startup spec (systemboot exec
                     file)
namecp   ldx   stulen
:loop    lda   stulen,x ;copy length byte too!
         sta   namebuf,x
         dex
         bpl   :loop
* init Applesoft and come back to life
exit     ldx   #$03
:loop    lda   cswl,x
         sta   varsave,x
         dex
         bpl   :loop
         lda   #<exit2 ;make control come back
                     here after
         ldy   #>exit2 ; Applesoft is initialized
         sta   kswl
         sty   kswl+1
         lda   #15      ;cursor vertical; don't
         sta   cv       ; step on title message
         jmp   c_new   ;init applesoft
exit2    ldx   #$03
:loop    lda   varsave,x
         sta   cswl,x
         dex
         bpl   :loop
         jmp   bootread   ;startup with exec file
titlemsg1 asc  "48K PDOS (ProDOS RDOS) By
                M.M. McFadden" ; msb is
                on
         dfb   $00
titlemsg2 asc  "Version 1.1 November 1991"
         dfb   $00
titlemsg3 asc  "(Simulates SSI's RDOS v2.1)"
         dfb   $00
* Start of program
reloc    equ   *      ;relocate from this point
                      ($20xx)
         org   $b300
begin    jmp   setstuff
dispatch ldy   #00
:loop    cmp   comtab,y
         beq   dis2     ;found command
         iny
         cpy   #19      ;end of commands?
         bne   :loop    ;not yet
notserv  jmp   $ff58    ;RTS; changed by &USR
dis2     tya
         asl
         tay
         lda   jmptab+1,y ;setup return vector
         pha
         lda   jmptab,y
         pha
         lda   textopen    ;is there a text file
                     open?
         beq   noneopen   ;nope
         jsr   restio   ;yup, close it
         jsr   closeall ;close text file
         lda   #$00
         sta   textopen
noneopen jmp   chrget   ;eat char
```

```
* Command handlers
* & CAT (catalog)
c_cat    cmp   #$c5    ;is next one "AT"?
         beq   :cat1   ;yes
         jmp   err_syn ;no, error
:cat1    jsr   mli
         dfb   GetPrefix
         dw    Prefix_prm
         lda   namebuf   ;is there a prefix?
         bne   :cat1_5
         jmp   err_io
:cat1_5
         ldy   #$00
:cat2    lda   cattext,y
         jsr   outdo
         iny
         cpy   #$24
         bcc   :cat2
         jsr   mli
         dfb   Open
         dw    Open_prm
         lda   Open_prm+5
         sta   Read_prm+1
         sta   GetEof_prm+1
         lda   #<datbuf
         sta   Read_prm+2  ;data buffer
         lda   #>datbuf
         sta   Read_prm+3
         lda   #$00
         sta   Read_prm+4  ;requested length
         lda   #$02
         sta   Read_prm+5
         jsr   mli
         dfb   Read
         dw    Read_prm
         bcc   :cat3
         jmp   proerr
:cat3    lda   #scrfiles
         sta   temp
         lda   filecount    ;# of active files
         sta   ctr2
         lda   namelen
         sta   holdlen
         lda   filesper ;# of files/dir
         sta   ctr
         sta   dirfiles
         lda   #<datbuf+$04
         sta   ptr
         lda   #>datbuf+$04
         sta   ptr+1
:cat4    jsr   :getvalid    ;move ptr to next entry
         jsr   :fiprint  ;print it
         dec   temp
         bne   :cat5
         jsr   keyin    ;wait for key
         lda   #scrfiles    ; every 18 files
         sta   temp
:cat5    dec   ctr2     ;all done?
         bne   :cat4    ;no, some remaining
         jsr   closeall ;close files
         jmp   chrget
:getvalid dec  ctr
         beq   :getnext
         lda   ptr
         clc
         adc   holdlen
         sta   ptr
         lda   ptr+1
         adc   #$00
         sta   ptr+1
:getvalid1 ldy #$00
         lda   (ptr),y
         beq   :getvalid     ;deleted, try again
         rts
:getnext jsr   mli
         dfb   Read
         dw    Read_prm  ;read another
         bcc   :getnext1
         jmp   proerr
:getnext1 lda  #<datbuf+$04
         sta   ptr
         lda   #>datbuf+$04
         sta   ptr+1
         lda   dirfiles
         sta   ctr
         bne   :getvalid1   ;(branch always)
:fiprint ldy   #$10     ;prints file entry
         lda   (ptr),y
         cmp   #$04     ;txt
         beq   :type1
         cmp   #$06     ;bin
         beq   :type2
         cmp   #$0f     ;dir
         beq   :type3
         cmp   #$fc     ;bas
         beq   :type4
         cmp   #$ff     ;sys
         beq   :type5
         lda   #"?" ;unknown type
         dfb   $2c
:type1   lda   #"T"
```

```
         dfb   $2c      ;bit opcode - skip rest
:type2   lda   #"B"
         dfb   $2c
:type3   lda   #"D"
         dfb   $2c
:type4   lda   #"A"
         dfb   $2c
:type5   lda   #"S"
         jsr   outdo    ;print character
         jsr   outspc
         ldy   #$13     ;get # of blocks
         lda   (ptr),y
         tax
         iny
         lda   (ptr),y
         tay
         bne   :form3   ;print with leading zeroes
         lda   #$b0
         cpx   #$0a
         bcs   :form1
         jsr   outdo
:form1   cpx   #$64
         bcs   :form2
         jsr   outdo
:form2   lda   #00
:form3   jsr   linprt
         jsr   outspc
:nameprt ldy   #$00
         lda   (ptr),y
         and   #$0f
         sta   ptr2
         inc   ptr2     ;adjust for len byte
         iny
:namloop lda   (ptr),y
         ora   #$80
         jsr   cout
         iny
         cpy   ptr2
         blt   :namloop
         lda   #30      ;print length...
         sta   ch       ;may not work with 80-cols
         ldy   #$15
         lda   (ptr),y
         tax
         iny
         lda   (ptr),y
         jsr   linprt
         jmp   crdo
* &LOAD "filename" {,load addr}
c_load   jsr   getbasprg
         jmp   basic2
* &RUN "filename" {,load addr}
c_run    jsr   getbasprg
         jmp   run
* &GOTO "filename" {,load addr} (chain)
c_goto   ldy   chainame
         sty   namebuf
:loop    lda   chainame+1,y
         sta   namebuf+1,y
         dey
         bpl   :loop
         lda   #<datbuf    ;read "CHAINSTUFF"
         sta   loadloc ; into $b100
         lda   #>datbuf
         sta   loadloc+1
         jsr   readfile
         jsr   chain
         ldy   #$07
:sloop   lda   vartab,y;save prog pointers
         sta   varsave,y
         dey
         bpl   :sloop
         jsr   getbasprg
         lda   vartab+1
         cmp   varsave+1  ;check loadloc
         bcc   :goto4
         lda   vartab
         cmp   varsave
         bcc   :goto4
         beq   :goto4
         jmp   err_syn ;program overwrote
                        variables (?)
:goto4   ldy   #$07
:goto5   lda   varsave,y
         sta   vartab,y
         dey
         bpl   :goto5
         jsr   stxtpt
         jmp   newstt
* &SAVE "filename"
c_save   jsr   getinstr
         lda   texttab
         ldy   texttab+1
         sta   loadloc
         sty   loadloc+1
         sec
         lda   prgend
         sbc   texttab
         sta   Write_prm+4    ;set save length
         lda   prgend+1
```

```
         sbc   texttab+1
         sta   Write_prm+5
         lda   #$fc     ;type = BAS
         jmp   writefile
* &STORE "filename",start,length (bsave)
c_store  jsr   getinstr
         bcs   :store1  ;need extra goodies
         jmp   err_syn
:store1  lda   linnum   ;start addr
         sta   loadloc
         lda   linnum+1
         sta   loadloc+1
         jsr   getextra;get length
         lda   linnum
         sta   Write_prm+4    ;set requested
                              length
         lda   linnum+1
         sta   Write_prm+5
         lda   #$06     ;type = BIN
         jmp   writefile
* &RECALL "filename" {,load addr} (bload)
c_recall jsr   getinstr
         php
         lda   #$06     ;type = BIN
         jsr   getldloc
         plp
         bcc   :recall1 ;use default loadloc
         lda   $50
         ldy   $51
         sta   loadloc
         sty   loadloc+1
:recall1 jmp   readfile
* &DEF "filename", size (create blank text file)
c_def    jsr   getinstr
         bcs   :def1    ;must have size (even
         jmp   err_syn ; though it's ignored)
:def1    lda   #$04
         sta   Create_prm+4
         lda   #$00
         sta   Create_prm+5
         lda   #$b1     ;load addr of $b100
         sta   Create_prm+6
         jsr   mli      ;update current time
         dfb   GetTime
         dw    $0000
         jsr   mli      ;create the file
         dfb   Create
         dw    Create_prm
         bcc   :def2
         jmp   proerr   ;if file exists, error
:def2    rts
* &PRINT
c_print  jsr   getinstr
         jsr   saveio
         lda   #$04
         jsr   getldloc ;check file type
         jsr   mli      ;(most errors caught by
         dfb   Open     ; getldloc)
         dw    Open_prm
         inc   textopen
         lda   Open_prm+5   ;refnum
         sta   Write_prm+1
         sta   GetEof_prm+1
         lda   #<datbuf     ;set buffer
         sta   Write_prm+2
         lda   #>datbuf
         sta   Write_prm+3
* Old behavior (v1.0): append to file
* jsr mli ;append file
* dfb GetEof
* dw GetEof_prm
* jsr mli
* dfb SetMark
* dw GetEof_prm ;append file
* New behavior for v1.1: truncate file
         lda   #$00
         sta   GetEof_prm+2
         sta   GetEof_prm+3
         jsr   mli
         dfb   SetEof
         dw    GetEof_prm
         lda   #<print_io
         sta   cswl
         lda   #>print_io
         sta   cswl+1
         lda   #<closerr    ;no input when writing
         sta   kswl
         lda   #>closerr
         sta   kswl+1
         rts
print_io pha            ;save regs
         and   #$7f     ;text files have hi-bit
         sta   datbuf   ; cleared
         tya
         pha
         txa
         pha
         ldy   #$01     ;write one byte
         sty   Write_prm+4
         dey
```

```
          sty   Write_prm+5
          jsr   mli
          dfb   Write
          dw    Write_prm
          pla
          tax
          pla
          tay
          pla
          rts
closerr   pha              ;close files, restore io
          jsr   closeall ; (called from c_read and
          jsr   restio   ; c_print)
          lda   #$00
          sta   textopen
          pla
          rts
* &READ "filename"
c_read    jsr   getinstr
bootread  jsr   saveio   ;(exec startup file)
          lda   #$04
          jsr   getldloc ;check file type
          jsr   mli      ;(most errors caught by
          dfb   Open     ; getldloc; rest by I/O)
          dw    Open_prm
          lda   Open_prm+5
          sta   Read_prm+1
          inc   textopen    ;set "text file open" flag
          lda   #<datbuf    ;set buffer
          sta   Read_prm+2
          lda   #>datbuf
          sta   Read_prm+3
          lda   #<read_io
          sta   kswl
          lda   #>read_io
          sta   kswl+1
          lda   #<checkeof
          sta   cswl
          lda   #>checkeof
          sta   cswl+1
          rts
read_io   lda   #$a0
          sta   (basl),y ;cover up char
          lda   #$01
          sta   Read_prm+4   ;length = 1
          lda   #$00
          sta   Read_prm+5
          jsr   mli
          dfb   Read
          dw    Read_prm
          bcc   :read_io1
          jsr   closerr
          jsr   prbyte
          jmp   err_syn
:read_io1 lda   datbuf
          bne   :read_io2   ;eof not hit
          jsr   closerr
          jsr   prbyte
          jsr   bell
          jmp   err_data    ;out of data
:read_io2 ora   #$80     ;set hi bit
          rts
checkeof  cmp   #$00
          bne   :checkeof1
          jsr   closerr
          jmp   restio
:checkeof1 rts
* &END (close)
* I used to send a $00 to cout, but that's no longer
* necessary... The call to &end is sufficient to
* close the open text file. Thus, this $00 will be
* sent to the screen, which is not what we wanted.
* so, just RTS.
*c_end lda #$00
* jmp cout ;send EOF character
c_end     rts
* &DEL "filename" (delete)
c_del     jsr   getinstr
          jsr   mli
          dfb   Destroy
          dw    Destroy_prm
          bcc   :del1
          jmp   proerr
:del1     rts
* &LEN (determine length of applesoft program)
* (omitted due to lack of space)
c_len     RTS
*c_len ldy #$00 ;"start"
* ldx #texttab
* jsr lenprint
* ldy #$07
* ldx #linnum
* sec
* lda prgend
* sbc texttab ;find program length
* sta linnum ;store in linnum
* lda prgend+1
* sbc texttab+1
* sta linnum+1
```

```
* jsr lenprint
* ldx #$69
* ldy #$0f
* jsr lenprint
* jmp crdo
*lenprint lda lenmsg,y
* beq lenprt1
* jsr outdo
* iny
* bne lenprint
*lenprt1 lda 01,x
* tay
* lda 00,x
* tax
* tya
* jsr linprt ;print 2-digit number
* jmp outspc
* &D #, next commmand (change drive)
c_d       ldx   #$03     ;must be <3
          jsr   getspnum
          dex            ;zero or one
          lda   curdev
          and   #$70     ;clear old drive #
          sta   curdev
          txa
          clc
          ror   a        ;shift to hit-bit
          ror   a
          ora   curdev
          sta   curdev
sd_done   jsr   newprefix  ;switch to new drive
          jsr   chkcom ;prepare for next command
          jmp   dispatch
* &S #, next command (change slot)
c_s       ldx   #$08     ;must be <1
          jsr   getspnum
          lda   curdev
          and   #$80     ;clear old slot #
          sta   curdev
          txa
          asl
          asl
          asl
          asl
          ora   curdev
          sta   curdev   ;*** assume drive 1
          jmp   sd_done
* &NEW (equivalent to DOS 3.3 "FP")
c_new     jsr   saveio
          lda   #<:new1
          ldy   #>:new1
          sta   kswl
          sty   kswl+1
          jmp   basic    ;coldstart
:new1     sta   (basl),y ;output the character
          lda   #<himem
          ldy   #>himem
          sta   memsize   ;reset himem to $b100
          sty   memsize+1
          jsr   restio
          jsr   scrtch
          jmp   basic2   ;warmstart
* &USR addr (chain user & vector)
c_usr     jsr   getextra1
          sty   notserv
          sta   notserv
          rts
* New PDOS commands
* &P"prefix" (change prefix - new command)
c_p       jsr   getinstr
          jsr   mli
          dfb   SetPrefix
          dw    Prefix_prm
          bcc   :p1
          jmp   proerr
:p1       rts
* &STOP (ProDOS QUIT code - new command)
c_stop    jsr   mli
          dfb   Quit
          dw    :quit_prm
          brk
:quit_prm dfb   $04
          dfb   0,0,0,0,0,0
* Subroutines
* Read BASIC program
getbasprg lda   texttab
          ldy   texttab+1
          sta   linnum
          sty   linnum+1
          jsr   getinstr ;get name, new load loc
          lda   #$fc     ;type = BAS
          jsr   getldloc
          lda   linnum   ;I don't know what this is
          sec
          sbc   loadloc
          sta   index
          lda   linnum+1
          sbc   loadloc+1
          sta   index+1
```

```
          lda   linnum
          ldy   linnum+1
          sta   loadloc
          sty   loadloc+1
          cpy   #$08     ;is load loc < $800?
          bcs   :getbas1 ;no, branch
          jmp   err_syn
:getbas1  lsr   errflg   ;shift 1 into hi-bit
          jsr   readfile
          clc
          lda   linnum
          sta   texttab  ;setup new start
          adc   GetEof_prm+2 ;setup
                         lomem:start+len
          sta   vartab
          lda   linnum+1
          sta   texttab+1
          adc   GetEof_prm+3
          sta   vartab+1
          lda   #$00     ;What does THIS do??
          ldy   #$ff
          dec   texttab+1
          sta   (texttab),y
          inc   texttab+1
          lda   cswl     ;save input vect
          ldy   cswl+1
          sta   iokeep
          sty   iokeep+1
          lda   #<:getbas2
          ldy   #>:getbas2
          sta   cswl
          sty   cswl+1
          pla
          sta   temp     ;return vector gets killed
          pla
          sta   temp+1
          jmp   apconv
:getbas2  lda   temp+1
          pha
          lda   temp
          pha
          lda   iokeep
          ldy   iokeep+1
          sta   cswl
          sty   cswl+1
          jmp   clearc
* Get file name string & extra goodies
* (extra number is placed into linnum)
getinstr  lda   #$55
          sta   $52      ;something about strings..
          jsr   frmevl
          jsr   chkstr   ;make sure it's a string
          ldy   #$02
:loop     lda   (facmo),y  ;pointer to str descriptor
          sta   strscr,y ;string scratch area
          dey
          bpl   :loop
          iny
:getin2   lda   (strscr+1),y ;read string
          and   #$7F     ;clear hi-bit
          cmp   #$60
          blt   :getin2_5
          sec
          sbc   #$20     ;convert lc->uc
:getin2_5 cmp   #$20     ;check if legal ProDOS
                         name
          bcs   :getin3
          jmp   illerr
:getin3   cmp   #$2F     ;/-9?
          bge   :val1    ;could be
          blt   :val3    ;no, too small
:val1     cmp   #$3a     ;is it 0-9?
          blt   :val4    ;yes!
          cmp   #$41     ;is it a-z
          bge   :val2    ;could be
          blt   :val3    ;no, too small
:val2     cmp   #$5b     ;is it <Z?
          blt   :val4    ;yes!
:val3     lda   #"." ;no.
:val4     sta   namebuf+1,y
          iny
          cpy   #$3f     ;max length
          beq   :getin4
          cpy   strscr   ;check length byte
          bcc   :getin2  ;more left
          lda   strscr
          sta   namebuf  ;store length of name
* New for v1.1: a length truncator
          tay
          tax
          lda   #$00
          sta   frespc   ;temp storage (Apsoft ptr)
:slloop   lda   namebuf,y
          cmp   #$2f     ;find last '/'
          beq   :foundsl
          dey
          bne   :slloop
          txa            ;no slash; get full len
          bne   :chklen  ;BRA
:foundsl  sty   frespc
          txa            ;get full len
```

```
          sec
          sbc   frespc   ;subtract prefix part
:chklen   cmp   #$10     ;15 chars max
          bcc   :lenok
          lda   #$0f
          clc
          adc   frespc
          sta   namebuf  ;set len to prefix+15
                         chars
:lenok                   ;length was ok, so no
                         change
:getin4   jsr   chrgot   ;look at next char
          cmp   #$2c     ;is there a comma?
          beq   getextra ;yes!
          bne   getin6   ;no. (branch always)
getextra  jsr   chkcom
getextra1 jsr   frmnum   ;evaluate number
          jsr   getadr   ;clean it up
          sec            ;signal additional info
          dfb   $24      ;z-page bit
getin6    clc            ;signal no numbers
          rts
* Get load loc & verify file type
getldloc  sta   temp     ;save wanted type
          jsr   mli
          dfb   GetInfo
          dw    GetInfo_prm
          bcc   :getldloc1
          jmp   proerr
:getldloc1 lda  GetInfo_prm+4
          cmp   temp
          beq   :getldloc2
          jmp   err_type
:getldloc2 lda  GetInfo_prm+5
          sta   loadloc
          lda   GetInfo_prm+6
          sta   loadloc+1
          rts
* Get number between 1 and x
getspnum  stx   temp
          jsr   getbyt   ;get num, put in x
          cpx   #$01
          bcc   :badspnum ;if < 1, error
          cpx   temp
          bcs   :badspnum ;if > x, error
          rts
:badspnum       jmp   illerr
* Get new prefix based on curdev
newprefix lda   curdev
          sta   Online_prm+1
          jsr   mli
          dfb   Online
          dw    Online_prm
          bcc   :newpre1
          cmp   #$28     ;allow "no dev con"
          beq   :newpre2 ; in case slot w/o drv 2
          cmp   #$27     ;allow i/o error
          beq   :newpre2 ; in case of empty drive
          jmp   proerr
:newpre1  lda   namebuf+1 ;OnLine shifts name
          and   #$0f     ;screen out dev #
          sta   namebuf
          inc   namebuf  ;+1 for "/"
          lda   #$2f     ;"/"
          sta   namebuf+1 ;fully qualify name
          jsr   mli
          dfb   SetPrefix
          dw    Prefix_prm
          bcc   :newpre2
          jmp   proerr
:newpre2  rts
* Read entire file at LoadLoc
readfile  jsr   mli
          dfb   Open
          dw    Open_prm
          bcs   readdone
          lda   Open_prm+5   ;transfer ref #
          sta   Read_prm+1
          sta   GetEof_prm+1
          jsr   mli
          dfb   GetEof
          dw    GetEof_prm
          lda   GetEof_prm+2 ;transfer eof
          sta   Read_prm+4
          lda   GetEof_prm+3
          sta   Read_prm+5
          lda   loadloc
          sta   Read_prm+2
          lda   loadloc+1
          sta   Read_prm+3
          jsr   mli
          dfb   Read
          dw    Read_prm
readdone  bcc   closeall ;if read okay, branch
          jmp   proerr
closeall  jsr   mli
          dfb   Close
          dw    Close_prm
          rts
* Write file at loadloc, length in Write_prm
```

```
writefile  sta  Create_prm+4 ;file type
           lda  loadloc
           sta  Create_prm+5
           sta  Write_prm+2
           lda  loadloc+1
           sta  Create_prm+6 ;aux type = loadloc
           sta  Write_prm+3
           jsr  mli
           dfb  Create
           dw   Create_prm
           bcc  :writfile1
           jmp  proerr
:writfile1 jsr  mli
           dfb  Open
           dw   Open_prm
           bcc  :writfile2
           jmp  proerr
:writfile2 lda  Open_prm+5   ;ref #
           sta  Write_prm+1
           jsr  mli
           dfb  Write
           dw   Write_prm
           jmp  readdone
* ProDOS error handler
proerr     pha
           jsr  closeall
           pla
           cmp  #$2b
           beq  err_wp
           cmp  #$40
           beq  err_syn
           cmp  #$44
           beq  err_fnf
           cmp  #$45
           beq  err_fnf
           cmp  #$46
           beq  err_fnf
           cmp  #$47
           beq  err_dup
           cmp  #$48
           beq  err_full
           cmp  #$49
           beq  err_full
           cmp  #$4c
           beq  err_data
           cmp  #$4d
           beq  err_data
           cmp  #$55
           beq  err_full
           cmp  #$57
           beq  err_dup
           bne  err_io
* RDOS error handler
* (Note: numbers must match ("e = peek(222)")
err_fnf    ldx  #01      ;file not found
           dfb  $2c
err_syn    ldx  #$02     ;syntax
           dfb  $2c
err_dup    ldx  #$03     ;duplicate
           dfb  $2c
err_full   ldx  #$04     ;disk full
           dfb  $2c
err_data   ldx  #$05     ;out of data
           dfb  $2c
err_io     ldx  #$06     ;i/o error
           dfb  $2c
err_type   ldx  #$07     ;file type mismatch
           dfb  $2c
err_wp     ldx  #$08     ;write protected
           bit  errflg
           bpl  :err1    ;onerr not active; print
           jmp  error
:err1      jsr  crdo
           ldy  #$ff
:loop      iny
           lda  errmsg,y
           bpl  :loop
           dex
           bne  :loop
           jsr  outqst
:err3      iny
           lda  errmsg,y
           php
           jsr  outdo
           plp
           bmi  :err4
           jmp  :err3
:err4      lda  curlin
           cmp  #$ff
           beq  :err5
           jsr  inprt
:err5      jsr  bell
           jmp  basic2
* Save/restore I/O vectors
saveio     ldx  #$03
:saveio1   lda  cswl,x
           sta  iokeep,x
           dex
           bpl  :saveio1
           rts
restio     ldx  #$03
```

```
:restio1   lda  iokeep,x
           sta  cswl,x
           dex
           bpl  :restio1
           rts
* Startup code
* This DOES get called on occasion, so we can't
                         just
* shove it into the load-time startup stuff.
setstuff   jsr  setkbd
           jsr  setvid
           jsr  home
           lda  #$4c
           sta  ampvect
           lda  #<dispatch
           sta  ampvect+1
           lda  #>dispatch
           sta  ampvect+2
           lda  #<himem
           sta  memsize   ;set himem
           lda  #>himem
           sta  memsize+1
           lda  #$00      ;disable trace
           sta  traceflg
           rts
* Data
* Applesoft tokens for commands
comtab     dfb  $43   ;C at
           dfb  $b6   ;load
           dfb  $ac   ;run
           dfb  $ab   ;goto
           dfb  $b7   ;save
           dfb  $a8   ;store
           dfb  $a7   ;recall
           dfb  $b8   ;def
           dfb  $ba   ;print
           dfb  $87   ;read
           dfb  $80   ;end
           dfb  $85   ;del
           dfb  $e3   ;end
           dfb  $44   ;D
           dfb  $53   ;S
           dfb  $bf   ;new
           dfb  $d5   ;usr
           dfb  $50   ;P (prefix)
           dfb  $b3   ;stop
jmptab     dw   c_cat-1 ;locations of commands
           dw   c_load-1
           dw   c_run-1
           dw   c_goto-1
           dw   c_save-1
           dw   c_store-1
           dw   c_recall-1
           dw   c_def-1
           dw   c_print-1
           dw   c_read-1
           dw   c_end-1
           dw   c_del-1
           dw   c_len-1
           dw   c_d-1
           dw   c_s-1
           dw   c_new-1
           dw   c_usr-1
           dw   c_p-1
           dw   c_stop-1
curdev     dfb  $60   ;current device #
holdlen    dfb  $00   ;temp storage (cat)
dirfiles   dfb  $00   ;temp storage (cat)
loadloc    ds   2
iokeep     ds   4     ;holds $36-39
temp       ds   2     ;temporary storage
varsave    ds   8     ;used by &goto
textopen   dfb  $00   ;TRUE if a text file is open
* Parameter lists
Create_prm    dfb  $07
              dw   namebuf
              dfb  $c3     ;unlocked
              dfb  $00     ;file type
              dfb  $00,$00;aux type (load address)
              dfb  $01     ;seedling file
              dfb  $00,$00;create date
              dfb  $00,$00;create time
Destroy_prm   dfb  $01
              dw   namebuf
Online_prm    dfb  $02
              dfb  $60     ;slot 6, drive 1
              dw   namebuf+1 ;room for leading "/"
GetInfo_prm   dfb  $0a
              dw   namebuf
              dfb  $00
              dfb  $00     ;file type
              ds   2       ;aux type
              dfb  $00
              ds   2       ;blocks used
              ds   8       ;dates/times
Prefix_prmdfb $01
              dw   namebuf
Open_prmdfb   $03
              dw   namebuf
              dw   filebuf
```

```
              dfb  $00   ;ref #
Read_prm dfb  $04
              dfb  $00   ;ref #
              dw   datbuf
              ds   2     ;len
              ds   2     ;actual len
Write_prm dfb $04
              dfb  $00   ;ref #
              dw   datbuf
              ds   2     ;length
              ds   2     ;actual len
Close_prmdfb  $01
              dfb  $00   ;ref # - 0 closes all
GetEof_prm    dfb  $02
              dfb  $00   ;ref #
              ds   2     ;eof
              dfb  $00   ;hi-byte always 0
cattext    asc  " LEN      -<NAME>-
                         LENGTH"
           dfb  $8d
lenmsg     asc  "START:"
           dfb  $00
           asc  "LENGTH:"
           dfb  $00
           asc  "LOMEM:"
           dfb  $00
* RDOS error messages
errmsg     dfb  $80   ;(dci=msb clear exc last)
           dci  'FILE NOT FOUND'
           dci  'DOS SYNTAX ERR'
           dci  'DUPLIC ENTRY'
           dci  'DISK FULL'
           dci  'OUT OF DATA'
           dci  'I/O ERR'
           dci  'FILE TYPE ERR'
           dci  'WRITE PROTECTED'
* File containing chain information
chainame dfb $0a     ;length
           asc  'CHAINSTUFF'
* Pathname buffer
namebuf    ds   65    ;len + 64 chars
* Save the assembly output file
           lst  on
           nop        ;last byte before $BB00?
           typ  $ff   ;make it a SYS file
           sav  PDOS.SYSTEM   ;save the first
                         chunk here
* Chainstuff source
* This is loaded from the
* current directory when the
* &goto command is executed.
* (it's simply an adapted
* version of DOS 3.3 chain.)
*
           lst  off
           org  $b100 ;data buffer
chain      jsr  garbag
           lda  #$07
           sta  scrub
           lda  vartab
           ldx  vartab+1
           sta  dsctmp
           stx  dsctmp+1
chain1     cpx  arytab+1
           bne  chain2
           cmp  arytab
           beq  chain2_5
chain2     jsr  chain9
           beq  chain1
chain2_5   sta  facmoh
           stx  facmo
           lda  #$03
           sta  scrub
chain3     lda  facmoh
           ldx  facmo
chain4     cpx  strend+1
           bne  chain5
           cmp  strend
           bne  chain5
           rts
chain5     sta  dsctmp
           stx  dsctmp+1
           ldy  #$00
           lda  (dsctmp),y
           tax
           iny
           lda  (dsctmp),y
           php
           iny
           lda  (dsctmp),y
           adc  facmoh
           sta  facmoh
           iny
           lda  (dsctmp),y
           adc  facmo
           sta  facmo
           plp
           bpl  chain3
           txa
```

```
           bmi  chain3
           iny
           lda  (dsctmp),y
           ldy  #$00
           asl
           adc  #$05
           adc  dsctmp
           sta  dsctmp
           bcc  chain6
           inc  dsctmp+1
chain6     ldx  dsctmp+1
chain7     cpx  facmo
           bne  chain8
           cmp  facmoh
           beq  chain4
chain8     jsr  chain9_5
           beq  chain7
chain9     lda  (dsctmp),y
           bmi  chain11
           iny
           lda  (dsctmp),y
           bpl  chain11
           iny
chain9_5   lda  (dsctmp),y
           beq  chain11
           iny
           lda  (dsctmp),y
           tax
           iny
           lda  (dsctmp),y
           sta  lowtr+1
           stx  lowtr
           cmp  prgend+1
           beq  chain10
           bcs  chain11
chain10    dey
           dey
           lda  (dsctmp),y
           pha
           sec
           lda  fretop
           sta  highds
           sbc  (dsctmp),y
           iny
           sta  (dsctmp),y
           sta  fretop
           iny
           lda  fretop+1
           sta  highds+1
           sbc  #$00
           sta  (dsctmp),y
           sta  fretop+1
           pla
           clc
           adc  lowtr
           sta  hightr
           lda  lowtr+1
           adc  #$00
           sta  hightr+1
           jsr  mbltu
chain11    lda  scrub
           clc
           adc  dsctmp
           sta  dsctmp
           bcc  chain12
           inc  dsctmp+1
chain12    lda  dsctmp
           ldx  dsctmp+1
           ldy  #$00
           rts
* Save "ChainStuff"
           lst  on
           typ  $06   ;change to "bin" file type
           sav  CHAINSTUFF ;save this last part
                         in a separate file
           lst  off
```

Stephen Rich SC

A "LISTable" version of Warship & WWI Battlecruiser

Softkey for...

Warship
WWI Battlecruiser
SSI

1. Make back-up copies of Warship and both sides of Battlecruiser using the technique reported by Jack R. Nissel in COMPUTIST #51.

POKE 47426,24 *Ignore epilog & check-sum errors*

POKE 47447,0 *Ignore address prolog byte #1*
RUN COPYA

2. Boot COPY II+ Bit Copy program. Using either Bit Copy or Sector Copy, copy Track 00 from the WW II side of the BATTLECRUISER back-up disk to Track 00 of the back-ups of WARSHIP and WW I side of BATTLECRUISER.

Listable back-up disks result from step 2. Programs from any of the 3 sides of the back-up disks may now be stopped and listed by using ctrl-C for BASIC programs and ctrl-reset for binary programs.

Modify or duplicate Warship & Battlecruiser games saved on RDOS data disks

It is necessary to use a listable back-up disk to perform this procedure with WARSHIP or WW I BATTLECRUISER as it will only work with the RDOS from WW II BATTLECRUISER.

1. Set up the WARSHIP/BC GSTART program to build a NEW GAME. Select 'TWO-PLAYER' under category (3) to modify ships of either side.
2. When the map selection appears, select '1. OPEN SEA', even if the game you will be modifying has a map.
3. Press ctrl-reset to halt the BUILD program when '(B)UILD SCENARIO.' appears on the screen.
4. Place Gamesave Disk in disk drive and Type in: & RECALL "SAVED. GAME.NAME" (quotes must be between the & RECALL command and the name of the game).
5. Place WARSHIP/BC Disk in disk drive and Type in: & RUN "DEPLOY",16384 (or GOTO 40 if you haven't erased the VECTOR program from memory with a 'NEW' command).
6. Modify the game as you would a game just constructed. If you wish to modify ships of both sides and did not select 'TWO-PLAYER' as described in step 1, then stop the program by pressing 'ctrl-C', type 'POKE 37648, 2' and 'RUN'.
7. Save the game with the same name if the modification was a correction or a different name if you want to create a new variation in addition to previous game.

Note: This method is used without using Step 6 as the only available method of duplicating a single game already saved on a disk.

Experimental modification

3b. Put data in memory with the BUILD program such as the value of DAMAGE CONTROL before stopping the program with ctrl-reset. These values may or may not be overwritten by the recall of the game from the disk but some may be retained. The only way to know is by trying!

Viewing computer controlled enemy ships during play (Ultra?)

1. Press ctrl-C to stop the ORDERS program after pressing <space> to continue.
2. Type: GOTO 505 to view computer controlled Axis ships.
3. Type: GOTO 605 to view computer controlled Allied ships.

During use of the ORDERS program, PEEK (NP) holds the value that determines jumps in the program for orders. (NP=address 37648 in the WW I ORDERS pgm). The values are indicative as follows:

0 = Allied
1 = Axis
2 = Both computer or two-player

It is unnecessary to alter these values to view enemy ships during orders if step 2 or 3 in the procedure above is used.

Warship build program with modified ship selection points

Here's how to get a WARSHIP listable back-up disk's BUILD program's auto-selection of Japanese ships to work when the ship selection points data has been modified and the program halts during auto selection:

ctrl-reset
CALL -151
40A0G *and then select Allied ships.*

-OR-

409AG *first add some additional Japanese ships.*

If the BUILD program's auto-selection of Allied ships halts during auto selection then:

ctrl-reset
CALL -151
4151G *and then attempt adding more Allied ships.*

Other useful locations in the WARSHIP BUILD program are:

3E54G	select type of action
3FFCG	Japanese select ships
40AAG	Allied auto-select ships Y/N

Increasing ship selection points for BUILDS in Warship/BC

To create a Jutland type battle using all capital ships with expensive SSP costs, it is necessary to modify some of the data locations of the BUILD program to increase the total of ship selection points available.

1. Using a listable back-up disk, set up the WARSHIP/BC GSTART program to build a NEW GAME.
2. Select the type of map you want when the map selection appears.
3. Press ctrl-reset to halt the BUILD program when '(B)UILD SCENARIO...' appears on the screen.
4. Enter the Monitor (CALL -151) and modify the data at the following locations as desired:

WW I BATTLECRUISER

Battleline	$3400:8C 48;was $8B 16 now 3200 ssp
Transport	$3405:8A 70;was $89 34 now 960 ssp
Intercept Tran.	$340A:8A 57;was $89 20 now 860 ssp
Bombardment	$340F:8B 20;was $89 70 now 1280 ssp
Intercept Bomb.	$3414:8B 07;was $89 48 now 1080 ssp. Also found on Track $0D Sector $0C

WW II BATTLECRUISER

Battleline	$3400:8D 02;was $8A 16 now 4160 ssp
Transport	$3405:8C 02;was $89 16 now 2080 ssp
Bombardment	$340A:8C 2F;was $89 48 now 2800 ssp. Also found on Track $0D Sector $0B

WARSHIP

Battleline	$33D7:8D 20;was $8B 16 now 5120 ssp
Transport Carry	$33DC:8B 3E ;was $89 34 now 1520 ssp
Intercept Carry	$33E1:8B 2A;was $89 20 now 1360 ssp
Bombardment	$33E6:8B 7F;was $89 70 now 2040 ssp
Intercept Bomb.	$33EB:8B 57;was $89 48 now 1720 ssp. Also found on Track $0D Sector $0C

5. Upon completion of data modification, press ctrl-C to leave the Monitor and type: GOTO 30 You will now be able to build any 'type' of ships needed for your simulations. Unfortunately, without some type of memory expansion and a major restructuring of the programs, the number of ships must remain at 20 ships per side.

Making all ship nationalities available for selection

It is possible to use switch the side of ships, such as placing French ships on the Axis side in a Mers-el-Kebir Vichy French vs British type action. It is necessary to modify two data bytes of the BUILD program to allow a complete listing of ships so they will then be available for selection.

1. Using a listable back-up disk, set up the BC WW II GSTART program to build a NEW GAME.
2. Select the type of map you want when the map selection appears.
3. Press ctrl-reset to halt the BUILD program when '(B)UILD SCENARIO ...'appears on the screen.
4. Enter the Monitor and modify the data at the following locations:

Battlecruiser WW II
CALL -151
33E2:87 *was $85*
33E3:1A *was $48*

Equivalent locations are:
BC WW I: $33B2 and $33B3
Warship: $33B9 and $33BA

5. Upon completion of data modification, press ctrl-C to leave the Monitor and type: GOTO 30 You will have a complete listing of all ships regardless of nationality to choose from during the BUILD program.

Ship Data for the Various Fleets

SSI's Warship and Battlecruiser simulations are superb in attaining detailed tactical action results for surface combat units in a relatively quick manner - especially when compared to other non-computerized, or semi-computerized methods. Some limitations imposed by 64K memory size should be kept in mind when judging the results.

1) The simulation is purely tactical and non-strategic, allowing no delayed entry of reinforcements.

2) No provision is made for starshells, mine warfare, submarine, or air attacks.

3) The programs are not set up to accommodate more than 20 units per side and fleet reversal maneuvers can only be done manually on a per ship basis.

4) Ship silhouettes are non-specific and do not even display an orange tint for burning ships.

Any revision for 128K of either product would help to alleviate some of these limitations and contribute to an even greater degree of realism in the results obtained.

The only problems I've heard encountered by other program users usually concern the creation of a historical or factually based simulation. In this regard, the limitation certainly doesn't apply to the program but rather to the user's data base - or lack of one! Quandaries created by being unable to christen but one of several Fubuki class destroyers or by trying to remember just how many North Carolina Class BBs were built can be remedied by having the information available in a data base.

The charts provided here should help to identify some of those unknown ships and provide a factual basis for creating computer simulations. Any additional data or correction of any erroneous data is welcomed.

Stephen Rich sent several Appleworks databases that are too extensive to print. If you need the info, send your request to the author (thru Computist) along with $2 to cover costs. If Stephen is unwilling or unable to fill requests perhaps the data could be uploaded to the Computist BBS. ....RDEXed

---

Seymour Joseph NJ

Softkey for...

Calendar Crafter v1.3

MECC

Requirements:
Apple IIgs 768K
Disk editor
3.5" disk copier that ignores errors

Upon finding the program copy protected, I tried all of the older softkeys for Calendar Crafter published in Computist. None worked quite right, But using information from the one on Page 8 of Issue 62, I was able to discover the correct softkey for this, newer, version.

I used the previous author's hint to search for the hex bytes: C9 27 00 D0 02 18 60 38. This pattern of bytes appeared only once on my disk in block $4A8. Once I located them, I changed the final 38 (SEC) to an 18 (CLC) to defeat the protection.

Step-by-step

1. Lock the original disk and copy it with any 3.5" disk copier that will ignore bad blocks.
2. Sector edit the copy with your favorite utility (I use Prosel 16 ZAP).

Blk	Byte	From	To
$4A8	$75	38	18

3. Write the sector back to disk.

Voila! Make as many backups as you want, or even install the program on your hard disk under GS/OS. I think it's a great Apple IIgs utility.

Scott A Jelsma IA

Copy II Plus discontinued?

I am a Central Point Software Beta Tester for the COPY II PLUS and have noticed that several Computist readers have found a few bugs in version 9.1 and are wondering when they will be fixed.

Some readers are also wishing for a few new features in the COPY II PLUS.

Central Point Software discontinued the Copy II MAC about a year ago and I figured that the Copy II Plus was next. I heard from Central Point Software, September 4 & 9, and they are STRONGLY CONSIDERING DROPPING THE COPY II PLUS from there software line. This means NO bug fixes or future utility improvements. Version 9.1 will probably be the last version of the Copy II Plus released.

The reason they give for discontinuing these wonderful products is that on-disk copy-protection is a thing of the past since more people are using hard drives. So the software manufactures are now making there software hard drive compatible.

I was also told that Central Point Software will NOT make any future Apple II products. They will be abandoning the Apple II line if the Copy II Plus is dropped from their current line. But, will offer technical support, up load new parameters sent in by users on their BBS, and may send out a list of new parameters to registered users upon request. One reason for dropping the Copy II Plus is advanced users are not sending in any parameters to copy their copy-protected software.

The date on the final decision for dropping the Copy II Plus has not been currently set.

To all Computist readers who think the Copy II Plus is one of the best Apple II utility program available, my suggestion is to write Central Point Software a letter and tell them how much you like the Copy II Plus and would like them to have another bug fix and make some improvements in there utilities. If you write them, be sure to address your letter to Marie Smith. Or if you call, be sure and ask for her. She is in charge of the technical support of the Copy II Plus.

Bugs in the Copy II Plus

1. When copying DOS 3.3 files with an Apple IIgs with 1.25 Meg of memory and one 5.25" disk drive. When the first DOS 3.3 file copies it will take an extremely long time to copy and then a message saying DISK FULL will appear on the screen. This happens when the disk is completely blank and the disk has been formatted in DOS 3.3. I have tried this option with an Apple IIe (two 5.25" drives), IIc (one 5.25" disk drive), and a IIgs with 512K of memory and two Apple 5.25" disk drives and it will work correctly. This option will also work correctly if I remove my Apple memory expansion card in my IIgs and just use the 256K on the motherboard with one 5.25" drive

2. The 3.5" sector edit printer dump will not stop printing after the one sector specified. It will continue on until you turn your computer off or turn your printer off.

3. When you are using the 5.25" bit copier on a IIgs which has a RAM disk set up, it doesn't matter how big the RAM disk is, the 5.25" bit copier will read the first set of instructions in the parameter (example: T0-T11, Sector Copy) and then write those tracks to the target disk. At track 11 it will continually try to write it to the target disk. It will not allow the Copy II Plus to go on to the next line of instructions in the parameter (for example T12-T22). If you do not have a RAM disk set up, the 5.25" bit copier will work correctly. This bug will show up on any parameter, not just the parameter I was using as an example, that has more than one line of instructions in the parameter with a RAM disk set up.

The sample parameter I was referring to was:

```
T0-T11, Sector Copy
T12-T22
```

4. When exiting the Copy II Plus and have installed ProDOS version 1.9 on your 3.5" or 5.25" disk the Copy II Plus will not exit to the new ProDOS quit code. You will still see the old ProDOS quit code /COPY II PLUS/.

If you know of anymore bugs please, let me know! If you know how to fix any of these bugs, please let other Computist readers know because it looks like we are on are own now!

Some improvements I would like to see

• the ability to read more than one file into memory before writing the files to the target disk so there is less disk swapping if the user has only one disk drive

• the ability to read and write GS/OS extended files or fork files

• the remaining bugs in version 9.1 fixed

A bug in AppleWorks 3.0

I have also found a new bug in AppleWorks 3.0. To create this bug follow the procedure listed below:

1. Create a new database from scratch.
2. Put in the category names - Attention, Company Name, City/State/Zip
3. Now type in the three records listed below:

```
Attention: (leave blank)
Company: Beagle Brothers
Street Address: 6215 Ferris
            Square
City/State/Zip: San Diego CA
            92121
Attention: John Scully
Company: Apple Computer Inc.
Street Address: 20525 Mariani
            Avenue
City/State/Zip: Cupertino CA
            95014
Attention: (leave blank)
Company: Broderbund Software Inc.
Street Address: 17 Paul Drive
City/State/Zip: San Rafael CA
            94903-2101
```

4. Create a new labels format from scratch.
5. Give it any report name - I used Scott.
6. It shows that the address label will use four lines.
7. Do an open-apple O and change PW to 3.0", PL to 1" and type in PH to turn off the printing of the Report Header at the top of each page. Leave all other setting at their default settings.
8. Print or address labels to the screen!

Notice there isn't any space between the 2nd and 3rd records. When printing the labels on the Imagewriter II printer, the labels looked fine. The bug only shows up when you preview the labels on the screen.

I printer, the labels looked fine. The bug only shows up when you preview the labels on the screen.

Bugs in "Where In The USA Is Carmen Sandiego GS"

I have also found two bugs in "Where In The USA Is Carmen Sandiego GS Version" and have wrote several letters detailing the two bugs I have found and Broderbund's technical support says that they are unable to recreate these particular bugs. Has any Computist reader been able to recreate the following bugs? If so, I would appreciate knowing that I am not the only one experiencing these problems.

The 1st bug

I have found this bug occurs after a case ends. This means the bug shows up every time you win a case and sometimes when you lose a case.

1. First, win a case or lose a case. I would suggest win a case because it is more likely to show up.
2. When it says "Ready for your next case, NAME?" (Y/N), pull down the game menu and select Acme Detective roster. DISPLAY THE ROSTER. Then select exit to exit out of the detective roster.

You will see the lower half of the printer's paper is still there BUT the upper half has now turned into the upper half of the picture of the state you caught the crook in or ran out of time in. There is the first bug.

The 2nd bug

This bug showed up after I received the rank of super sleuth. If I do not finish a game at this time and go back later to finish it, it reports that my rank is master detective when I have already been promoted to super sleuth. If I do not save a game and just start a new game from scratch, it reports my rank correctly as super sleuth.

My hardware is:

Apple IIgs (1.25Mbyte - ROM 01)
3.5" Apple 800K drive
5.25" Apple Drive
Imagewriter II Printer
Apple Mouse
Apple RGB Monitor

I would like to thank John C. De La Cruz for his softkeys for the "Teacher's Tool Kit v3.1" and Big Al for his softkey for "Where In The USA is Carmen Sandiego GS v1.0"!

Softkey for...

Kinder Koncepts

Midwest Software

Requirements:
Apple II with at least 128K
5.25" disk copier that can ignore any errors (Copy II Plus's 'Copy disk)
5.25" disk editor
2 - 5.25" blank disks

Copy the disk(s) and perform ONE of the following edits:

Trk	Sct	Byte	From	To
0B	0D	5F	20 4D 49	18 EA EA
07	0A	0D	20 00 BD	18 EA EA

Perform the edit on both the reading and math series disks. The first edit will bypass the check by not letting the file MIDWEST get Bloaded into memory. The file MIDWEST contains only Midwest's copy-protection so no other important program information will be lost.

The second edit will edit the Midwest file and make the program think that any disk is a master disk.

Bitkey for...

Midwest Software

Midwest Software

Requirements:
Copy II Plus
2 blank 5.25" disks

```
TRY MIDWEST SOFTWARE (Midwest
Software)
T0-T2, SECTOR COPY
T3
T4-T22, SECTOR COPY
""
"THIS PARAMETER WORKS WITH BOTH"
"THE READING AND MATH SERIES"
"DISKS"
```

Don Westcott CO

A few months ago I bought a TULIN Half Shell 120 meg hard drive and a CVT RAMFAST SCSI card for my GS. I was very impressed with its speed. I began loading it and, with the help of some hardkeys from COMPUTIST, I got about a third or more of my GS software to work on it.

After loading Software Toolworks' LIFE AND DEATH game onto it I tried launching it and the monitor went haywire. When I tried to reboot I found out the boot partition had crashed so I had to reformat it. I wondered why the boot partition had crashed since I had put LIFE AND DEATH on a different partition. After reformatting I loaded LIFE AND DEATH and tried it again and it crashed again. The only connection between LIFE AND DEATH and the boot partition is the icon file I put into SYSTEM/ICON folder in the boot partition. So I reformatted it a third time and loaded LIFE AND DEATH without putting the icon file into the SYSTEM/ICON folder. This time it didn't crash.

I was recently trying to load APPLEWORKS GS onto the fourth partition of the hard drive. It was getting errors during the transfer of the main system file. I then discovered that the fourth partition had crashed. Now I'll have to reformat again.

? Has anyone else had similar experiences with TULIN or other hard drives? Is there any software or hardware that can prevent, stop or recover from a hard drive crash so to avoid reformatting?

I recently bought ReadySoft's SPACE ACE for the GS. The only controls for this game are numbers on the GS keyboard's numeric keypad. My GS is an upgraded IIe so I don't have the numeric keypad. Why didn't they include joystick control? SPACE ACE comes on 9 disks but it ISN'T hard drive installable!

Krakowicz NY

The Basics of Kracking Part 11 & 12

Softkey for...

Cyclod

Sirius Software

Sirius Software, in their latest releases (Minotaur, Bandits, Fly Wars, Cyclod, etc) has raised the science of copy protection to new heights. As you know, most disks that do a lot of disk accessing are not easily cracked, and most people work very hard developing parms for the popular backup programs. Because of the techniques used by Sirius, it is doubtful that any of the presently available copiers will be successful, and new effort must be focused on the cracking of these programs.

Don't be alarmed if the terms used here are unfamiliar to you. We will be doing many of these in the future, and you'll have a chance to get used to the techniques and jargon as we go along. You might also like to read previous "Basics", parts 1-10.

This discussion assumes a basic knowledge of cracking techniques - memory moves, probable starting locations, Exclusive-ORing to hide sensitive code, etc, and a good working knowledge of a sector editor. My favorite is the Inspector, but the one in Nibbles Away II also has some nice features. Having the Inspector in ROM is just about a necessity for today's Software Artist, anyway. All addresses are given in hexadecimal, with binary or decimal equivalents as required.

The listings below were extracted from Cyclod, but are virtually the same for all of the new Sirius programs. If you can get your hands on an original, you will be able to experiment with some of the tips given here and learn considerably more.

The first protection device being used, and one of the oldest, is loading a crucial part of the program across the text screen memory from $400-$7FF, so it will scroll the top line off the screen when you hit reset. The part loaded there on these programs is one we will call "loader", since it acts as the substitute "DOS" for all disk accesses. (If you have an original, now is the time to copy track 0 onto a blank diskette using your favorite copier - almost any will get it. All future references to the disk are for the single track you just copied - don't take a chance with the original). To get a look at this loader, however, we have to go back to the fundamentals of the Apple Disk System. Remember track 0, sector 0 of every disk must always, always be readable by the boot ROM, and more or less by most sector editors. Read T0,S0 into location $800 up, and from the monitor type in "801L" (recall that location $800 is used to tell the boot ROM how many pages to load in) to list this "Preloader". The listing below is a disassembly of all the meaningful code.

```
0801:AD 52 C0   LDA $C052
0804:AD 57 C0   LDA $C057
0807:AD 55 C0   LDA $C055
080A:AD 50 C0   LDA $C050
080D:AD 81 C0   LDA $C081
0810:AD 81 C0   LDA $C081
0813:A0 00      LDY #$00
0815:84 00      STY $00
0817:A9 D0      LDA #$D0
0819:85 01      STA $01
081B:A2 30      LDX #$30
081D:B1 00      LDA ($00),Y
081F:91 00      STA ($00),Y
0821:C8         INY
0822:D0 F9      BNE $081D
0824:E6 01      INC $01
0826:CA         DEX
0827:D0 F4      BNE $081D
0829:A6 2B      LDX $2B
082B:BD 89 C0   LDA $C089,X
082E:A9 04      LDA #$04
0830:85 01      STA $01
0832:BD 8C C0   LDA $C08C,X
0835:10 FB      BPL $0832
0837:C9 DD      CMP #$DD
0839:D0 F7      BNE $0832
083B:BD 8C C0   LDA $C08C,X
083E:10 FB      BPL $083B
0840:C9 AD      CMP #$AD
0842:D0 F3      BNE $0837
0844:BD 8C C0   LDA $C08C,X
0847:10 FB      BPL $0844
0849:C9 DA      CMP #$DA
084B:D0 EA      BNE $0837
084D:BD 8C C0   LDA $C08C,X
0850:10 FB      BPL $084D
0852:38         SEC
0853:2A         ROL
0854:85 02      STA $02
0856:A5 01      LDA $01
0858:C9 08      CMP #$08
085A:F0 10      BEQ $086C
085C:BD 8C C0   LDA $C08C,X
085F:10 FB      BPL $085C
0861:25 02      AND $02
0863:91 00      STA ($00),Y
0865:C8         INY
0866:D0 E5      BNE $084D
0868:E6 01      INC $01
086A:D0 E1      BNE $084D
086C:4C 1F 04   JMP $041F
086F:D2         ???
0870:A6 AD      LDX $AD
0872:5D B6 F0   EOR $F0B6,X
0875:08         PHP
0876:EE BD B5   INC $B5BD
0879:D0 03      BNE $087E
087B:EE BE B5   INC $B5BE
087E:A9 00      LDA #$00
```

After the preliminary stuff at locations $801-$82D, you will see LDA #$04, STA $01 at $82E. This is the location where the rest of track 0 is loaded: $400-$7FF. Change the $04 at $82F to $14 to change the loading location to $1400, then write the sector back to Sector 0 of Track 0. If you then boot your single-track disk, the loader will be stored at $1400-$17FF (it will probably re-boot after a few seconds - we'll see why in a minute). Interrupt it with a reset, and look at locations $1400-$17FF. Write down the byte at $1400! You have now captured the Sirius loader but before we discuss it, lets save it under DOS. Boot a 48K Slave Disk - not a master (this way no memory between $0900 and $95FF is touched during the boot), and do BSAVE LOADER, A$1400, L$400 now let's look for a second at the track that the loader was loaded from - we'll need to know before this is over.

Using inspector, NA II, or LS 4, do a nibble read of Track 0, and locate the string "D5 AA 96". As everyone(?) knows, this will locate the start of a sector. (In this case the only DOS 3.3 sector on the track). About $180(hex) bytes later, you will find a string "DD AD DA" (a tradition at Sirius) look at the length of this sector - it's certainly not normal DOS! Go back to the preloader listing and look at the sequence from $832 to $84C which is looking for these three bytes in sequence on the track. A careful study of the code from $84D to $86C would explain why the sector is so long - it keeps on loading in bytes (really nibbles) until the page counter at 01 becomes 8 (CMP #$08 at 858). Since we didn't change this, the disk kept on loading, trying to find an 08 after we started at 14! Notice on your nibble read that the nibbles used after the "DD AD DA" marker, are only A, B, E, and F. The reason is that the sector is "encoded" using the "old" frequency modulation technique described as 4+4 nibblizing on page 3-14 of "Beneath Apple DOS" (called B.A.D. henceforth). To see quickly how it's done, write down the fifth and sixth nibbles after the marker: FB AE. the FB byte, in binary, is:

```
1 1 1 1 1 0 1 1
```

Follow the instructions at location $852, and set the carry bit, then rotate left once, with the carry:

```
          C
          1  1 1 1 1    1 0 1 1
ROL <=One 1  1 1 1 1    0 1 1 1
```

Then get the "AE" byte, which is:

```
1 0 1 0 1 1 1 0
```

Next, do a logical "and" of the two bytes, as directed by LOC $861: (Remember, for the result to be a "1" in an "and" operation, both bits being compared must be "1"):

```
           1 1 1 1 0 1 1 1
"AND"      1 0 1 0 1 1 1 0
           ---------------
Result =   1 0 1 0 0 1 1 0
```

Which is "A6" in hex. This is the byte stored in memory in the loader file at location $402 (for us, $1402). (We did the third byte because the first two were $EA, which doesn't show the principle). Compare it to the byte loaded in at $1402. If this is new to you, try making the next few bytes out of the nibble pairs which correspond to them from the nibble read - nibbles 7 & 8 make byte 4 (loc $1403), and so on.

Ok, so that's how they load in the loader, let's get down to serious business. Notice the "JMP $41F" instruction at $86C - this is the jump into the loader routine. A disassembly of the first part of the loader code follows, taken from locations $1400 up.

```
1400:EA         NOP
1401:EA         NOP
1402:A6 34      LDX $34
1404:BD 8A C0   LDA $C08A,X
1407:BD 89 C0   LDA $C089,X
140A:A0 64      LDY #$64
140C:A9 64      LDA #$64
140E:20 89 07   JSR $0789
1411:88         DEY
1412:D0 F8      BNE $140C
1414:A6 34      LDX $34
1416:BD 8E C0   LDA $C08E,X
1419:EA         NOP
141A:EA         NOP
141B:EA         NOP
141C:4C 51 04   JMP $0451
141F:86 34      STX $34
1421:BD 8E C0   LDA $C08E,X
1424:A9 00      LDA #$00
1426:85 26      STA $26
1428:EA         NOP
1429:EA         NOP
142A:4C CF 07   JMP $07CF
142D:AE EE BB   LDX $BBEE
1430:FF         ???
1431:AB         ???
1432:FF         ???
1433:AF         ???
1434:BB         ???
1435:44         ???
1436:00         BRK
1437:FF         ???
1438:A9 02      LDA #$02
143A:85 57      STA $57
143C:A9 00      LDA #$00
143E:A0 00      LDY #$00
1440:59 00 04   EOR $0400,Y
1443:59 00 05   EOR $0500,Y
1446:59 00 06   EOR $0600,Y
1449:59 00 07   EOR $0700,Y
144C:C8         INY
144D:D0 F1      BNE $1440
144F:85 2C      STA $2C
1451:A5 34      LDA $34
1453:4A         LSR
1454:4A         LSR
1455:4A         LSR
1456:4A         LSR
1457:18         CLC
1458:69 C0      ADC #$C0
145A:85 33      STA $33
145C:A9 00      LDA #$00
145E:85 32      STA $32
1460:A5 2C      LDA $2C
1462:F0 03      BEQ $1467
1464:6C 32 00   JMP ($0032)
1467:A9 90      LDA #$90
1469:8D 62 04   STA $0462
146C:A5 32      LDA $32
146E:8D FE 03   STA $03FE
```

The first thing to notice in the listing is that the bytes from $42D to $434 are not code, and that the program jumps around them (as with most cracking work, if it looks suspicious, chase it down!). The "JMP $7CF" goes to a routine which clears all of memory from $800 to $B800, then jumps back to $438 (notice that references are made without the "1" in front of the address just as the disassembled code does). The program next sets up location $57 as the track counter (actually twice the track number, since half-tracks are counted), and does a checksum on the screen memory program (loc $143C to $144F). The checksum result, if it equals 0, is stored in $2C. We'll see later that it's necessary to avoid this to do the crack. After setting up trap vectors for reset, IRQ, and NMI interrupts at $3F0-$3FF, the actual loading begins.

Before the program is loaded, all the active tracks on the disk are checked by reading them in and checking the track checksum. This is the "quick check" that the Sirius DOC always mentions. A destination address is picked out of a table at loc $7AB-$7BC (for CYCLOD: This table varies for each game), and the read head (arm) is moved to the right track. The listing below shows what happens next:

```
1500:A9 FC      LDA # $FC
1502:85 EA      STA $ EA
1504:A0 00      LDY # $00
1506:BD 8C C0   LDA $ C08C,X
1509:10 FB      BPL $ 1506
150B:D9 2D 04   CMP $042D,Y
150E:F0 07      BEQ $1517
1510:A0 00      LDY #$00
1512:D9 2D 04   CMP $042D,Y
1515:D0 EF      BNE $1506
1517:C8         INY
1518:C0 08      CPY #$08
151A:90 EA      BCC $1506
151C:BD 8C C0   LDA $C08C,X
151F:10 FB      BPL $151C
1521:C5 53      CMP $53
1523:D0 3D      BNE $1562
1525:BD 8C C0   LDA $C08C,X
1528:10 FB      BPL $1525
152A:38         SEC
152B:2A         ROL
152C:85 3F      STA $3F
152E:BD 8C C0   LDA $C08C,X
1531:10 FB      BPL $152E
1533:25 3F      AND $3F
1535:85 42      STA $42
1537:20 9F 05   JSR $059F
153A:AD 50 C0   LDA $C050
153D:AD 57 C0   LDA $C057
1540:A6 34      LDX $34
1542:A9 71      LDA #$71
1544:AD FE 07   LDA $07FE
1547:A9 00      LDA #$00
1549:49 21      EOR #$21
154B:4D FD 07   EOR $07FD
154E:A5 41      LDA $41
1550:C5 42      CMP $42
1552:F0 2D      BEQ $1581
1554:A9 14      LDA #$14
1556:20 88 05   JSR $0588
1559:C6 43      DEC $43
```

```
155B:10 21     BPL  $157E
155D:A9 3C     LDA  #$3C
155F:20 88 05  JSR  $0588
1562:A9 06     LDA  #$06
1564:85 43     STA  $43
1566:C6 44     DEC  $44
1568:30 0C     BMI  $1576
156A:A9 5A     LDA  #$5A
156C:85 26     STA  $26
156E:A9 00     LDA  #$00
1570:20 2E 07  JSR  $072E
1573:4C B0 04  JMP  $04B0
1576:A9 FF     LDA  #$FF
1578:20 88 05  JSR  $0588
157B:6C 32 00  JMP  ($0032)
157E:4C B0 04  JMP  $04B0
1581:E6 57     INC  $57
1583:E6 57     INC  $57
1585:4C AC 04  JMP  $04AC
```

The program begins to search the track for the 8-byte sequence that it jumped around at loc $42D to $434. This is a unique sequence used to start each track on the disk; it varies from game to game. (Those of you who are thinking that you now have enough information to copy the disk with NA or LS are wrong. So far, we have only seen a few of the really sneaky things that Sirius has in store for us). When the sequence is found, the track is loaded, starting at the location picked from the table. Each track is a single sector, in 4+4 "FM" encoding, which loads twelve consecutive pages in memory, without any buffers or extra translation - That's why the load is so fast!

Now comes the really sneaky part! (The listing is not included, since it's long and obscure, but try to follow the procedure outlined below). Sirius is fooling around with the timing of the nibble read from the track, in a most devious way. In a normal disk read, you want to be sure that no bits slip away, so you monitor the input latch from the read head on the disk. Look back at the instructions at $832-$84B. The combination of "LDA $C08C,X" and 'BPL $832' means: Keep checking the latch, and when the 8th bit is no longer a 0, take the nibble and run. (By definition, the leftmost or first bit is always a one in the disk nibbles used, in DOS 3.3 as well as the Sirius FM encoding). On average, a new nibble is "built up" a bit at a time every 32 microseconds, and if you want to be sure to get all the data stored, you must come back and empty the latch every 32 microseconds during a read. Sirius, however, recorded the track in a different timing pattern (sort of a stutter-step), and a specific matching pattern must be used to read it out. Their code for doing this runs from $59F to $6FE, and reads in a carefully timed pattern for an 8-byte series. The pattern repeats every 8 bytes, but there is additional jiggery-pokery being done with a variable offset byte in location $EA to further confuse the issue. This is why, although both Nibbles Away and Locksmith can read the tracks given the address marker, the bytes read in at normal 32-usec timing rates are never correct when read by the loader off the copy disk.

After loading in 12 pages ($C00 locations) and checking the checksum, the track number is incremented twice (loc $581-$584), and the destination for the new track is picked from the $7AB table. This continues until a zero is found in the table, where the program jumps to $6FF to decrypt all the data in memory with an old-fashioned exclusive-or technique. Having loaded and "unhid" all of the program, it jumps to location $8EAG to begin the game.

You will note from the load process that each track is always loaded into the same range of memory, since the loader always picks the starting location from the table at $7AB up. It is possible, then, to use the loader to load the program into memory for the first real steps in cracking the program. Our eventual goal is, as always, to save the program as a binary file.

To begin the process, load your single track loader into locations $1400-$17FF. Change locations $1440-$1442 to "4C 4F 04" to avoid the checksum on the screen memory, then change $172D-$172D to "4C 59 FF" (Jump to Reset): normally, we would just insert a 00 (BRK) instruction, but Sirius has, as usual, trapped the break vector to a reboot routine. The following changes make life easier for the intrepid cracker - change locations $1402-$1403 to "A2 60" (put 60 in the X-REG to reference slot 6 for all disk operations), then change $141C-$141E to three nop's - EA EA EA. This routine should be saved to a normal DOS disk by "BSAVE HALTLOAD, A$1400, L$400". When run, it will load the program, decrypt the code, and halt in the monitor after a reset.

Put the original in drive 1 (it is write-protected, isn't it?), and type in, from the monitor:

400<1400.17FFM
400G

The drive will run and rapidly load in tracks $1-$11. The load locations of these tracks, taken from the table at $7AB are:

Track #	Start	End
1	4000	4BFF
2	4000	4BFF
3	4000	4BFF
4	4000	4BFF
5	4000	4BFF
6	4000	4BFF
7	4000	4BFF
8	0A00	15FF
9	1600	21FF
A	6000	6BFF
B	6C00	77FF
C	7000	7BFF
D	7C00	87FF
E	8800	93FF
F	9000	9BFF
10	9600	A1FF
11	A200	ADFF

There are two interesting things about the list, and one suspicious. Sirius was kind enough to leave most of both HI-RES page open to us, so you can "fold-in" some of the program where its feet stick out from under DOS's blanket at $9600 (actually $9D00). Second, there is some overlapping among the tracks; the order in which they are loaded could be crucial. Finally, the fact that tracks 1 to 7 load in from $4000 to $4BFF probably indicates that they get loaded in at level changes (we know there are 20 levels, so that doesn't sound quite right, but keep it in mind). Type 2200<9600.ADFFM to put the high stuff from $2200 to $37FF. Next, boot a slave diskette (remember that booting a slave diskette only destroys $800-$8FF and leaves $900-$9D00 untouched, while booting a master wipes out $1B00-$3FFF), and save the game with BSAVE CYCLOD1, A$A00,L$8C00 (if you get a range error, trying to save a long binary file, you need to change location $A964 from $7F to $BF). This file contains almost all of the memory required to run the game, but the crucial parts at $0-$7FF are missing. To catch this part of memory normally requires a modified "F8" ROM, such as the KRAKROM (much more about this subject in future episodes), but we can do it with software in this case, since we have a clean "halt" location to reference from.

Load in haltload and this time change locations $142A-$142C to "4C 38 04" to avoid the memory wipe routine at LOC $7CF. Change $172B-$172D to "4C 00 08"; add the following short routine:

```
800:    LDY #0         See below
        LDA $00,Y
        STA $1000,Y
        INY
        BNE $802
        INC $805
        INC $808
        LDA $808
        CMP #$14
        BNE $802
        JMP $FF59
```

This is a standard move routine which puts the contents of zero page, the stack, the keyboard buffer and $300-$3FF up at locations $1000-$13FF. Since we "jump" to location $8EA6 to begin, we don't need to worry about subroutine returns and the stack pointer, and the processor status word is probably okay as it sits. Since locations $400-$7FF contain the loader program which is totally useless for a DOS disk, it need not be saved. Notice that it's better to write this routine with the LDA $00,Y since there is no LDA $00,Y which refers specifically to zero page as there is for LDA 00,X. (Keeps the mini-assembler from screwing you up).

Again, type in 400<1400.1820M 400G and await the reset beep. You can now boot a slave (a little S&M) and save this stuff as CYLOW,A$1000,L$400. now reload your CYCLOD1 file, load CYLOW at $5000, and BSAVE the new file as "CYCLOD2,A$A00,L$4C00."

Now, with the game nestled all safe and snug in binary files, it's time to see if we can do something about those disk accesses which occur every time we elevate to a new level. Experience has taught that a disk access under this system is a "JSR $400". You can puzzle it out if you stare at the code long enough, but take my word for it for now. Searching through memory with the Inspector in "find" mode set for 20 00 04, you will find only one call (this is in marked contrast to Bandits, where there were three separate calls, each obscured with a slightly different exclusive-or technique and a complex algorithm to compute the EX-OR byte). You should appreciate by now how important it is to avoid any disk accesses, since the old Sirius loader is useless for normal DOS, and putting the files into specific tracks for RWTS access is at best wasteful of disk space, and at worst not possible (Bandits, again) due to memory space. Let's spend a few minutes then to analyze the code surrounding the disk call at $8262:

```
8236:00        BRK
8237:A9 30     LDA #$30
8239:85 53     STA $53
823B:AD 45 70  LDA $7045
823E:A2 00     LDX #$00
8240:8E 35 82  STX $8235
8243:C9 04     CMP #$04
8245:30 09     BMI $8250
8247:38        SEC
8248:E9 03     SBC #$03
824A:EE 35 82  INC $8235
824D:4C 43 82  JMP $8243
8250:8D 36 82  STA $8236
8253:EE 35 82  INC $8235
8256:AD 35 82  LDA $8235
8259:0A        ASL
825A:85 57     STA $57
825C:18        CLC
825D:69 01     ADC #$01
825F:8D 37 04  STA $0437
8262:20 00 04  JSR $0400
8265:CE 36 82  DEC $8236
8268:AD 36 82  LDA $8236
826B:0A        ASL
826C:0A        ASL
826D:0A        ASL
826E:8D 00 70  STA $7000
8271:0A        ASL
8272:18        CLC
8273:6D 00 70  ADC $7000
8276:85 00     STA $00
8278:A9 40     LDA #$40
827A:85 01     STA $01
827C:A0 17     LDY #$17
827E:B1 00     LDA ($00),Y
8280:99 00 10  STA $1000,Y
8283:88        DEY
8284:10 F8     BPL $827E
8286:A5 53     LDA $53
8288:09 15     ORA #$15
828A:C9 BF     CMP #$BF
828C:F0 05     BEQ $8293
828E:A9 01     LDA #$01
8290:8D 9D 7B  STA $7B9D
8293:A9 40     LDA #$40
8295:8D 5B 70  STA $705B
8298:A9 60     LDA #$60
829A:8D 5C 70  STA $705C
829D:20 CC 76  JSR $76CC
82A0:A9 20     LDA #$20
82A2:8D 5B 70  STA $705B
82A5:A9 40     LDA #$40
82A7:8D 5C 70  STA $705C
82AA:60        RTS
```

The routine from $8237 to $8264 determines which track to read in by looking at the game level in location $7045. If the level is above 3, it subtracts 3 and increments location $8235. This becomes the track number to load from, as follows:

Level	Track
1-3	1
4-6	2
7-9	3, etc

And location $8236 contains the remainder after the Track * 3 is subtracted. After the track is loaded (JSR $400), this number is manipulated to give $0, $18, or $30 (hex) which is stored at Location 0. The $18 bytes pointed to by 0 & 1 are then stored at $1000-$1017:

Level	Locations	Track#
1	4000-4017	1
2	4018-402F	1
3	4030-4047	1
4	4000-4017	2
5	4018-402F	2
6	4030-4047	2, etc

The routine at $8288 checks to see if you accessed the right disk (or just maybe checks to see if you didn't do it), and then clears all of both(!) HI-RES pages at $8293-$82AA.

Note Carefully

Since the rest of the track that was loaded in at $4000-$4BFF is wiped by the screen clear, only those $18 bytes were really used to establish the game level after accessing the disk. Obviously, Sirius is making it unnecessarily hard in order to use the disk and make life difficult for the Crackist. Here's how we get around it: load in your old friend haltload, and change the following locations in the track load address table:

Addr	Old	New
7AC	40	58
7AD	40	59
7AE	40	5A
7AF	40	5B
7B0	40	5C
7B1	40	5D
7B2	40	5E
7B3	0A	00 (To end)

Do the same load routine as we did earlier to get the main program in. This will load in everything we need for all the levels, and eliminate most of the garbage. Boot the slave again, and BSAVE TRACKS,A$5800,L$700. Next write a short subroutine to pick up the right range of memory and the right group of the three $18-byte level blocks and store it in locations $1000-$1017. Save this routine in memory, and later tuck it into locations $3800-$38FF of the main file. Finally, make one big file which contains all of the above pieces and routines, and write a short memory move routine (or use Masterkey Plus) to unfold all of this "Tucked-in" memory after the program is loaded. The following list is approximately what I used for the single 144-sector binary file:

Routine Name	Storage Location	Unfolded Location
Main PRG	0A00-9600	0A00-9600
Mover	0900-09FF	0900-09FF
Hipart	2200-37FF	9600-ADFF
Levlcalc	3800-38FF	AE00-AEFF
Cylow	5000-53FF	0000-03FF
Tracks	5800-5EFF	B000-B6FF

A couple of minor changes, and we're done: change locations $8262-$8264 to disk, and change $8265-$8267 to "4C 93 82" (JMP to screen clear). Make sure your mover routine ends with a JMP $8EA6 to start the game, and you are set to BSAVE CYCLOD,A$900,L$8D00 as a single file which you can "BRUN" to your heart's content.

Edward Eastman NE

Dazzle Draw Patch to Save the Configuration

Requirements:
Issues #21 & #59 or a softkeyed Dazzle Draw from #21
A sector editor

In this issue I redo yet another of sombody else's work. I show you how to apply Bill Jetzer's configuration save routine for Dazzle Draw in issue #59 onto Clay Harrel's softkey from issue #21. If you already have made a backup using Clay's softkey, jump to the sector edits below.

For those of you who have not yet made a backup, follow Clay's softkey in #21 except for the following. Do step five from issue #59 in leu of step eleven and change the byte at 7138 from 18 to F0. Skip step 17. Because you skip step 17, ignore the sector edit in step 27, but do NOT forget to copy track zero sector zero from a ProDOS disk onto your new backup. Also, perform the sector edit in step eleven from #59.

That's it, you are done unless you want to change your quit routine from a reboot to ProDOS's quit routine. Do step 13 in #59 to change the quit routine and messages, this is especially useful if you have a friendly quit code. The edits are in the same place on the older version, just be sure to put the new ProDOS in the subdirectory over the old file.

The following is for deprotects already done with #21. *Note:* All sector scans can start at track $1F.

Scan bytes	Change to bytes
4C C4 64	4C 23 65
A9 40 20 19 61 A0 00 84	BD 8A C0 BD 89 C0 18 60
A9 F5 85 00 A9 F4 85 01	20 00 BF CB 3B 71 B0 06
A9 F3 85 02 A9 F2 85 03	20 00 BF CC 43 71 60 03
A2 0D A9 04 20 2A 71 A2	24 71 00 20 00 11 2F 44
0C A9 60 86 3C 85 3B A0	44 2F 33 41 5A 5A 4C 45
00 84 3A A6 2B BD 8D C0	2E 53 59 53 54 45 4D 02
BD 8E C0 10 02 38 60 A0	00 F0 07 00 04 00 F0 66
08 A9 FF 9D 8F C0	05 00 00 00 01 00
20 F3 73 A9 1F 20 09 67	20 00 BF C8 1E 71 B0 12
A9 60 85 50 A0 00 A6 2B	AD 23 71 8D 37 71 8D 3C
88 D0 06 C6 50 D0 02 38	71 8D 44 71 20 00 BF CE
60 BD 8C	36 71 60

To finish, do the sector edit in step 11 and remove the bogus ProDOS file from the root directory. See last paragraph above.

Alan Chaney MD

Softkey for...

Clue
Risk 1.4
Leisure Genius

Requirements:
COPYA
Sector Editor

I looked thru my back issues for a softkey to this program and found Risk 1.3, which didn't work on this version. Well, I might as well start from the beginning. Fast copy prove to be not the way to go. I found that the program is ProDOS based, after starting the original (with a tab on it). With that info, I decided to get out Mr. B. Dudley Brett's article on Reading Protected ProDOS Disks, issue #67 page 9.

Copy II plus would not read the sectors, But it will tell you sometimes what epilog bytes that was read (which was AA DE EB). I decided to use Mr. Brett's article on MECC ProDOS Software (same issue and page as above), since that softkey seem to be close to the disk I was trying to softkey.

Step-by-step

1. Use COPYA to copy the disk.
```
RUN COPYA
ctrl C              at prompt
POKE 47397,24
POKE 47398,96
70
RUN
```
2. Scan disk for 10 30 AA DE EB FF and change AA DE to DE AA.
3. Scan disk for 10 FB C9 AA 18 F0 and change AA to DE.

That's a rap!

Softkey for...

The Scoop
Spinnaker

Requirements:
1 Blank 5.25" disk
Sector Editor
Any copy program

This programs protection requires you to look for a code word in the manual. You only get two chances to type in the correct word. After your second try fails, you are asked by the program to press control-reset to start the program again. Sorry, But I am not smart enough yet to translate the code that I found on the disk (Maybe I will be that smart in my next life!).

Step-by-step

1. Copy both sides of the original program.
2. Scan copy for 8D 10 C0 20 1D 59 AD 00 C0.
3. Change 20 to 60 and write sector back to copy.

Put manual away and Swoop the Coop.

Advanced Playing Technique for...

The Duel: Test Drive II GS
Accolade

To break in on this program on the II GS, Scan for F4 01 00 A2 03 23 and change 01 to 00, write change back to copy (You will not be able to return to the game once you are in the monitor). Start game and complete course 1. When the program asks "FILL'ER UP", take game out of drive and insert any other "write protected disk" in the drive and press return or joystick button. When the game starts to pole the drives for the Duel disk, HOLD DOWN open apple-control-esc.

Make these patches:

Tickets

Scan for	Change to
EE B3 D3	EA EA EA or 9C B4 D3

Crashes

Scan for	Change to
EE B0 D3	EA EA EA or 9C B0 D3

Out of gas

Scan for	Change to
EE B6 D3	EA EA EA or 9C B6 D3

Lives left

Scan for	Change to
CE 5F D0	EA EA EA or EE 5F D0

Lives at start of game

Scan for	Change to
D0 03 A9 **05** 00 8F	Change 05 to # (Hex)

Police won't stop you. (If lives is set to EE 5F D0 you can crash police car without ending game).

Scan for	Change to
7C 22 D2 A5 E6 30	Change A5 to 60

Note: You can break out of the game and get to the control panel, but the keyboard does not work. That is because the keyboard is turned off with this code F4 01 00 A2 03 23. By changing 01 to 00 this turns the keyboard back on allowing you to operate the control panel.

Softkey for...

Word Attack Plus
Davidson & Associates

Word Attack Plus shows a bad block error when you copy with Copy II Plus fast copy.

Step-by-step

1. Copy disk with any fast copier.
2. Scan the copy for 85 FF 60 A9 00 85 FF 60.
3. Change 00 to FF and write sector back to copy.

Mr. Ross's article (issue #74, pg.11) could be useful on any Davidson ProDOS program.

Locksmith Fastcopy (2 GS) Help

Requirements:
Issue 43 and 50
Sector Editor

While flipping thru issue #71 I came across Mr. Brett's article on Locksmith 6.0 Fastcopy with E.A. RWTS (revised) page 16. Well, let's give it a try. After following Mr. Lewis's article in #43 page 12 to the T, I was disappointed when I ran E.A. RWTS program. It seem to be the LS 6.0 fastcopy to be at fault, because I kept getting a break at 1D00 in the monitor. Then I saw Mr. Romine's article in issue 50 page 37, for how to save the fastcopy for a 2 GS, and thought this would get my E.A. RWTS program running since it talked about the problem I was having on my GS. But after a day or 3 of work with issues 43,49,50,55,56 no go on the RWTS program. Then out of pure DESPERATION, I decided to compare the fastcopies made in issue 43 to issue 50 as Mr. Romine had done in issue 50. I bloaded issue 43's fastcopy at A$2000 and bloaded issue 50's fastcopy at A$4000. I then verified the two programs in the GS monitor (2000<4000.4300V). Other than the written changes in the two articles 2000-2012, there were two other differences that showed up. In issue 43's fastcopy addresses 2041:0C and 2042: A9, But in issue 50's fastcopy these addresses showed 4021:CF and 4022:21, These changes were not written in the text file that was created. Now comes the weird part, 43's fastcopy showed 2141:CF and 2142:21 which was written in the text file. 50's fastcopy showed 4141:FA and 4142:22, But the text file was written to place a CF at 2141 (4141) and a 21 at 2142 (4142). Maybe someone can explain why this happened, because its way out of my hands.

1. Use issue 43 to make the Fastcopy program, But substitute issue 50 to make FC file and text file.

Note: Issue 50's (SAVE FC,A$2000, L$18FD) should be (BSAVE FC, A$2002,L$18FD). (Issue 55, page 37, Mr.Cook.)

2. Scan for FA EE 94 CF 21 91 8D and change CF 21 to 0C A9 (Write change to disk).
3. Scan for 48 BD FA 22 95 80 and change FA 22 to CF 21 (Write change to disk). Change may occur twice on disk due to the text file change and the fastcopy change, But the second occurrence is the Fastcopy.

Note: Maybe I did something wrong in those instructions, but now I have a Locksmith 6.0 Fastcopy with E.A. RWTS program.

Fastcopya Enhancement problem

Requirements:
Issue 68,72 and 78

While trying to enhance Fastcopya in issue 78, I must have spaced out again, because I could not figure out where the patch for Mr. Reid's track selection (issue 68, page 20), was to be placed in Super 6.0 Fastcopya. Mr. Brett said the patch should go in lines 284,288,322-328 (That may not be what he meant, but that the way I understood it). Well as usual I had a problem. The program kept saying there was a mismatch error in line 288. So I put the patch at 1381, 1382, 1383, 1384, 1385 (Which is the end of the program Super 6.0 Fastcopya, then I changed line 290 to read as: 290 IF X=4 AND FL=2 THEN 1381 REM EXIT (Exit= to track selection program). Since this program patch also exits with a Call 8192 (Which is the call to run Locksmith 6.0 Fastcopy), I felt removing the same call from line 290 would not change the program. Change both (THEN 10) statements in lines 1382 and 1383 to (THEN 1381). Thanks goes out to Mr. Brett and Mr. Reid for such a fine job.

Note: Line #70 reads:

```
70 CHR$ (4;"RUN
```

It should be:

```
70 CHR$ (4) "RUN"
```

...but it still runs.

Question and Help for Wings Of Fury by Broderbund

In issue #65 page 30, Mr. Dave Morgan gave 3 APT's for Wings Of Fury, Where he used sector edits pass track 1 sector F (which is how far I got before the sector editor quit reading). Mr. Morgan can you or anyone that reads this article, give a how to edit Wings Of Fury article, So I can make permanent changes on the disk (Then my kids won't call me every time they need more planes).

While I am on the subject of Wings Of Fury here is a helper. 01/2027:# and 01/203A:# Holds the number of planes you start the game having. Address 01/09BD:# holds the number of hits your aircraft carrier received so far. If this number reaches 4 your carrier will sink. I already tried to zero the EE BD 09 in bank 1 (No happenings). So I used Mr. Morgan's patch CE 86 09 to 9C BD 09 there are 2 changes to be made in bank 01. What this change does is zero the address 09BD (Which is the hits on your ship) every time you lose a plane. So if you get 3 hits on your ship, crash a plane quick. Now the enemy planes need to hit your ship 4 more times before it sinks. If you don't want your ship attacked by enemy planes change 01/5A5D:B0 to 80 (for those who can scan the disk AD BD 09 C9 04 B0 37).

If up was down and down was up, where would the middle be?

Good evening everyone.

Rich Etarip WI

Softkey for...

Airheart
Broderbund

To the best of my knowledge, this is the FIRST released Softkey for a Broderbund 18 sector-per-track disk. Even the 'cracked' copy of Airheart released by one of the pirate organizations was just a working bit copy which warned me that an attempt to crack this disk might be a waste of time...but it turned out not to be.

I left the Airheart disk sit and collect dust for a while before I picked it up and started examining the format and how it loads. I knew that if I learned how to work their DOS, I could read the disk track by track and write it back to a normal disk. There was one problem. Except for track 0, Airheart has 18 sectors per track as opposed to the normal 16. Each track is divided into 6 sectors, but each sector is 768 bytes long which is the equivalent of 3 normal DOS sectors. We have 34 tracks of 18 sectors (612 sectors) which would use approximately 39 tracks of 16 sectors. It is possible to format a disk for 40 tracks but not all disk drives are capable of that. Plus, with 40 tracks, you can't use a normal disk copier to make copies. However, since Airheart is only one-sided, that leaves an entire second side to work with. It's quite convenient that the opening picture is stored on tracks 1-4 and is never reloaded after the game starts. With that in mind, I decided to write tracks 0-4 on side 1 of the copy and tracks 5-22 on side 2 and insert a keypress routine so the disk can be flipped once the picture is displayed.

To begin, freshly initialize both sides of a 2 sided disk. Then, using a copier that allows you to select tracks (a bit copier will also work) copy track 0 from Airheart to side 1 of the copy disk. Then run your sector editor and make the following patches:

Trk	Sct	Byte	From	To
$00	$00	$C0	?	8D 0C C0 AD 82 C0 4C 59 FF
$00	$02	$4F	?	A9 5C EA
$00	$02	$7C	?	4C C0 08

This causes the disk to partially boot in order to get the Airheart DOS in memory. This could also be accomplished by boot code tracing but this method is easier.

Exit the sector editor and boot side 2 of the copy disk. Then flip the disk over and boot side 1 with a PR#6 command. This is to assure that both the RWTS and the Airheart DOS are in memory. The disk drive will still be running so turn it off.

```
PR#6
C0E8
```

Airheart's DOS is operated by a JSR $D000 followed by a command code and one or two bytes used by the command. For instance, the command code to read a track is $C3 followed by the page in memory to read it. The loader is stored in the RAM card but it can't be used until you read enable the RAM card.

```
C081 C081 N F800<F800.FFFFM
C08B
```

Write a short routine at $1000 to call the Airheart loader telling it to read an entire track starting at $2000. After reading a track it will increment the page # by $12 to tell it where to read the next track, then it will jump to the monitor.

```
1000:20 00 D0 C3 20 AD 04 10
:18 69 12 8D 04 10 4C 59 FF
```

Each time $1000 is executed, the track stored in $FE will be read into memory. The DOS automatically increments the track number if the high bit is set on the command code ($C3). In order to read multiple tracks, type '1000G' for each track. I tried reading with a routine using a loop but it doesn't work correctly. All of Airheart's loading is done with individual calls to $D000.

We'll begin by reading the first 4 tracks that contain the picture. These are the only tracks that will be written to side 1. After each '1000G' the computer will beep.

```
(insert Airheart)
C0E9
1000G                          (4 tracks)
1000G
1000G
1000G
C0E8
```

The RWTS is still intact and will be used to write these tracks to the copy disk. Remember, Airheart contains 18 sectors per track so the write will use more than 4 tracks. Before writing, install a reverse sector skew in the RWTS. This is to maximize loading speed when booting the copy disk. The write process will take a bit longer than usual but it's just because the sectors are being written with a reverse skew. Occasionally, the disk drive may recalibrate at the beginning of the write process but don't be alarmed.

```
(insert copy disk side 1)
BFB8:00 02 04 06 08 0A 0C 0E
:01 03 05 07 09 0B 0D 0F
B7E1:48
B7EC:05 07 FB B7 00 67 00 00 02
B793G
```

This time we'll read 8 tracks. Reset the page pointer to $20 and store the current track times 2 in $FF so Airheart's seek routine knows what track the read/write head is positioned at. This is necessary in order for it to seek the correct track. Be sure to carefully type '1000G' EXACTLY 8 times.

```
(insert Airheart)
1004:20
FF:02
C0E9
1000G                          8 tracks
1000G
1000G
1000G
1000G
1000G
1000G
1000G
C0E8
(insert copy disk side 2)
B7E1:90
B7EC:09 0F FB B7 00 AF
B793G
(insert Airheart)
1004:20
FF:02
C0E9
1000G                          8 tracks
1000G
1000G
1000G
1000G
1000G
1000G
1000G
C0E8
(insert copy disk side 2)
B7E1:90
B7EC:12 0F FB B7 00 AF
B793G
(insert Airheart)
1004:20
FF:14
C0E9
1000G                          7 tracks
1000G
1000G
1000G
1000G
1000G
1000G
C0E8
(insert copy disk side 2)
B7E1:7E
B7EC:1A 0D FB B7 00 9D
B793G
```

Insert Airheart for the final read pass. Also, tell it to start reading on Track $1D. Right now, it is set to read track $1C but for some strange reason, track $1C is not readable by the DOS.

```
1004:20
FE:1D 26
C0E9
1000G                          6 tracks
1000G
1000G
1000G
1000G
1000G
C0E8
(insert copy disk side 2)
B7E1:6C
B7EC:22 0B FB B7 00 8B
B793G
```

Step 1 is now complete, but unfortunately, that was the easy part. What we need to do now is rewrite Airheart's DOS using normal DOS read routines and then perform a few sector edits but this involves quite a bit of typing. To make it harder, Airheart uses two loaders. There is the boot loader on track 0 of side 1, and another main game loader on track 1 of side 2. The first one we'll modify is the boot loader at $D000. It should still be in memory so move it down to $2000 so we don't have to deal with the RAM card.

```
2000<D000.D4FFM
```

Follow the 'cookbook procedure' below to convert Airheart's DOS to read from a normal disk. I really can't begin to explain what is being done here. It's a bit complex and really takes a knowledge of DOS to understand it. Type very carefully and it may even be a good idea to double check your typing. One small error could take hours to find. In such typing situations, I find it much easier to have a friend dictate while I type. That way, I never have to take my eyes off the screen.

```
21FD:2C
2246:C7
227B:E0 D0
2288:A9 00
2292:E0 D0
22E0:E0 D0
2400<B944.B96CM
240A:00
2429<B971.B980M
2439:00 D3 88 10 EB 60
243F<B8DC.B924M
2488<B8C2.B8D6M
249D:D0 ED 60
2443:00
246E:D3
2473:D5
247F:D3
2482:91 2E EA
248F:B1 2E EA
2494:D5
2498:D5
2483:2E
2490:2E
249B:2E
2396<BA96.BAFFM
24A0:00 08 01 09 02 0A 03 0B
:04 0C 05 0D 06 0E 07 0F
```

The following is a lookup table for the new DOS to find the correct track. As an example, the original track 5 starts on track 1 sector 0 on side 2 of the copy disk.

```
24B0:00 01 02 03 04 01 02 03
:04 05 06 07 08 0A 0B 0C
:0D 0E 0F 10 11 13 14 15
```

```
:16 17 18 19 1A 1C 1D 1E
:1F 20 21
24D8:00 00 02 04 06 00 02 04
:06 08 0A 0C 0E 00 02 04
:06 08 0A 0C 0E 00 02 04
:06 08 0A 0C 0E 00 02 04
:06 08 0A
20E0:A4 FE B9 D8 D4 85 2C B9
:B0 D4 85 2B 4C 71 D1
```

The next section is the main track loader of the DOS.

```
2084:A0 00 84 2E 84 2A A2 60
:20 00 D4 A4 FE C0 FF 18
:F0 42 AD 02 D3 C5 2B F0
:09 0A 85 FF 20 E0 D0 4C
:84 D0 AC 01 D3 B9 A0 D4
:C5 2C D0 DA A4 2A B9 43
:D3 85 2F A2 60 20 3F D4
:E6 2C A5 2C C9 10 D0 0B
:A9 00 85 2C E6 2B A5 2B
:20 71 D1 E6 2A A5 2A C9
:12 D0 B3 18 AD 02 D3 60
```

The new Airheart DOS is complete. Insert side 1 of the copy disk and write it to track 0, sectors 8-C. Before doing this, restore the DOS 3.3 sector skew. Track 0 is the only track with a normal skew.

```
BFB8:00 0D 0B 09 07 05 03 01
:0E 0C 0A 08 06 04 02 0F
B7E1:05
B7EC:00 0C FB B7 00 24 00 00 02
B793G
```

Unfortunately, we have to do the same thing over again with the second DOS loader. The memory usage of this second DOS is somewhat different from the first, but most of the rewritten routines can be relocated into this DOS. First of all, flip the disk over, install the reverse skew, and read in the text page DOS.

```
BFB8:00 02 04 06 08 0A 0C 0E
:01 03 05 07 09 0B 0D 0F
B7E1:06
B7EC:01 07 FB B7 00 69 00 00 01
B793G
```

The DOS has been loaded into $6400 through $69FF but it normally runs in the text page at $400. Begin by moving the DOS 3.3 routines from the first loader at $2000.

```
6418<2400.24FFM
6514<2084.20EFM
6996<BA96.BAFFM
```

Now make the following modifications so the DOS routines work correctly in their new location. Once again, type very carefully.

```
6451:80 09
6486:09
648B:09
6497:09
64AC:09
64B0:09
651D:18 04
6527:82 09
6531:70 05
6534:14 05
6537:81 09
653A:B8 04
6543:63 09
654A:57 04
655D:81 05
6569:82 09
6573:F0 04
6578:C8 04
657D:81 05
660D:2C
6656:56
6673:14 05
668B:70
669D:14 05 A9 00
66A9:70
66F7:70
66D2:63
66E1:63
66E8:63
66EF:63
```

Write the DOS back to the disk.

```
B7E1:06
B7EC:01 07 FB B7 00 69 00 00 02
B793G
```

At this point, I would consider the softkey 95% complete. There are just a few finishing touches to get this disk working correctly. On track $14, sector $A, side 2, there is an encoded half sector that does not decode correctly when you try to run the game. We'll decode it right here and write it back to the disk.

```
B7E1:01
B7EC:14 0A FB B7 00 40 00 00 01
B793G
800:A2 4A A9 FF 5D 00 40 9D
:00 40 E8 E0 EA D0 F5 60
800G
B7E1:01
B7EC:14 0A FB B7 00 40 00 00 02
B793G
```

Now, reboot DOS and run your sector editor. Read track 0, sector 2 from the original Airheart disk and write it back to side 1 of the copy disk. Then do the following sector edits to the copy disk, side 1. These sector edits cause the boot code to wait for a keypress after the picture is loaded in so the disk can be flipped.

Trk	Sct	Byte	From	To
$00	$02	$55	99	2C
$00	$02	$8F	63	E0
$00	$04	$E0	00	20 63 E2 AD 10 C0 AD 00 C0 10 FB AD 10 C0 20 00 D0 C2 FF 05 60

Flip the disk over to side 2.

Trk	Sct	Byte	From	To
$01	$0A	$A9	91	24
$01	$0A	$CC	??	4C F3 07
$14	$05	$79	A5 35	A9 11

At this point, one would guess that Airheart has been cracked—but after play-testing the game for awhile, I found the disk drive to occasionally have problems loading after a game is finished. Even though some people would let this problem slide, I never consider a disk 'cracked' until it works completely. Therefore, I set out to find the root of this problem.

My conclusion was that the disk drive was not always seeking the correct track. In most cases, the best way for a DOS seek routine to find the correct track is for it to first know what track it is currently on. Even though Airheart's DOS uses location $FF for this value, sometimes the value in this byte may not necessarily be the correct value. Even though the 'SEEK' routine works fine on the original disk, it doesn't always work with the normal DOS disk and I wish I could explain why. However, every problem has a solution.

The boot program at $C600 (assuming slot 6) automatically recalibrates the drive head to track 0. This is the sound you hear from the disk drive when you boot a disk. If the drive head is lost, you can recalibrate it back to track 0 and then seek the correct destination track. The disk drive will sound like it is re-booting but it is only seeking track 0.

The question is where to put this routine. When a game is finished, it goes to page $7 which takes care of the loading. At $772 is a decode routine that we disabled earlier. This leaves room for the recalibrate routine. Page 7 is on track 1, sector $A. Make sure side 2 of the copy disk is in the drive.

```
B7E1:01
B7EC:01 0A FB B7 00 47 00 00 01
B793G
471C:75 07
4772:4C F6 07 A2 60
4777<C62F.C651M
478C:09 60
4795:9F 07
479A:A9 00 85 FF 60
479F<FCA8.FCB3M
B7E1:01
B7EC:01 0A FB B7 00 47 00 00 02
B793G
```

And that should do it for possibly the FIRST deprotected version of Airheart. I could be wrong, but I haven't seen one yet. When booting the disk, wait until it displays the picture and then flip the disk and press a key. The copy will not load quite as fast as the original just because of the DOS 3.3 format. If something doesn't work correctly with the copy, remember that typing errors are quite common with an extensive procedure such as this. You may have to go back and check that the DOS modifications were all correct.

My current project is cracking Prince of Persia which is also an 18 sector disk but is 2-sided. I would say that I'm 75% finished with it so far. Also, if anyone has an original or a working copy of WINGS OF FURY, feel free to send it my way. I'd like to give it a shot. See my ad in the back of the magazine.

IBMIBMIBMIBMIBMIBM

Unknown

IBM Softkey for...

Carrier Command

?

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

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

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

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

IBM Softkey for...

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

Broderbund

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

1. COPY "Where in the USA is Carmen Sandiego?" disks to a new subdirectory.
2. Copy DEBUG.COM to the new subdirectory.
3. Patch CARMEN.EXE using DEBUG.

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

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

IBM Softkey for...

Colonel's Bequest

Sierra

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

IBM Softkey for...

Continuum

Data East

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

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

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

IBM Softkey for...

Crime Wave

Access

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

IBM Softkey for...

Crimewave v1.1

Access

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

IBM Softkey for...

Curse of the Azure Bonds

?

Requirements:

Norton Utilities (or similar program)

A copy of the file START.EXE from your Azure Bonds disk A

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

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

IBM Softkey for...

Dragon's Lair II

?

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

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

IBM Softkey for...

Dragon's Lair

?

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

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

That's it! Enjoy!

IBM Softkey for...

Dragon's Lair

?

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

IBM Softkey for...

Earthrise

?

Well it looked like another simple doc check, but these guys are a little sneaky. The game program actually begins in the file SOL.EXE, but it is set up to exit to DOS if you try to run it. You must run EARTHRIS.EXE which then runs SOL.EXE.

EARTHTRIS.EXE was designed to make you think this is the program to tamper with. It overrides INT 3 and give you a "Mind your own business. It's a wild goose chase anyway" message. There is a decisive jump in EARTHRIS .EXE for the DOS exit routine, but altering the program at this point makes a "Security Violation" message appear upon playing. Also the program uses a JMP to decide your answer, not a JZ or JNZ or anything like that as shown below. It calls a routine which then uses a JMP to exit instead of a RET. But by eliminating the "you are wrong jump" in SOL.EXE this game is at your feet.

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

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

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

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

IBM Softkey for...

Escape From Hell

?

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

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

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

<u>offsets</u>
14DFC
14E3A
14E78
14EB6
14EF3
14F1E

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

IBM Softkey for...

Earl Weaver's Baseball v1.5

?

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

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

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

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

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

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

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

IBM Softkey for...

F-15

?

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

1. Start up DEBUG.

```
DEBUG
```

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

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

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

IBM Softkey for...

Gunship

?

To remove the read for original disk, use PCtools or other program and edit START.EXE.

Sct	Offset	From	To
52	296	CD 13	90 90
	306	75 02	90 90
	329	B9 00 06 8B	B8 34 12 BA
		16 36 49 0A	36 2F 8E DA
		F6 74 06	33 D2 90
	341	00 01	90 0D
	344	66	67
	348	CD 13	BE 93
	350	72 DC 9A 01 67	0F EB 04 03 97
90	419	CD 21 73	90 90 EB
91	60	B4 3D CD 21	B0 06 90 90
95	204	00 43 CD 21	20 00 B1 20
	212	75 04	90 90
	249	00 44 CD 21 73	40 00 BA 40 00
		05 B8 05 00	EB 04 90 90

To get rid of the ID question:

Sct	Offset	From	To
36	5	74	EB

That's it, no more question.

IBM Softkey for...

Caveman Ugh-Lympics

?

Use Norton to search SOS.EXE for 76 01 E8 BB 48 9A and change the E8 BB 48 to 90 90 90. Write the changed data and your done! No more look up screen!

IBM Softkey for...

Firehawk Thexder II

This deprotect works on file GAME .EXE dated 9/24/90 with a length of 37,378 bytes. This game is a real nuisance to play with the passive protection system requiring you to consult the manual each and every time you boot it up. The game relies on 20 words picked at random from the manual to "prove" that the game player is in possession of an official manual (and is presumably a registered bona fide owner).

To remove this nuisance you can proceed in one of two ways — either get into the trenches and slug it out on an assembly language level using Debug, Periscope or some other debugger to find the pivot point where the program compares your entry to the correct answer and then change the pivot point (JNZ) to a forced branch (JMP) or you can change the stored tables on the disk to make the program think your answer is always correct.

In this particular case the latter seemed the easier choice possibly because I stumbled across the page/paragraph/word table and hence knew where it was. The entries are stored in 5 digraph series (20 entries of 5 digraphs each) with the first three digraphs being the page/paragraph/word-number in Hexadecimal. Numbers 1-9 are the same in Hex or Decimal for the purposes of this encryption process. The other two digraphs point to the encrypted word in some fashion. I did not bother to locate them since it's not necessary to actually find them on the disk for this deprotect.

What we are going to do is change all the word pointers to point to the same word so that no matter what page number/paragraph and word number are selected at random; your entry will be seen as correct.

The page/para/word locations are at 8F58 to 8FBB on my version while the screen text is located at 8E4F to 8F00 (for those who are interested). You can find them for yourself using PCTOOLS FIND function looking for the HEX string "090202".

I chose the fourth word in the series (page 09; para 02; word 02) - SYSTEM as an easy one to remember. I also changed the on screen prompt to prompt you to enter the word "system" to proceed with the game. Any word on the list could have been chosen - however a shorter one is easier to type.

Copy the file GAME.EXE to a disk or subdirectory together with DEBUG .COM.

```
REN GAME.EXE GAME.DAT
DEBUG GAME.DAT
E 8E4F
54
E 8E50
4F 20 50 52 4F 43 45 45 44 20 57 49 54 48
   20 54
E 8E60
48 45 20 20 20 20 00 47 41 4D 45 20 57 49
   54 48
E 8E70
4F 55 54 20 4C 4F 4F 4B 49 4E 47 20 55 50
   20 20
E 8E80
20 00 41 4E 59 20 57 4F 52 44 53 2C 20 4A
   55 53
E 8E90
54 20 54 59 50 45 20 49 4E 20 54 48 45 20
   20 00
E 8EA0
57 4F 52 44 20 27 53 59 53 54 45 4D 27 2E
   20 49
E 8EB0
47 4E 4F 52 45 20 20 20 00 41 4C 4C 20 54
   48 45
E 8EC0
20 50 41 47 45 20 42 55 4C 4C 43 52 41 50
   21 21
E 8ED0
20 20 20 00 20 20 20 20 20 50 52 45 53 53
   20 5B
E 8EE0
45 4E 54 45 52 5D 20 57 48 45 4E 20 00 59
   4F 55
E 8EF0
20 41 52 45 20 44 4F 4E 45 2E 20 00 20 00
   20 00
E 8F00
20
E 8F58
06 01 01 E6 03 06 08 03
E 8F60
E6 03 06 09 06 E6 03 09 02 02 E6 03 09 05
   04 E6
E 8F70
03 1B 05 02 E6 03 1B 08 01 E6 03 1C 02 01
   E6 03
E 8F80
1C 07 04 E6 03 1F 02 04 E6 03 1F 07 04 E6
   03 21
E 8F90
01 05 E6 03 21 02 03 E6 03 23 03 01 E6 03
   23 08
E 8FA0
03 E6 03 25 01 04 E6 03 21 07 05 E6 03 21
   04 04
E 8FB0
E6 03 1C 03 02 E6 03 1C 07 02 E6 03
W
Q
REN GAME.DAT GAME.EXE
```

GAME.EXE in its deprotected form should be copied back to the COPY of Firehawk that you are trying to deprotect.

unClassifieds

How to place an UnClassified Ad

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

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

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

★★★★ · New Rates (per line) ★★★★

Computist club member	25¢
All others	35¢

The minimum order is $5.

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

COMPUTIST unCLASSIFIEDS
33821 East Orville Road
Eatonville, WA 98328

WANTED

"Most Wanted List" Software

Need help to deprotect a disk

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

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

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

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

RDEX Contributors

Alan	Chaney	19
Edward	Eastman	19
Rich	Etarip	20
Jeff	Hurlburt	4
Scott A	Jelsma	15
Seymour	Joseph	15
	Krakowicz	16
M.M.	McFadden	6
Stephen	Rich	14
	Unknown	21
Don	Westcott	16

Apple Most Wanted

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

IBM Most Wanted

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

#79• The Product Monitor• *Bitkeys:* Kabul Spy• *Softkeys:* ABM• Algebra 1-6• Cause and Effect• Chemistry: Series I• Computer Generated Mathematics Vol. 2• Cribbage• Designer Puzzles• Dungeon Master Assistant Vol. 2• Economics• Genesis• Gin King• Go• Graphmaster• Hard Hat Mack• Hi Res Computer Golf• Integer Arcade• Laser Bounce• Mammals Reptiles and Insects• Master Grades• Mickey's Crossword Puzzle Maker• Mind Benders• Missing Links• Non-Western Cultures• RoboCOP• Safari Search• SAT Score Improvement Series• Special Product and Algebraic Factors• Stickybear GS Talking series Talking Alphabet• Talking Opposites• Talking Shapes• Task Force• Teacher's Toolkit version 3.1• The Great Knowledge Race• The History of Europe• The Solar System• The Time Tunnel• Thief• TrianGO• US History• Wasteland• Water and Weather• Who Am I?• Word Problems for Algebra• Worksheet Generator• Writing Chemical Formulas• Your Body• Your Body: Series II• *Playing Tips:* Baneful Tales• Elite• *Mac Features:* Mac Hard Disk Ejection Fix• *Mac Softkeys and other Patches:* ABCBase• Animation Toolkit1• Aztec C 1.0• Aztec C version 1.00c• Championship Boxing• Chart• Checkminder• Cutthroats• Cutthroats alternate• Deja Vu• Desk Toppers• Dollars & Sense• Dollars & Sense alternate• Electric Checkbook• Excel• Excel alternate fix• Fact Finder 1.0• Factfinder• Farenheit 451• Feathers & Space• File• FileMaker• Filevision• Filevision alternate• Forecast• Frogger• FunPak• Gato• Grid Wars• Griffin Terminal• Haba-Comm• Haba-Comm alternate• HabaCheckMinder• Habadex 1.1• Harrier Strike Mission• Hayden Speller• Hayden Speller alternate• Hippo^C Level 1• Hitchhiker's alternate• Hitchhiker's Guide to the Galaxy• Home Accountant• Legacy• Lode Runner• Mac Fortran• Macattack• MacChkrs/Rvrsi• MacCommand• MacDraft 1.0• MacDraft 1.1• MacGammon/ Cribbage• MacJack/Poker II• MacLabeller• MacMatch• MacPascal (version 1.0)• MacPoker• MacType• Master Type• Master Type alternate• Mouse Stampede• Multiplan alternate• Multiplan version 1.02• OverVue• PageMaker• PageMaker 1.0• Pensate• PFS• PFS File/ Report• PFS version A.03• Real Poker• Rogue• Sargon III• SkyFox• Smooth Talker• The Quest• Think Tank• ThinkTank 1.1• ThinkTank 128• ThinkTank 512• Transylvania• Triple Play 1.0• Trivia Arcade• Trivia Fever• Typing Intrigue• Ultima][• Ultima III• VideoWorks 1.0• WellTris• Winter Games• Xyphus• *Features, Notes & such:* COPYA-able Questron II• How to make Thief into a BRUNable file• How to run Task Force on your hard drive• Making Genesis into a single BRUNable file• Making Hard Hat Mack into a single BRUNable file• Making PLATO software run on the Enhanced //e• Multi-Column Print Utility (MCP)• Notes on Battle Chess• Notes on Silent Service GS• Notes on Wildcard II card• Object Module Format (OMF)• ORCA/Disassembler Scripts• ORCA/Disassembler utilities• Other Notes• Running Teacher's Toolkit v3.1 (3.5") on a Laser 128• Task Force on a hard drive and Wings by Vitesse• The Basics of Kracking (part 5): Deprotection of Modified DOS disks• The Basics of Kracking Part 6: Mating Zone & Nibblizing Mysteries• Update on the Silent Service GS v925.01 crack• Xternal Commands for BASIC: CWD (Change Working Directory)• ONLINE• **#80**• The Product Monitor• *Features, Notes & such:* Add Copy II Plus file handling to your BASIC program• Comments on the Beginner's Book• Formatting 720K disks as 1.44M HD• How to SAVE hexdumps as CDA's• Logging ProDOS Drives• The Basics of Kracking (part 7)• The Basics of Kracking (part 8)• *Bitkeys:* Black Magic• Guild of Thieves• Gunslinger• King's Quest Series• Leisure Suit Larry• Man Hunter: New York• Police Quest• Realms of Darkness• Saracen• Sierra Boot Disks• Silicon Dreams• Space Quest Series• Ultima V• Wizardry Series• Xyphus• *Softkeys:* Ancient Art of War• Battle Chess• Bridge 6.0• Captain Blood GS• Dinosaur Days v1.0• Empire• Fahrenheit 451• Fay's Word Rally• GATO v1.3• Greeting Card Maker• Hostage• Keef The Thief• Magic Spells v2.0• Magic Spells v2.1• Mickey's Crossword Puzzle Maker• Monsters and Make Believe v1.1• Pipe Dream• Pipe Dreams• Rear Guard• Rendezvous with Rama• Same or Different• Teacher's Tool Kit• Teacher's Tool Kit (IIc)• War of the Lance• Where in the USA is Carmen Sandiego?• Windwalker GS• Windwalker IIe• *APTs:* Space Rogue• Wizardry III• *Playing Tips:* Countdown• Space Rogue• *IBM Softkeys:* Serve and Volley• Welltris
#81• The Product Monitor• *Bitkeys:* Micro Typewriter• *Softkeys:* Backyard Birds• Balance of Power• Chemistry: Balancing Equations• Chemistry: The Periodic Table• Chuck Yeager's AFT• Equation Math• Estimation: Quick Solve I• Estimation: Quick Solve II• Five-Star Forecast• Fossil Hunter• Grammar Toy Shop• Instant Survey• Micro Typewriter v4.0• Murphy's Minerals• Patterns• Picture Chompers• Probability Lab• Professor Al's Sequencing Lab• Stickybear Shapes (ProDOS 1.5)• Studymate (the grade booster)• Sun and Seasons• The Duel: Test Drive II• Time Navigator• Tomahawk• Windwalker• *APTs:* Where in Europe is Carmen Sandiego?• Where in the USA is Carmen Sandiego?• Where in the World is Carmen Sandiego?• Where in Time is Carmen Sandiego?• *Playing Tips:* Windwalker• *IBM Softkeys:* Crime Wave• Gauntlet II• Stunt Driver• Thexder II• Wing Commander• *IBM Reader Review:* Copyright•
and much more...

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

Special Software Sale

(while they last)

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

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

SubLogic

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

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

SoundQuest CZ Master

Sound Quest In

(Commodore Amiga) **$10.00**

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

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

Issue	Mag	Disk	Issue	Mag	Disk	Issue	Mag	Disk	Issue	Mag	Disk
Core1	☐	☐	22	☐	☐	46	☐	☐	70	☐	☐
1	☐		23	○	☐	47	☐	☐	71	☐	☐
2	○		24	☐	☐	48	☐	☐	72	☐	☐
Core2	☐	☐	25	☐	☐	49	☐	☐	73	☐	☐
3	○		26	☐	☐	50	☐	☐	74	☐	☐
4	○		27	☐	☐	51	☐	☐	75	☐	☐
Core3	☐	☐	28☆	☐	☐	52	☐	☐	76	☐	☐
5	○		29	☐	☐	53	☐	☐	77	☐	☐
6	○		30	☐	☐	54	☐	☐	78	☐	☐
7	○	☐	31	☐	☐	55	☐	☐	79	☐	☐
8	○		32	☐	☐	56	☐	☐	80	☐	☐
9	○	☐	33	☐	☐	57	☐	☐	81	☐	☐
10	○	☐	34	☐	☐	58	☐	☐	82	☐	☐
11	○	☐	35	☐	☐	59	☐	☐	83	☐	☐
12	○	☐	36	☐	☐	60	☐	☐	84	☐	☐
13	○	☐	37	☐	☐	61	☐	☐			
14	○	☐	38	☐	☐	62	☐	☐			
15	○	☐	39	☐	☐	63	☐	☐			
16☆	☐	☐	40	☐	☐	64	☐	☐			
17	○	☐	41	☐	☐	65	☐	☐			
18	○	☐	42	☐	☐	66✱	○	☐			
19☆	☐	☐	43	☐	☐	67	☐	☐			
20	☐	☐	44	☐	☐	68	☐	☐			
21	○	☐	45	☐	☐	69	☐	☐			

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

☆ Limited supply — first-come-first-serve basis.
○ Out-of-print — only "Zeroxed" copies for sale.
✱ Issue 66 is laser printed on $8^1/_2$ by 11 paper.

Back Issue Order Form

Back Issue and Library Disk Rates

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

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

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

Shipping is included in all the prices shown.

What's a library disk?

A library disk is a $5^1/_4$ inch floppy diskette that contains programs that would normally have to be typed in by the user. Documentation for each library disk can be found in the corresponding issue.

•Library disks are available for all issues of COMPUTIST.

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

______________ Number of back issues. $______________

______________ Number of Zox back issues. $______________

______________ Number of library Disks. $______________

Washington state residents add 7.8% tax $______________

Total enclosed $______________

Name ______________________________________

Address ______________________________________

____

City ______________ State ______ Zip ______________

Country ______________ Phone ______________

VISA

MC ________-________-________-________ Exp. ______

Signature ______________________________________ 85

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

COMPUTIST 33821 E Orville Road Eatonville WA 98328