$19.95

THE BEST OF HARDCORE Computing

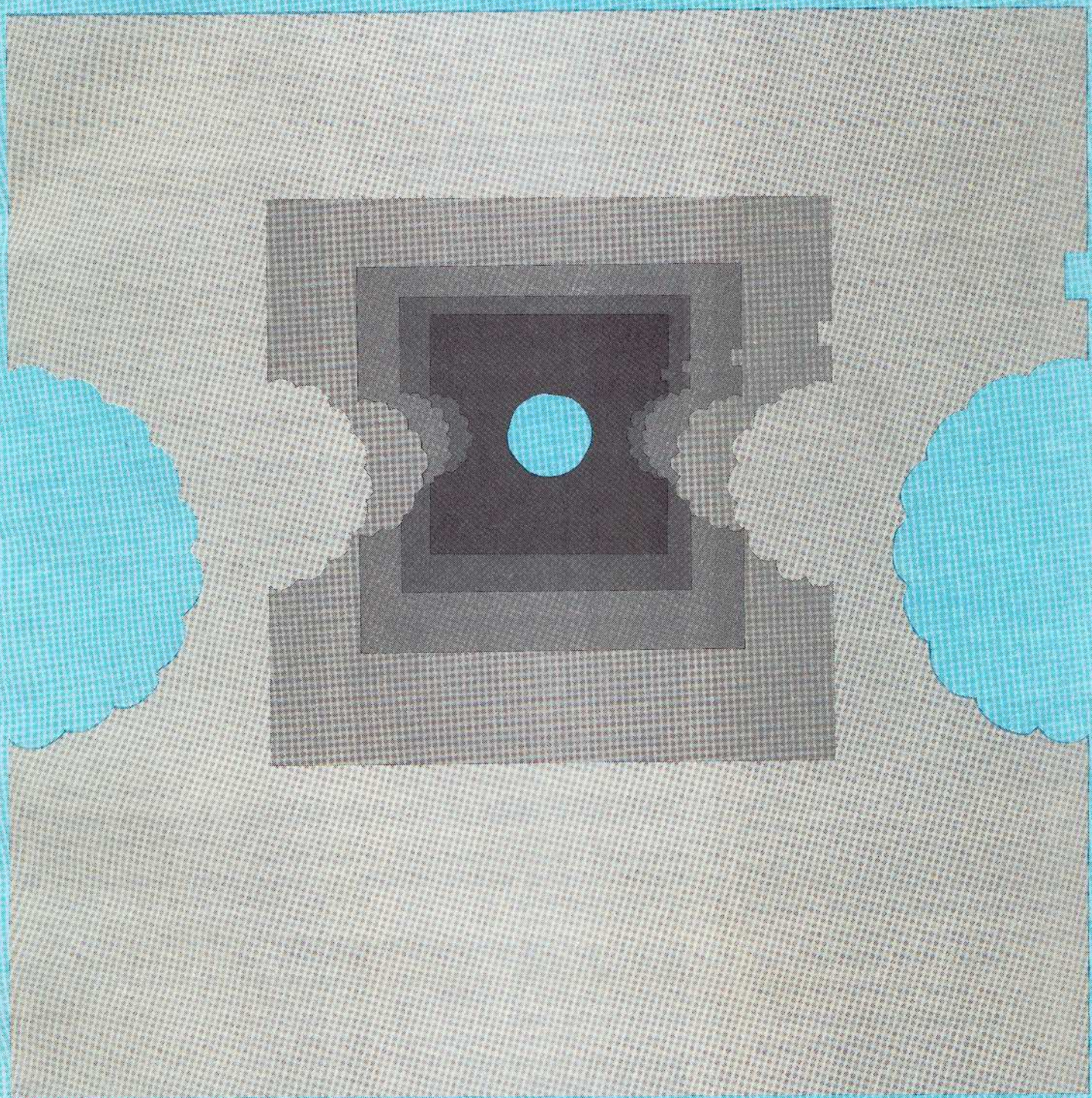


Table of Contents

The original Hardcore Computing was created when the 'right-to-copy' vs. locked software controversy was at its fanatical height. At that time the major computer magazines began a policy of information suppression by not publishing information about making back-up copies of copy-protected diskettes and refusing to publish ads that would have informed their readers about products that would make those copies.

Hardcore published not only the ads that were 'blacklisted' but articles and how-to information on making back-up copies as well as step-by-step instructions on how to de-protect (softkey) various locked-up software so that even Apple's Copy-A program would make back-up copies. In time the fanaticism waned and now most computer users know well the wisdom of making back-up copies of such 'locked-up' software.

If you took all the old Hardcores, tore off the fancy covers, deleted all the editorial material, out-of-date interviews and letters, updated the remaining material, and then included the most recent and most complete list of parameters for the major bit-copy programs... and packed it all into a single volume... you'd have the core of Hardcore Computing. We call it: The Best Of Hardcore Computing.

'The Best of...' volume is not merely a reprint of old data. Everything has been updated or re-written and consolidated, improved, concentrated. 'The Best of...' is not just a collection of unrelated articles. From the first article and program to the last, you'll discover the soul of your Apple. DiskEdit and DiskView starts you on a tutorial of disk formatting, ending in 'parms' for the four leading bit-copy programs as well as Super IOB, Hardcore's own de-protecting and copy program. Then when all the serious stuff is over, fun and games begin with Text Invaders and Zyphyr Wars.

Here's a helpful hint if you're planning to type in any of the listings in this volume. First type in Checksoft and Checkbin (see the back of this volume) so that you can use the checksums printed with each listing to insure that your typed version is correct.

A Fix For DiskEdit

(from the Best of Hardcore Computing)

Like the other pitied people who actually typed the huge hexdump for DiskEdit starting on page 10 of The Best Of Hardcore Computing, I was very disappointed when all I got was some dumb program which said something about AceWriter.

This forced me to purchase The Best Of Hardcore Computing library disk. On it, I noticed DiskEdit was correct. Using the monitors "V" command, I discovered that the Hexdump printed in The Best Of Hardcore Computing is incorrect from the first byte all the way to location $B07.

Following is the correct Hexdump listing up to $B07. Since the change in the first part of the hexdump changes the checksums for the rest of it, after location $B07, the new checksums are listed.

```
0800: 00 11 08 00 00 8C 32 30   $68DE
0808: 36 37 3A AB 31 30 3A B2   $2B6A
0810: 00 00 00 4C 73 08 01 60   $D0FC
0818: 01 00 00 00 27 08 00 09   $1EE0
0820: 00 00 01 00 00 60 01 00   $DE04
0828: 01 EF D8 00 00 00 00 00   $43A8
0830: 01 01 00 00 00 01 00 09   $6EDD
0838: 00 00 8B 0C 00 00 00 4C   $792A
0840: 90 08 4C 29 0A 4C 0E 12   $62DF
0848: 4C 7E 0F 4C 00 0A 4C 6B   $E0AD

0850: 0B 4C C9 0C 4C BD 0C 4C   $4637
0858: 5D 0E 4C AE 0B 4C 03 12   $8C0F
0860: 4C DA 11 4C 0A 0E 60 60   $457A
0868: 60 00 FF 01 00 00 00 01   $1CD9
0870: 10 23 00 20 E3 03 84 48   $8710
0878: 85 49 A0 01 B1 48 8D 17   $9AD8
0880: 08 C8 B1 48 8D 18 08 A9   $2778
0888: 1F 85 67 A9 12 85 68 60   $896E
0890: A9 08 A0 16 20 D9 03 90   $49B9
0898: 06 AD 23 08 8D 2E 08 60   $AAF8

08A0: 00 00 00 00 00 00 00 00   $8A78
08A8: 00 00 00 00 00 00 00 00   $AAF8
08B0: 00 00 00 00 00 00 00 00   $8A78
08B8: 00 00 00 00 00 00 00 00   $AAF8
08C0: 00 00 00 00 00 00 00 00   $8A78
08C8: 00 00 00 00 00 00 00 00   $AAF8
08D0: 00 00 00 00 00 00 00 00   $8A78
08D8: 00 00 00 00 00 00 00 00   $AAF8
08E0: 00 00 00 00 00 00 00 00   $8A78
08E8: 00 00 00 00 00 00 00 00   $AAF8

08F0: 00 00 00 00 00 00 00 00   $8A78
08F8: 00 00 00 00 00 00 00 00   $AAF8
0900: 00 00 00 00 00 00 00 00   $8A78
0908: 00 00 00 00 00 00 00 00   $AAF8
0910: 00 00 00 00 00 00 00 00   $8A78
0918: 00 00 00 00 00 00 00 00   $AAF8
0920: 00 00 00 00 00 00 00 00   $8A78
0928: 00 00 00 00 00 00 00 00   $AAF8
0930: 00 00 00 00 00 00 00 00   $8A78
0938: 00 00 00 00 00 00 00 00   $AAF8
0940: 00 00 00 00 00 00 00 00   $8A78
0948: 00 00 00 00 00 00 00 00   $AAF8
0950: 00 00 00 00 00 00 00 00   $8A78
0958: 00 00 00 00 00 00 00 00   $AAF8
0960: 00 00 00 00 00 00 00 00   $8A78
0968: 00 00 00 00 00 00 00 00   $AAF8
0970: 00 00 00 00 00 00 00 00   $8A78
0978: 00 00 00 00 00 00 00 00   $AAF8
0980: 00 00 00 00 00 00 00 00   $8A78
0988: 00 00 00 00 00 00 00 00   $AAF8

0990: 00 00 00 00 00 00 00 00   $8A78
0998: 00 00 00 00 00 00 00 00   $AAF8
09A0: 00 00 00 00 00 00 00 00   $8A78
09A8: 00 00 00 00 00 00 00 00   $AAF8
09B0: 00 00 00 00 00 00 00 00   $8A78
09B8: 00 00 00 00 00 00 00 00   $AAF8
09C0: 00 00 00 00 00 00 00 00   $8A78
09C8: 00 00 00 00 00 00 00 00   $AAF8
09D0: 00 00 00 00 00 00 00 00   $8A78
09D8: 00 00 00 00 00 00 00 00   $AAF8

09E0: 00 00 00 00 00 00 00 00   $8A78
09E8: 00 00 00 00 00 00 00 00   $AAF8
09F0: 00 00 00 00 00 00 00 00   $8A78
09F8: 00 00 00 00 00 00 00 00   $AAF8
0A00: A9 8D 20 ED FD A5 3A 8D   $79DA
0A08: 2B 08 20 0E 12 A9 04 85   $45AC
0A10: 24 A9 AD 20 ED FD A2 01   $2560
0A18: 20 4A F9 20 8C F8 20 D3   $461D
0A20: F8 20 53 F9 85 3A 84 3B   $D75B
0A28: 60 AD 35 08 0A 0A AA A0   $DEC9

0A30: 00 BD 95 0C 99 E7 00 E8   $2A16
0A38: C8 C0 04 90 F4 A9 00 85   $0EAE
0A40: E4 85 25 20 CF 0A E6 25   $6CF7
0A48: A5 25 C9 14 D0 F5 A5 25   $FD3E
0A50: 20 8F 0A A0 27 A9 A0 91   $5AC2
0A58: 28 88 10 FB 4C C9 0C 00   $0CED
0A60: 04 80 04 00 05 80 05 00   $6629
0A68: 06 80 06 00 07 80 07 28   $1411
0A70: 04 A8 04 28 05 A8 05 28   $DE15
0A78: 06 A8 06 28 07 A8 07 50   $5CC5

0A80: 04 D0 04 50 05 D0 05 50   $7681
0A88: 06 D0 06 50 07 D0 07 0A   $3600
0A90: AA BD 5F 0A 85 28 18 69   $C20B
0A98: 1B 85 26 BD 60 0A 85 29   $9DE2
0AA0: 85 27 60 A2 14 CA BD B8   $3A77
0AA8: 0A CD 34 08 90 05 F0 03   $0A7A
0AB0: 4C A5 0A 86 25 85 E4 60   $F356
0AB8: 00 0D 1A 27 34 41 4E 5B   $AB3E
0AC0: 68 75 82 8F 9C A9 B6 C3   $53C6
0AC8: D0 DD EA F7 20 A3 0A A5   $9520

0AD0: 25 20 8F 0A A9 0D 8D 6C   $81D9
0AD8: 08 A6 E4 BD 00 09 48 EC   $5368
0AE0: 34 08 D0 05 AE 6F 08 F0   $02FF
0AE8: 03 20 B1 0B A0 00 91 26   $EAE0
0AF0: E6 26 68 AE 6F 08 F0 44   $9C6B
0AF8: 48 4A 4A 4A 4A A6 E4 EC   $C880
0B00: 34 08 D0 0B 09 30 C9 3A   $75D4
```

```
0B08-$DE13
0B10-$C4B7
0B18-$A6E0
0B20-$9975
0B28-$0EFE
0B30-$B69C
0B38-$D12D
0B40-$859D
0B48-$D64D
0B50-$28EE
0B58-$AC00
0B60-$BE55
0B68-$3987
0B70-$12C9
0B78-$FE56
0B80-$2822
0B88-$42A3
0B90-$586C
0B98-$7764
0BA0-$963A
0BA8-$5151
0BB0-$3CB6
0BB8-$3097
0BC0-$316B
0BC8-$11FA
0BD0-$6398
0BD8-$F2D1
0BE0-$6366
0BE8-$629D
0BF0-$420D
0BF8-$C23D
0C00-$29E3
0C08-$DBF5
0C10-$A1FF
0C18-$018F
0C20-$719F
0C28-$D1CF
0C30-$017F
0C38-$D1CF
0C40-$017F
0C48-$D1CF
0C50-$C15F
0C58-$214F
0C60-$F1FF
0C68-$214F
0C70-$F1FF
0C78-$214F
0C80-$F1FF
0C88-$214F
0C90-$69BC
0C98-$5D58
0CA0-$5E21
0CA8-$9E99
0CB0-$1EE1
0CB8-$A5BF
0CC0-$EF0C
0CC8-$F9E7
0CD0-$B3C3
0CD8-$FBA0
0CE0-$CA42
0CE8-$9FBA
0CF0-$31B5
0CF8-$F436
0D00-$DCE4
0D08-$17B8
0D10-$DA82
0D18-$FB3B
0D20-$BD6A
0D28-$2AD0
0D30-$ACD9
0D38-$D60D
0D40-$D3FB
0D48-$A45C
0D50-$E1BE
0D58-$9F9A
0D60-$DD57
0D68-$A0BA
```

```
0D70-$0C3B
0D78-$3F3F
0D80-$1E61
0D88-$DA03
0D90-$D5EB
0D98-$B658
0DA0-$E4F0
0DA8-$041F
0DB0-$8EDB
0DB8-$0FA2
0DC0-$03D7
0DC8-$B2D8
0DD0-$F4C5
0DD8-$B40A
0DE0-$95EA
0DE8-$2151
0DF0-$0719
0DF8-$88C7
0E00-$AED5
0E08-$1EBD
0E10-$E3CE
0E18-$0A9A
0E20-$4EC5
0E28-$E4DB
0E30-$1CEF
0E38-$56D8
0E40-$4FEB
0E48-$EA28
0E50-$B9AC
0E58-$DE6F
0E60-$95F4
0E68-$7786
0E70-$B198
0E78-$5E7D
0E80-$25B5
0E88-$3225
0E90-$7455
0E98-$A66A
0EA0-$47BE
0EA8-$47B3
0EB0-$C440
0EB8-$CB60
0EC0-$C37F
0EC8-$430A
0ED0-$7F8D
0ED8-$473E
0EE0-$73F5
0EE8-$B11E
0EF0-$043C
0EF8-$4443
0F00-$2535
0F08-$213C
0F10-$4A53
0F18-$E450
0F20-$2479
0F28-$EA36
0F30-$6648
0F38-$B9BE
0F40-$9DA3
0F48-$8568
0F50-$5F35
0F58-$5B3B
0F60-$67F7
0F68-$1665
0F70-$0541
0F78-$D3E9
0F80-$C4F4
0F88-$07BE
0F90-$F3F2
0F98-$102A
0FA0-$B90C
0FA8-$D268
0FB0-$7DF8
0FB8-$2D7E
0FC0-$4329
0FC8-$E655
0FD0-$71DB
0FD8-$8960
```

```
0FE0-$70E9
0FE8-$A984
0FF0-$8502
0FF8-$E767
1000-$FB2F
1008-$100C
1010-$4696
1018-$2B9D
1020-$A2E6
1028-$9049
1030-$CD28
1038-$2ADE
1040-$4567
1048-$6286
1050-$A4DE
1058-$8F0F
1060-$CE90
1068-$3C8F
1070-$FF81
1078-$6C3E
1080-$EFA6
1088-$72A3
1090-$3128
1098-$9ED4
10A0-$E6A9
10A8-$5216
10B0-$9FDE
10B8-$9103
10C0-$3359
10C8-$59A7
10D0-$AA0B
10D8-$5294
10E0-$FC00
10E8-$ABF8
10F0-$D2ED
10F8-$CEEF
1100-$A28C
1108-$FAB1
1110-$292E
1118-$0904
1120-$89BA
1128-$AB23
1130-$AFAB
1138-$39F3
1140-$AE90
1148-$9BD1
1150-$5211
1158-$760E
1160-$33CA
1168-$A35E
1170-$F0B8
1178-$F126
1180-$7899
1188-$D7A2
1190-$26EA
1198-$EDA3
11A0-$8F2E
11A8-$D263
11B0-$6D6D
11B8-$6FC1
11C0-$9FE5
11C8-$1D96
11D0-$B5B4
11D8-$EE35
11E0-$B602
11E8-$F0D6
11F0-$ECA7
11F8-$6E6C
1200-$B9A3
1208-$ED42
1210-$0FAE
1218-$0313
1220-$1583
```

Getting into DOS with

DiskEdit

By Charles Haight

Certain tools are required to understand DOS and to manipulate disk files. The first is a nibbler or bit editor. The second and most important of these is a sector editor.

DiskEdit is one such utility.

DiskEdit is a user oriented direct disk access program. Simply stated, DiskEdit allows the user to read or write any sector on a disk. This means that the user can:

Directly edit files on disk.
Change text in binary files.
Insert illegal characters in REMs.
Directly alter data base files.
Move sectors (even between disks).
Repair crashed disks.
Format catalog names.
Remove illegal codes in file names.
Write flashing and inverse titles.
Repair the VTOC.
UnDELETE deleted files.
Hide file names.

DiskEdit will display an entire sector as hexadecimal and ASCII.

The keyword in DiskEdit is simplicity. The commands are single key entry (you don't have to keep hitting return). With DiskEdit you can directly enter control, inverse, flashing and lower case characters. Input and display information can be in hex or decimal. The shimmering cursor is easy to identify even with a screen full of inverse and flashing characters. You can jump the cursor to any absolute position within a sector. The NEXT and LAST commands allow you to single-step through track/sectors. And DiskEdit has a simple escape. If you change your mind, pressing the escape key will set the defaults and return you to the command mode.

Disk Overview

Before we begin entering DiskEdit, let's take a closer look at DOS and a normal disk.

The flexible (or floppy) diskette can be thought of as a disc-shaped piece of recording tape, and essentially that's all it is. A flat disk shape is used, instead of a flat strip (as in a tape), in order to maximize the rate of data transfer. For instance, to transfer data to and from a tape, the computer would have to READ all of the tape preceding the area where the data was stored before it could transfer the required data. This method of information retrieval is known as "sequential access" and is about the same as scanning a cassette tape for a favorite song.

The disk, on the other hand, is set up in such a way that the computer can go directly to a piece of data or program by scanning the disk laterally. This method of information retrieval is known as "random access" and is similiar to selecting a particular song on a record.

Before a disk can be used, it must be formatted. The INIT command is used for this purpose.

When a disk is initialized, the Disk Operating System (DOS) writes 35 concentric tracks. Each track is divided into 16 blocks called "sectors". (DOS version 3.2 writes only 13 "sectors".) Each sector contains an address mark and a data mark. These marks start and end with a unique pattern of bytes.

The address mark tells the DOS what track/sector it is currently reading. It contains the volume, track, sector and checksum information. The data mark contains the actual data. It tells the DOS where the data begins and ends and includes a checksum that is used to verify the accuracy of the data.

If you have ever tried to load a program and the disk drive started making a slight chatter, chances are that the DOS could not read one of these markers. It then recalibrates the read/write head by moving it back to track zero and stepping (counting each track that it passes over) back out to where it was supposed to be.

The tracks are numbered from $00 (0) to $22 (34) and the sectors from $00 (0) to $0F (15). Tracks $00 through track $02 (a total of three tracks; zero, one and two) contain the DOS program.

The DOS gives the Apple the ability to manipulate data on a diskette. In this program are all of the commands related to controlling the disk drive (i.e. CATALOG, INIT, LOAD...) and a set of ERROR messages which, unless you either are a magician or don't use the Disk II, you have probably seen before.

The disk controller card that connects the Disk II to the Apple also has a small program on it. When you boot a disk, this program tells the Disk II to read track $00 (0), sector $00 (0) (remember, we start counting at zero instead of one) into memory.

The program on track $00, sector $00 contains the information required to read in sectors $00 through $09 on track $00. The program on sectors $00-$09 reads in the remaining information on track $00-$02. When this process is completed, the entire operating system (DOS) will be in memory.

At this point, DOS takes over and runs the "HELLO" program. The program that was used to initialize a disk is usually refered to as the hello or greeting program.

In order to find your "HELLO" program, DOS goes to the Volume Table of Contents (VTOC) and Directory located on track $11 (17). The VTOC and Directory are used by DOS whenever you read or write to the disk. The VTOC or "bit map" shows which sectors are in use and which are free. The second and third byte of the VTOC point to where the directory starts.

The Directory begins on sector $0F (15) and continues down to sector $01 (1). The second and third byte of each directory sector point to the next available sector. If these two bytes are zero, then there are no more sectors. The Directory contains a list of all the files on the disk. Each entry contains a pointer to the track/sector list, a file status (locked/unlocked) code, a file type code (1 letter), the file name (30 characters) and the file size. The track/sector list is a list of track/sector pairs that are used to store that program. This is why saving a blank file always takes two sectors. One for the blank file and one for the track/sector list.

DOS will read the VTOC which will point to the directory. DOS then finds the program name in the directory and finds where the track/sector list is. DOS then loads all of the track/sector pairs into the proper memory locations. Finally, DOS transfers control to the resident BASIC (Applesoft?) which will run the program.

Entering the Program

Enter the machine code portion of DiskEdit first. Save it to disk as ED.OBJ.

BSAVE ED.OBJ, A$800, L$A21

Enter the BASIC listing and save it to disk as ED.BAS.

SAVE ED.BAS

Bload the binary file.

BLOAD ED.OBJ

Type "RUN" and press return. After the "?UNDEF'D STATEMENT ERROR"

message, run ED.BAS.

RUN ED.BAS

This will combine the two programs to form DiskEdit.

Type 'X' to exit to BASIC. Now, insert a blank disk in the drive and type 'INIT DISK EDIT'. Use this back up copy for the following examples and ALL other uses.

Getting Familiar

This exercise will aid you in understanding how to use the commands by taking you on a tour of a normal DOS diskette. Please read each paragraph before pressing any keys and follow the directions carefully.

Insert the DiskEdit back-up disk in Drive 1. Turn on your computer. DiskEdit will prompt you when it is ready.

Press any key to start.

What is your status?

On the bottom of the screen are the status indicators and prompts. They tell you the slot (SL), drive (DR), track (T), sector (S), volume (V), byte position (B), filter (F) and data entry mode currently selected.

Reading

Press the 'R' key. This tells DiskEdit that you want to READ a sector from the disk. A flashing prompt will appear next to the track (T) indicator. DiskEdit is asking you what track to read.

Type '01'. This tells DiskEdit that you wish to read track $01 (1). The flashing prompt will move over to the sector (S) indicator. Respond to this prompt by typing '8'.

The disk drive should whirr for about two seconds, and then stop. The screen should be full of numbers and letters. You are now looking at the contents of track $01 (1), sector $08 (8) in what is known as hex or hexadecimal format on the left side of your screen and ASCII on the right side.

Hex a what?

Hexadecimal is a base sixteen numbering system. It gets its name from the fact that it contains all of the numbers found in normal base 10 (decimal 0-9) plus six alphabetic characters (A thru F).

Say 'AS-KEY'

ASCII stands for "American Standard Code for Information Interchange." This is the alphanumeric equivalent of all of those hex symbols on the right.

Error messages

The sector you are now viewing ($08) contains the DOS error messages (they are continued on sector $09).

Press the 'N' key. This will increment the sector count and cause Diskedit to read the next sector. If the sector count had been at $0F (15), the track count would have been incremented by one and the sector count reset to $00 (0).

The "Boot" Program

You are now viewing the sector where the "Boot" program name is stored. In the center of the screen is the file name 'DISK EDIT'. This is the name of the program that the DOS will automatically 'RUN' when this disk is booted. (If you decide later to change the boot program name on this disk, this is where you should come.)

Let's follow how DOS located the file "DISK EDIT" when you booted this disk.

Press 'R' to read. Type '11' for the track and '0' for the sector.

You are looking at the VTOC or bit map. The second and third byte point to the first directory (catalog) sector. These bytes should be '11 0F'.

Press 'R' and type '11' for the track and 'F' for the sector.

The sector you are viewing is the first part of the directory, which extends downward to sector $01 (1). Press the zero key. This is a special function key designed to make viewing catalog sectors more meaningful. The screen will return to normal when you press any other key.

Moving the cursor

The I, J, K and M keys are the cursor movement keys. The cursor has a wrap around feature. If you go off the screen on one side, you will come back on the opposite side.

Press the 'O' key. The flashing prompt will appear next to the byte position (B) indicator.

This command allows us to move the cursor to a specific location on the screen. Move the cursor to the beginning of the file name by typing '0E'. The cursor should now be in front of the 'D' of "DISK EDIT".

Move the cursor back one character by pressing 'J'. Look at the hex portion of your screen. The '02' is used by DOS to tell what type of program DiskEdit is and whether it is locked or unlocked. The '0' means that the file is unlocked. The '2' means the file is Applesoft.

Editing

Press the 'E' key. This tells DiskEdit that you wish to edit the sector.

Type '82'. Press 'ESC' to exit the EDIT mode. Press the 'O' key. Type '2C'. The byte you are looking at and the '00' following it are the hex equivalent of the sector use count for the file. Press the 'E' key. Type '00'. Press 'ESC' to exit the EDIT mode.

Press the zero key.

The program HELLO is shown with an asterisk. Changing the '02' into a '82' locked the file. Entering the '00' will change the sector count for the file to zero.

Writing

WARNING: Read the following paragraph completely before you press any keys.

Up to this point, you have only been editing the disk information that is in the computer's memory. In order to make the changes permanent you need to WRITE this information back to the disk.

The command to do this is 'W' for WRITE. Press the 'W' key. Press 'RETURN' for the track (T) and sector (S).

When the RETURN key is pressed in response to a prompt the program will act as if the default values were entered. The default values for the track and sector are the last track/sector that was read or written.

The program will beep and a warning will be printed. This is your last chance to change your mind. You must press RETURN to have DiskEdit write to your disk. Any other key will abort this operation.

Press RETURN. The buffer contents are now written to the disk. Press the 'C' key to see the catalog. The first file will be locked (indicated by the asterisk '*' next to the file type) and the sector count will be '000'. Press any key to continue.

This completes the exercise. Experiment with DiskEdit using this same scratch diskette.

Summary of Commands

ESC This is the "I changed my mind" key. Press this key to reset defaults and exit back to the command mode.

RTN The RETURN key, when used to answer an input prompt, will accept the current default and continue. (Example: When prompted for the track and sector during a read command, pressing RETURN twice will cause the current track and sector to be read.)

> Track skip command. Increments the track number and performs a READ. Does not increment the sector.

< Track skip command. Decrements the track number and performs a READ. Does not decrement the sector number.

A Sets character entry mode to ASCII

B Disassemble buffer command. Calls the monitor to disassemble buffer contents starting at the cursor location. Use the space bar to continue disassembly one line at a time or press RETURN to disassemble 20 additional lines. Press 'P' to print the screen display. (Press ESC to exit.)

C Displays the disk catalog using the current slot and drive. Prints the number of free sectors on the disk.

D Flips the active drive from 1 to 2 or from 2 to 1 on each keypress.

E A continuous-edit mode, this mode allows you to type changes just like on a typewriter. Pure cursor movement is supported using control keys. If you are in

hexadecimal format, only valid hex digits are accepted as input. In ASCII format all keys are valid except the control keys listed below. (Press ESC to exit.)

Ctrl Key	Function
F	set FLASH mode
I	set INVERSE mode
N	set NORMAL mode
Q	move cursor up
Z	move cursor down
→	move cursor right
←	move cursor left

+ This edit submode is entered using the plus (+) key. The '>>EDIT<<' prompt is changed to '++EDIT++'. It is identical to the normal edit mode except that it does not support control functions. All keys are valid except ESC. Control characters may be directly entered. The plus (+) key or the semi-colon (;) may be used to enter this submode.

F This is the filter format command it allows you to change the filter values so that you can configure your own filters.

G Turns the sound on or off each time you press the 'G' key. (Default at BOOT is on.)

H Sets character entry mode to Hexadecimal

I Moves cursor up.

J Moves cursor left.

K Moves cursor right.

M Moves cursor down.

L Reads last sector.

N Reads next sector.

O Allows cursor to be jumped to any absolute position in the displayed sector.

P Sends the buffer contents to your printer. A header is printed first which shows the track, sector, and volume. When first used, the program will ask which slot your printer is using and whether you wish to use 40 or 80 columns.

R Prompts you for the track and sector to read. Use the RETURN key to accept default values.

S Prompts you for a new slot. Valid entries are from 1 to 7.

U Toggles the status indicators between hex and decimal and updates the display information. Only the track, sector, and cursor are affected by this key. (Default at BOOT is hex.)

W Prompts you for the track and sector to write to. Use the RETURN key to accept default values. After entering the track and sector, DiskEdit will beep and pause. This is your last chance to change your mind. Press RETURN to WRITE, or any other key to escape.

X Clears the screen and exits to BASIC.

ASCII Filters

The number following the filter (F) indicator is the filter currently selected.

There are 9 filters. Each affects the format of the displayed screen contents. They do not change the actual buffer contents in any way. They may be selected by pressing the corresponding number (1-9) key.

Rolling your own

The filters can be modified from the keyboard. Select a filter (1-9) by pressing the appropriate number key. Press the 'F' key.

The 256 screen characters are divided into 8 blocks. The prompt under 'BLOCK' indicates the original group of characters while the prompt under 'CHG:' indicates what characters will be displayed on the screen.

The first prompt is 'INV1' for inverse letters. Press '7'. This causes all inverse characters in block 1 to display as normal. Block 7 is normal letters. The 'INV1' prompt under 'CHG:' will change to 'NOR2'. By pressing a number from 1 to 8, each of the original blocks can be changed to display as any other block. Pressing 'RETURN' will skip a block.

Next to 'CHG:' is 'FN#'. The "FN#' is short for function number. There are 3 functions.

1. Print block, delete one character
2. Delete block, print one character
3. Delete entire block

Customizing the Program

DiskEdit is an Applesoft program with packed machine code. This means that the machine code portion of the program is hidden in such a way that DOS thinks it is part of the Applesoft program.

The machine code is hidden behind the REM in line 0 rather than at the end of the BASIC program. This was done in order to allow program modification while keeping the program size as small as possible.

If you load the program and list it, you will see a single BASIC line:

```
0 CALL 2167 : GOTO 10 : REM
```

In order to make changes you will need to follow these steps:

1. RUN the program.
2. When the copyright notice is on the screen, press RESET to exit the program.
3. LIST the program and make changes.
4. After making any changes, RUN the program and exit using the "X" key. This will change the zero page pointers so that DOS can save the machine code along with the modified program.
5. SAVE the modified program to disk.

DiskEdit BASIC program

```
10 TEXT: HOME: GOSUB 2150: GOTO 750
20 REM CLEAR TEXT WINDOW
30 POKE 35,21: HOME: RETURN
40 REM GET CHARACTER WITH PROMPT
50 POKE - 16368,0
60 GET N$:KY = ASC (N$) + 128: IF KY
    < > 155 THEN RETURN
70 REM RESET ALL DEFAULTS
80 POKE TR,TS: POKE SC,SS: POKE
    CM,RD:TK = TS:SE = SS: CALL TT:
    CALL MV
90 REM CLEAR STACK, GOTO CMD PARSER
100 CALL - 10621: GOTO 750
110 REM MAKE NOISE AND RETURN
120 PRINT G$G$;: RETURN
130 REM FIND BINARY START
140 IF PEEK (1024) = 164 THEN 190
150 REM FOR DECIMAL NUMBER
160 A1 = PEEK (1024) - 176:A2 = PEEK
    (1025) - 176: IF A2 > - 1 THEN
    GOSUB 400:A1 = KY:A2 = PEEK
    (1026) - 176: IFA2 > - 1 THEN
    GOSUB 400: RETURN
170 KY = A1: RETURN
180 REM FOR HEX NUMBER
190 KY = PEEK (1025): GOSUB 280:A1 =
    KY:KY = PEEK (1026): GOSUB
    280:A2 = KY:KY = A1 * 16 + A2:
    RETURN
200 REM GET KEY WITHOUT PROMPT
210 KY = PEEK ( - 16384): IF KY <128
    THEN 210
220 POKE - 16368,0: RETURN
230 REM HANDLE AN ERROR
240 A1 = PEEK (EF): GOSUB 30:
    VTAB12: HTAB 12: IF A1 = 16 THEN
    PRINT "UNABLE TO WRITE": GOTO260
250 PRINT "DISK DRIVE ERROR"
260 PRINT G$G$;: FOR X = 1 TO 1000:
    NEXT: POKE EF,0: POKE 35,24:
    CALL MV: GOTO 80
270 REM PROCESS HEX/DEC INPUT
280 KY = KY - 176: IF KY < 0 OR KY >
    22 THEN KY = 128: RETURN
290 IF KY > 9 THEN KY = KY - 7: IFKY
    < 10 OR KY > 15 THEN KY =128
300 RETURN
310 REM GET HEX OR DEC ONLY
320 GOSUB 50
330 IF KY = 141 THEN RETURN
340 GOSUB 280
350 IF KY = 128 THEN GOSUB 120: GOTO
    320
360 IF PEEK (HF) AND KY > 9 THEN
    GOSUB 120: GOTO 320
370 RETURN
380 REM CALCULATE HEX/DEC NO.
390 IF NOT PEEK (HF) THEN KY =A1 *
    16 + A2: RETURN
400 KY = A1 * 10 + A2: RETURN
410 REM GET TRACK VALUE
420 VTAB 22: HTAB 14 - PEEK (HF):
    GOSUB 320: IF KY > 15 THEN KY =
    TK: GOTO 480
430 IF NOT PEEK (HF) AND KY >2 THEN
    480
440 IF KY > 3 THEN 480
450 A1 = KY: PRINT N$;: GOSUB 320:
    IF KY > 15 THEN KY = A1: GOTO480
460 A2 = KY: GOSUB 390
470 REM CHECK FOR VALID TRACK#
480 IF KY < 0 OR KY > 34 THEN PRINT
    G$;: GOTO 420
490 REM SAVE OLD TRK#, POKE NEW
500 TS = TK:TK = KY: POKE TR,TK:
    CALLTT
510 REM GET SECTOR VALUE
520 VTAB 22: HTAB 21 - ( PEEK (HF))
    * 2: GOSUB 320: IF KY >15 THEN
    KY = SE: GOTO 620
530 REM CHECK FOR HEX I/O
540 IF NOT PEEK (HF) THEN 620
```

```
550 REM SAVE KEY
560 IF KY > 1 THEN 620
570 REM GET ANOTHER KEY
580 A1 = KY: PRINT N$;: GOSUB 320:
    IF KY > 15 THEN KY = A1: GOTO 620
590 REM CHECK FOR VALID SECTOR#
600 A2 = KY: GOSUB 390: IF KY < 0 OR
    KY > 15 THEN PRINT G$;: GOTO 520
610 REM SAVE OLD SCT#, POKE NEW
620 SS = SE:SE = KY: POKE SC,SE:
    CALLTT
630 REM IF WRITE THEN LAST CHANCE
640 IF PEEK (CM) = WR THEN VTAB24:
    HTAB 2: PRINT "PRESS RETURN TO
    ->";: FLASH : PRINT "WRITE";:
    NORMAL : PRINT "<-, ESC TO
    EXIT"G$;: NORMAL : POKE -
    16368,0: GOSUB 210: IF KY < >
    141 THEN 80
650 GOTO 710
660 REM PRINT 40 "="'S
670 FOR X = 1 TO 40: PRINT "=";:
    NEXT : RETURN
680 REM PRINT SCREEN PROMPTS
690 CALL TT
700 REM READ OR WRITE A SECTOR
710 CALL IO
720 REM PRINT BUFFER TO SCREEN
730 CALL MV: RETURN
740 REM COMMAND PARSER
750 POKE 216,0: CALL TT: VTAB 23:
    HTAB 1: CALL - 958: IF PEEK(EF)
    > 0 THEN GOSUB 240
760 REM SAVE CURRENT TRACK/SECTOR
770 TS = PEEK (TR):SS = PEEK (SC):TK
    = TS:SE = SS
780 CALL XC:KY = PEEK (225) - 192
790 IF KY = - 5 OR KY = - 21 THEN1380
800 IF KY < 0 OR KY > 26 THEN 750
810 ON KY GOSUB 100,1870,1830,100,
    1400,840,1450 ,100,100,100,100,
    100,100,100,1590,1480,100,420,
    1680,100,100,100,1720,1740,100
    ,100: GOTO 750
820 PRINT G$;: GOTO 750
830 REM *** DEFINE FILTER ***
840 TEXT : HOME : VTAB 22: HTAB7:
    PRINT "CONFIGURATION FOR FILTER
    #" PEEK (FL)
850 VTAB 2: PRINT G$"# BLOCK
      # CHG:    FN# CHR$ STATUS"
860 PRINT
870 DL = PEEK (231) + PEEK (232) *
    256 - 1:CG = PEEK (233) + PEEK
    (234) * 256 - 1
880 FI = PEEK (FL)
890 REM PRINT CURRENT VALUES
900 FOR X = 1 TO 8: PRINT X".
    "F$(X)" -> ";
910 F = PEEK (CG + X)
920 F1 = INT (F / 32) + X: IF F1 > 8
    THEN F1 = F1 - 8
930 F2 = F - ( INT (F / 32) * 32)
940 F3 = PEEK (DL + X)
950 F4 = PEEK (NO + FI)
960 F1(X) = F1:F2(X) = F2:F3(X) =F3
    + (F(F1) * (F2 < > 0)) +(F(X) *
    (F2 = 0))
970 PRINT F1". "F$(F1);: HTAB 23:
    PRINT F2;: HTAB 27: POKE
    2091,F3: CALL HP: CALL AP: IFX <
    > 1 THEN 1000
980 HTAB 36: IF F4 = 1 THEN PRINT"ON
    ";
990 IF F4 = 0 THEN PRINT "OFF";
1000 PRINT : PRINT : NEXT
1010 REM EDIT CURRENT VALUES
1020 FOR X = 1 TO 8: VTAB X * 2 +2:
    HTAB 12
1030 REM GET BLOCK #
1040 GOSUB 50:A = KY - 176: IF N$ =
    CHR$ (13) THEN A = F1(X):N$ = ""
1050 IF A < 1 OR A > 8 THEN PRINTG$;:
    GOTO 1040
1060 PRINT N$;: HTAB 15: PRINT
    F$(A);: HTAB 23
1070 C = F2(X)
1080 REM CALCULATE OFFSET
1090 IF A > = X THEN F = A - X
1100 IF A < X THEN F = (8 - X) +A
1110 POKE CG + X,F * 32 + C
1120 REM GET FUNCTION #
1130 GOSUB 50:C = KY - 176: IF N$ =
    CHR$ (13) THEN C = F2(X):N$ = ""
1140 IF C < 0 OR C > 3 THEN PRINTG$;:
    GOTO 1130
1150 PRINT N$;
1160 REM CHANGE FILTER VALUE
1170 POKE CG + X,F * 32 + C
1180 KY = F3(X): IF C = 0 THEN KY = 0
1190 IF C < 1 OR C = 3 THEN 1270
1200 VTAB 20: HTAB 1: PRINT "ENTER
    CHARACTER: ";: GOSUB 50: IFKY =
    141 THEN KY = F3(X)
1210 IF KY < 160 OR KY > 223 THEN
    PRINT G$;: GOTO 1200
1220 IF KY < 192 THEN KY = KY +(2 +
    A) * 32: GOTO 1240
1230 IF KY > 191 THEN KY = KY + (1 +
    A) * 32
1240 KY = KY - 256: HTAB 1: CALL -
    868: VTAB X * 2 + 2
1250 POKE DL + X,KY
1260 HTAB 27: POKE 2091,KY: CALLHP:
    CALL AP
1270 NEXT
1280 REM GET FILTER STATUS
1290 PRINT : PRINT : PRINT "LEAVE
    FILTER ON DURING EDIT? (Y/";:
    INVERSE : PRINT "N";: NORMAL:
    PRINT "): "G$;: GOSUB 50
1300 HTAB 1: CALL - 868: VTAB 4:
    HTAB 36: IF N$ = "Y" THENA = 1:
    PRINT "ON ";: GOTO 1320
1310 PRINT "OFF";:A = 0
1320 POKE NO + FI,A
1330 REM RESTORE SCREEN, EXIT
1340 FOR X = 1 TO 500: NEXT
1350 GOTO 730
1360 REM ++EDIT++ MODE ENTRY POINT
1370 IF FI = 0 THEN RETURN
1380 VTAB 24: HTAB 2: INVERSE :
    PRINT"++EDIT++";: POKE NC,0:
    GOTO1410
1390 REM EDIT MODE ENTRY POINT
1400 VTAB 24: HTAB 2: INVERSE :
    PRINT">>EDIT<<";: POKE NC,1
1410 NORMAL : HTAB 12: PRINT
    "MODE";:
1420 PRINT "    PRESS <ESC> TO
    EXIT";
1430 CALL ED: VTAB 23: HTAB 1: CALL
    - 958: GOTO 80
1440 REM TURN SOUND ON/OFF
1450 PRINT G$;: IF G$ = CHR$ (7)
    THEN G$ = "": RETURN
1460 G$ = CHR$ (7): RETURN
1470 REM *** PRINT HARDCOPY ***
1480 IF NOT PR THEN GOSUB 1760
1490 GOSUB 30
1500 A1 = PEEK (BF) * 256 - 1
1510 PR# PR: PRINT
1520 PRINT "TRACK: ";: POKE NM,TK:
    CALL HX: PRINT " SECTOR:";: POKE
    NM,SE: CALL HX: PRINT" VOLUME: "
    PEEK (VO)
1530 FOR X = 0 TO 255 STEP 16 /LI:
    POKE NM,X: CALL HX: HTAB5: PRINT
    "-";
1540 FOR A = 1 TO 16 / LI: POKE2091,
    PEEK (A1 + X + A): CALLHP: NEXT
1550 FOR A = 1 TO 16 / LI: POKE2091,
    PEEK (A1 + X + A): CALLAP: NEXT
1560 PRINT : NEXT
1570 PR# 0: GOTO 80
1580 REM *** JUMP CURSOR ***
1590 VTAB 22: HTAB 32 - PEEK (HF):
    GOSUB 320: IF KY > 15 THEN CALL
    TT: RETURN
1600 A1 = KY: PRINT N$;: GOSUB 320:
    IF KY > 15 THEN KY = A1:
    GOTO1660
1610 A2 = KY: PRINT N$;: GOSUB 390:
    IF NOT PEEK (HF) THEN 1660
1620 IF KY > 25 THEN 1660
1630 A1 = KY: GOSUB 320: IF KY >15
    THEN KY = A1: GOTO 1660
1640 A2 = KY: PRINT N$;: GOSUB 390:
    IF KY < 0 OR KY > 255 THEN CALL
    TT: GOTO 1590
1650 REM CALCULATE NEW CURSOR POSN
1660 POKE CS,KY: CALL MV: CALL TT:
    RETURN
1670 REM CHANGE SLOT NO.
1680 VTAB 22: HTAB 4: GOSUB 320: IF
    KY > 15 THEN CALL TT: RETURN
1690 IF KY < 1 OR KY > 7 THEN 1680
1700 POKE SL,KY * 16: CALL TT:
    RETURN
1710 REM WRITE A TRACK/SECTOR
1720 POKE CM,WR: GOSUB 420:
    POKECM,RD: CALL TT: RETURN
1730 REM CLEAR SCREEN, RECONNECT DOS
    AND EXIT TO BASIC
1740 TEXT : HOME : POKE 103,1:
    POKE104,8: CALL 1002: END
1750 REM FIND PRINTER SLOT
1760 GOSUB 30: VTAB 12: PRINT "WHICH
    SLOT IS YOUR PRINTER USING? 1-7
    ";: GOSUB 320: IFKY > 15 THEN
    RETURN
1770 IF KY > 7 THEN GOSUB 120:
    GOTO1760
1780 IF NOT KY THEN RETURN
1790 PR = KY:LI = 2
1800 PRINT : PRINT : PRINT
    TAB(6)"PRINT USING 80 COLUMNS
    (Y/";: INVERSE : PRINT "N";:
    NORMAL: PRINT "):";: GOSUB 50:
    IFN$ = "Y" THEN LI = 1
1810 RETURN
1820 REM CALL FOR CATALOG
1830 CALL 1002: ONERR GOTO 1850
1840 GOSUB 30: PRINT : PRINT
    CHR$(4)"CATALOG,D" PEEK (DR)",S"
    PEEK (SL) / 16: PRINT : CALLFR:
    POKE 35,24: VTAB 24: HTAB7:
    PRINT "PRESS ANY KEY TO CONTINUE
    ";: GOSUB 210: GOTO730
1850 POKE 216,0: GOTO 240
1860 REM DISASSEMBLE THE BUFFER
1870 GOSUB 30: VTAB 21: PRINT :
    PRINT:KY = PEEK (CS)
1880 REM START AT CURSOR
1890 POKE 58,KY: POKE 59, PEEK (BF)
```

```
1900 A1 = 0:A2 = 21
1910 REM START AT LAST BYTE
1920 FOR X = 1 TO A2: IF PEEK (59) >
    PEEK (BF) THEN :A1 =1: IF PEEK
    (1152) < > 160 THEN PRINT : GOTO
    2090
1930 IF A1 THEN PRINT : NEXT :
    GOTO2090
1940 CALL BI
1950 NEXT
1960 REM <ESC> KEY? = EXIT
1970 GOSUB 210: IF KY = 155 THEN2130
1980 REM <RTN> KEY? = 20 LINES
1990 IF KY = 141 THEN 1900
2000 REM <SPACE> KEY? = 1 LINE
2010 IF KY = 160 THEN A2 = 1:
    GOTO1920
2020 IF KY = 213 THEN GOSUB 140:
    GOSUB 1550: VTAB 1: GOTO 1890
2030 IF KY < > 208 THEN 1970
2040 REM PRINT SCREEN
2050 GOSUB 140:L = KY
2060 IF NOT PR THEN GOSUB 1760
2070 HOME :KY = L: PR# PR: GOTO1890
2080 REM PRINT EXIT MESSAGE
2090 PRINT "END OF BUFFER PRESS
    RETURN TO CONTINUE";: GOSUB210
2100 REM LAST CHANCE TO PRINT
2110 IF KY = 208 THEN 2050
2120 REM EXIT BINARY ROUTINE
2130 PRINT : POKE 35,24: PR# 0:
    GOTO730
2140 REM DEFINE VARIABLES
2150 RD = 1:WR = 2:LI = 2
2160 SL = 2071:DR = 2072:VO =
    2084:TR = 2074:SC = 2075:CM =
    2082
2170 NM = 2091:FL = 2101:EF =
    2094:HF = 2095:CS = 2100:BF =
    2103
2180 NC = 2099
2190 FI = PEEK (FL)
2200 NO = PEEK (2106) + PEEK (2107)
    * 256
2210 IO = 2111:MV = 2114:HX =
    2117:ED = 2120:BI = 2123:FR =
    2126:TT = 2129:XC = 2135
2220 HP = 2141:AP = 2144
2230 F$(1) = "INV1":F$(2) =
    "INV2":F$(3) = "FLS1":F$(4) =
    "FLS2":F$(5) = "CTRL":F$(6) =
    "NOR1":F$(7) = "NOR2":F$(8)
    ="L/C "
2240 F(1) = 192:F(2) = 128:F(3)
    =128:F(4) = 64:F(5) = 64:F(6) =
    0:F(7) = 0:F(8) = - 64
2250 G$ = CHR$ (7)
2260 VTAB 8: PRINT "D I S K E DI
    T     V E R S I O N     4 . 0":
    PRINT " COPYRIGHT 1981 (C) HARD
    CORE COMPUTIST": PRINT
2270 HTAB 5: FOR X = 1 TO 32:
    PRINT"-";: NEXT : PRINT : HTAB
    6: PRINT "A DISK EDITING UTILITY
    PROGRAM"
2280 HTAB 5: FOR X = 1 TO 32:
    PRINT"-";: NEXT : PRINT : PRINT
2290 VTAB 22: PRINT "INSERT DISK --
    PRESS ANY KEY TO CONTINUE";:
    GOSUB 210: VTAB 22: CALL - 958:
    GOTO 730
```

Basic Checksums

```
10  - $DA54     40  - $6989     70  - $50BF
20  - $36E1     50  - $94C6     80  - $97E3
30  - $B237     60  - $195A     90  - $0D4D
100 - $2BF8     840 - $9D16     1590 - $8164
110 - $4A45     850 - $7702     1600 - $934D
120 - $B662     860 - $7FE7
130 - $E036     870 - $5F47     1610 - $A881
140 - $492A     880 - $356C     1620 - $C5AE
150 - $2087     890 - $292A     1630 - $426C
160 - $E1C7     900 - $AA2F     1640 - $454B
170 - $E57B     910 - $D306     1650 - $3AC5
180 - $DD6C     920 - $4C2F     1660 - $D694
190 - $D1C7     930 - $9EB4     1670 - $D136
200 - $C136     940 - $7AB7     1680 - $47A7
                950 - $00E8     1690 - $EA2E
210 - $AF97     960 - $E3C5     1700 - $AFAE
220 - $9409     970 - $7990     1710 - $6B9B
230 - $A7B0     980 - $BFC8     1720 - $438E
240 - $BBF7     990 - $34C7     1730 - $1B69
250 - $91AA     1000 - $A45A    1740 - $5F79
260 - $3363                     1750 - $80C9
270 - $FFFA     1010 - $6FC7    1760 - $F861
280 - $6A18     1020 - $1551    1770 - $481A
290 - $5E94     1030 - $1D31    1780 - $77C0
300 - $3719     1040 - $2DB9    1790 - $263A
310 - $AD76     1050 - $6113    1800 - $7F75
320 - $FCC9     1060 - $BE02
330 - $CFF1     1070 - $15A5    1810 - $AD1B
340 - $0EDA     1080 - $C1BA    1820 - $D0C7
350 - $C818     1090 - $2FDA    1830 - $DB39
360 - $E4F0     1100 - $7199    1840 - $CCB6
370 - $6B10     1110 - $6F5A    1850 - $086F
380 - $48F9     1120 - $724A    1860 - $3C82
390 - $AE66     1130 - $7D0C    1870 - $5970
400 - $D006     1140 - $6B09    1880 - $A79F
                1150 - $42D5    1890 - $53AB
410 - $463D     1160 - $0969    1900 - $168B
420 - $4D8C     1170 - $955E    1910 - $7051
430 - $76B7     1180 - $6291    1920 - $08D3
440 - $0448     1190 - $D5B3    1930 - $BA2B
450 - $EC87     1200 - $E778    1940 - $3C4D
460 - $D9BC                     1950 - $6E63
470 - $6CDA     1210 - $8F98    1960 - $80CA
480 - $034E     1220 - $90CF    1970 - $8A49
490 - $76EA     1230 - $3A4E    1980 - $72C8
500 - $18DF     1240 - $3512    1990 - $6CCD
510 - $2B9D     1250 - $203A    2000 - $6650
520 - $5F64     1260 - $2762
530 - $F31D     1270 - $82AD    2010 - $46CC
540 - $4BA1     1280 - $B30C    2020 - $C93C
550 - $52A1     1290 - $4F4E    2030 - $0CB8
560 - $262D     1300 - $57C5    2040 - $671E
570 - $F656     1310 - $681B    2050 - $3CCF
580 - $1CC2     1320 - $5BDA    2060 - $D0F2
590 - $A588     1330 - $1822    2070 - $7564
600 - $A0D7     1340 - $F8F9    2080 - $7AE8
                1350 - $E38A    2090 - $7654
610 - $B4A0     1360 - $402F    2100 - $C2A5
620 - $F38A     1370 - $CBC6    2110 - $F66F
630 - $5527     1380 - $F7F2    2120 - $E595
640 - $12A1     1390 - $80F8    2130 - $7DB0
650 - $699F     1400 - $A915    2140 - $4EAE
660 - $4C4A                     2150 - $7885
670 - $AB33     1410 - $AC24    2160 - $CDEB
680 - $B0BE     1420 - $433F    2170 - $69CC
690 - $A705     1430 - $5690    2180 - $07E6
700 - $6ECC     1440 - $5AE2    2190 - $67A6
710 - $5AB1     1450 - $F406    2200 - $C651
720 - $5243     1460 - $F87A
730 - $7825     1470 - $1013    2210 - $5F3F
740 - $56B3     1480 - $63B2    2220 - $00CD
750 - $34CF     1490 - $A314    2230 - $AABA
760 - $042D     1500 - $5D9F    2240 - $99F1
770 - $B88D     1510 - $96EA    2250 - $D51B
780 - $ADC9     1520 - $4CF7    2260 - $FD01
790 - $B47B     1530 - $B81E    2270 - $CD77
800 - $8CA5     1540 - $9C93    2280 - $373F
                1550 - $7F79    2290 - $2992
810 - $3FA0     1560 - $B78D
820 - $873F     1570 - $6BF7
830 - $B65C     1580 - $7EB8
```

DiskEdit Source Code

```
0005                .LI
0010
0015 * DISKEDIT II - VERSION 4.1
0020 * COPYRIGHT 1981 SOFTKEY
0025 * LAST UPDATED  MAR 24 84
0030
0035                .OR $800
0040                .TF  EDO
0045
0050 WNDTOP         .EQ  $22
0055 WNDBTM         .EQ  $23
0060 CH             .EQ  $24
0065 CV             .EQ  $25
0070 BASE2          .EQ  $26,27
0075 BASE1          .EQ  $28,29
0080 PCL            .EQ  $3A,3B
0085 IOBPL          .EQ  $48
0090 PRGSTR         .EQ  $67
0095 LOC            .EQ  $E0
0100 NUM            .EQ  $E1
0105 BUFPNTR        .EQ  $E4
0110 DCHR           .EQ  $E7,E8
0115 CFLT           .EQ  $E9,EA
0120 RWTS           .EQ  $3D9
0125 GETIOB         .EQ  $3E3
0130 KEY            .EQ  $C000
0135 STROBE         .EQ  $C010
0140 VTOC           .EQ  $B3F2
0145 LINPRT         .EQ  $ED24
0150 INSDS          .EQ  $F88C
0155 INSTDS         .EQ  $F8D3
0160 PRBLANK        .EQ  $F94A
0165 PCADJ          .EQ  $F953
0170 HOME           .EQ  $FC58
0175 CR.LF          .EQ  $FC62
0180 PRHEX          .EQ  $FDDA
0185 COUT           .EQ  $FDED
0190 *--------------------------------- CHARACTER CODES
0195 CTRL.AT        .EQ  $80
0200 CTRL.A         .EQ  $81
0205 CTRL.B         .EQ  $82
0210 CTRL.D         .EQ  $84
0215 CTRL.F         .EQ  $86
0220 CTRL.H         .EQ  $88
0225 CTRL.I         .EQ  $89
0230 CTRL.L         .EQ  $8C
0235 RETURN         .EQ  $8D
0240 CTRL.N         .EQ  $8E
0245 CTRL.Q         .EQ  $91
0250 CTRL.U         .EQ  $95
0255 CTRL.Z         .EQ  $9A
0260 ESCAPE         .EQ  $9B
0265 SPACE          .EQ  $A0
0270 STAR           .EQ  $AA
0275 PERIOD         .EQ  $AE
0280 FIVE           .EQ  $B5
0285 LTR.I          .EQ  $C9
0290 LTR.J          .EQ  $CA
0295 LTR.K          .EQ  $CB
0300 LTR.M          .EQ  $CD
0305
0310 *------------------------ 1ST LINE OF BASIC PROGRAM
0315
0320 START          .HS  00110800008C3230
0325                .HS  36373AAB31303AB2
0330                .HS  000000
0335                JMP  INITDOS
0340
0345 *--------------------------------- INPUT/OUTPUT BLOCK
0350
0355 IOBIND         .HS  01
0360 SLOT           .HS  60          SLOT * 16
0365 DRIVE          .HS  01          DRIVE #
0370 EXPVOL         .HS  00          REQ. VOLUME
0375 TRACK          .HS  00          TRACK #
0380 SECTOR         .HS  00          SECTOR #
0385                .DA  DCT
0390                .DA  BUFFER
0395                .HS  0000
0400 CMND           .HS  01          COMMAND
0405 ERCODE         .HS  00          ERROR CODE
0410 VOLUME         .HS  00          VOLUME #
```

```
0415 OLDSLOT      .HS 60            PREV. SLOT
0420 OLDRIVE      .HS 01            PREV. DRIVE
0425
0430 DCT          .HS 00            TYPE CODE
0435 PHASES       .HS 01            PHASES/TRK
0440              .HS EFD8          TIME COUNT
0445
0450 *------------------------------------- BASIC variables
0455
0460 BYTE         .HS 00            NM
0465 OLDTRK       .HS 00            OT
0470 OLDSCT       .HS 00            OS
0475 ERRFLG       .HS 00            EF
0480 HEX.OR.DEC.FLG .HS 00 HF
0485 ON.OFF       .HS 01            ST
0490 CFLG         .HS 01            PF
0495              .HS 00
0500 USE.CTRL.CHARS .HS 00 TH
0505 CRSVAL       .HS 00            CS
0510 FLTNUM       .HS 01            FL
0515              .HS 00
0520              .DA /BUFFER       BF
0525              .HS 0000
0530              .DA FSTAT         NO
0535              .HS 000000
0540
0545 *------------------------------------- BASIC Call table
0550
0555              JMP CALLIO        IO
0560              JMP PRINT.SCREEN.DATA
0565              JMP HXBYTE
0570              JMP EDIT          ED
0575              JMP BINARY        BI
0580              JMP CALC.FREE.SECTORS
0585              JMP PROMPT        TT
0590              JMP PROMPT0       T1
0595              JMP PARSE         XC
0600              JMP FILTER0       HC
0605              JMP HEXPRINT      HP
0610              JMP ASCPRINT      AP
0615              JMP RIGHT         UNUSED
0620              .HS 606060        UNUSED
0625
0630 *------------------------------------- INTERNAL VARIABLES
0635
0640 OFFSET       .HS 00
0645 FIRST        .HS FF
0650 EDFLG        .HS 01
0655 HCOUNT       .HS 00
0660 SPACES       .HS 00
0665 EDIT.MODE.FLAG .HS 00
0670 KEYFLG       .HS 01
0675 MAXSCT       .HS 10
0680 MAXTRK       .HS 23
0685 SPECIAL.FUNCTION .HS 00
0690
0695 *------------------------------------- Get DOS pointers
0700
0705 INITDOS      JSR GETIOB
0710              STY IOBPL
0715              STA IOBPL+1
0720              LDY #1
0725              LDA (IOBPL),Y
0730              STA SLOT
0735              INY
0740              LDA (IOBPL),Y
0745              STA DRIVE
0750 *------------------------------------- Reset program pointer
0755              LDA #STOP
0760              STA PRGSTR
0765              LDA /STOP
0770              STA PRGSTR+1
0775              RTS
0780
0785 *------------------------------------- Call Read/Write Track Sector
0790
0795 CALLIO       LDA /IOBIND
0800              LDY #IOBIND
0805              JSR RWTS
0810              BCC .1
0815              LDA ERCODE        GET ERROR #
0820              STA ERRFLG
0825 .1           RTS
0830
0835 *------------------------------------- Put buffer here
0840
0845              .BS $900-*
0850 BUFFER       .BS $100          256 bytes
0855
0860 *-------------------------------Disassemble an instruction
0865
0870 BINARY       LDA #RETURN
0875              JSR COUT          PRINT <CR>
0880              LDA PCL
0885              STA BYTE
0890              JSR HXBYTE
0895 STEP         LDA #4
0900              STA CH
0905              LDA #$AD
0910              JSR COUT          Print dash
0915              LDX #1            and a
0920              JSR PRBLANK       space.
0925              JSR INSDS         Disassem
0930              JSR INSTDS        current
0935              JSR PCADJ         instr.
0940              STA PCL           & update
0945              STY PCL+1         prg cntr.
0950              RTS
0955
0960 *------------------------------------- Select a filter
0965
0970 PRINT.SCREEN.DATA
0975
0980              LDA FLTNUM
0985              ASL
0990              ASL
0995              TAX
1000              LDY #0
1005 .1           LDA FLT.LOC,X
1010              STA DCHR,Y
1015              INX
1020              INY
1025              CPY #4
1030              BCC .1
1035
1040 *------------------------------- Print buffer data to screen
1045
1050              LDA #0
1055              STA BUFPNTR
1060              STA CV
1065 .2           JSR PRINT.OLD.LINE
1070              INC CV
1075              LDA CV
1080              CMP #20           Last line?
1085              BNE .2            No!
1090              LDA CV
1095              JSR FIND.BASE.ADDR
1100              LDY #39
1105              LDA #SPACE
1110 .3           STA (BASE1),Y
1115              DEY
1120              BPL .3
1125              JMP PROMPT
1130
1135 *  Memory locations for text scrn
1140
1145 TEXT.SCREEN.BYTE
1150
1155              .DA $400          Line 1
1160              .DA $480
1165              .DA $500
1170              .DA $580
1175              .DA $600
1180              .DA $680
1185              .DA $700
1190              .DA $780
1195              .DA $428
1200              .DA $4A8
1205              .DA $528
1210              .DA $5A8
1215              .DA $628
1220              .DA $6A8
1225              .DA $728
1230              .DA $7A8
1235              .DA $450
1240              .DA $4D0
1245              .DA $550
1250              .DA $5D0
1255              .DA $650
1260              .DA $6D0
1265              .DA $750
1270              .DA $7D0          Line 24
1275
1280
1285 *-------------------------------Enter with line# in ACC.
1290
1295 FIND.BASE.ADDR
1300
1305              ASL
1310              TAX
1315              LDA TEXT.SCREEN.BYTE,X
1320              STA BASE1
1325              CLC
1330              ADC #27
1335              STA BASE2
1340              LDA TEXT.SCREEN.BYTE+1,X
1345              STA BASE1+1
1350              STA BASE2+1
1355              RTS
1360
1365 *------------------------------- Convert CRSVAL to line#
1370
1375 FIND.CURRENT.LINE
1380
1385              LDX #20
1390 .1           DEX
1395              LDA FIRST.CHAR.POSN,X
1400              CMP CRSVAL
1405              BCC .2
1410              BEQ .2
1415              JMP .1
1420 .2           STX CV
1425              STA BUFPNTR
1430              RTS
1435
1440 FIRST.CHAR.POSN
1445
1450              .HS 000D1A2734414E
1455              .HS 5B6875828F9CA9
1460              .HS B6C3D0DDEAF7
1465
1470 PRINT.NEW.LINE
1475
1480              JSR FIND.CURRENT.LINE
1485
1490 PRINT.OLD.LINE
1495
1500              LDA CV
1505              JSR FIND.BASE.ADDR
1510              LDA #13
1515              STA HCOUNT
1520              LDX BUFPNTR
1525 .2           LDA BUFFER,X
1530              PHA
1535              CPX CRSVAL
1540              BNE .3
1545              LDX KEYFLG
1550              BEQ .4
1555 .3           JSR FILTER
1560 .4           LDY #0
1565              STA (BASE2),Y
1570              INC BASE2
1575              PLA
1580              LDX KEYFLG
1585              BEQ .9
1590              PHA
1595              LSR
1600              LSR
1605              LSR
1610              LSR
1615              LDX BUFPNTR
1620              CPX CRSVAL
1625              BNE .5
1630              ORA #$30
1635              CMP #$3A
1640              BCC .6
1645              SBC #$39
1650              JMP .6
1655 .5           ORA #$B0
1660              CMP #$BA
1665              BCC .6
1670              ADC #$06
1675 .6           STA (BASE1),Y
1680              INC BASE1
1685              PLA
```

```
1690              AND #$0F
1695              LDX BUFPNTR
1700              CPX CRSVAL
1705              BNE .7
1710              ORA #$30
1715              CMP #$3A
1720              BCC .8
1725              SBC #$39
1730              JMP .8
1735 .7           ORA #$B0
1740              CMP #$BA
1745              BCC .8
1750              ADC #$06
1755 .8           STA (BASE1),Y
1760              INC  BASE1
1765 .9           INC  BUFPNTR
1770              LDX BUFPNTR
1775              BEQ .10
1780              DEC HCOUNT
1785              LDA HCOUNT
1790              BNE .2
1795 .10          LDX KEYFLG
1800              BEQ .12
1805              LDA #$A0
1810              STA (BASE1),Y
1815              LDX CV
1820              CPX #19
1825              BNE .12
1830              LDA #SPACE
1835 .11          STA (BASE2),Y
1840              STA (BASE1),Y
1845              INC  BASE1
1850              STA (BASE1),Y
1855              INY
1860              CPY #4
1865              BNE .11
1870              STA (BASE1),Y
1875 .12          RTS
1880
1885 *------------------------------------------------------------
1890
1895 CALC.FREE.SECTORS
1900
1905              LDA #$00
1910              STA NUM
1915              STA NUM+1
1920              LDY #$C8
1925 NXTBYTE      LDA VTOC,Y
1930 NXTBIT       BEQ .2
1935 .1           ASL
1940              BCC NXTBIT
1945              INC  NUM
1950              BNE .1
1955              INC  NUM+1
1960              BNE .1
1965 .2           DEY
1970              BNE NXTBYTE
1975              LDX #15
1980 .3           LDA FSTEXT-1,X
1985              JSR COUT
1990              DEX
1995              BNE .3
2000              LDX NUM
2005              LDA NUM+1
2010              JSR LINPRT
2015              LDA #RETURN
2020              JSR COUT
2025              RTS
2030 FSTEXT       .AS -"'= EERF SROTCES "
2035
2040 *------------------------------------- Screen character filter
2045
2050 FILTER0      LDA BYTE
2055 FILTER       STA LOC
2060              LSR
2065              LSR
2070              LSR
2075              LSR
2080              LSR
2085              TAY
2090              LDA (CFLT),Y
2095              TAX
2100              AND #$F0
2105              CLC
2110              ADC LOC
2115              STA LOC
2120              TXA
2125              AND #$0F
2130              BNE .2
2135 .1           LDA LOC
2140              RTS
2145
2150 *------------------------------------------------Select function
2155
2160 .2           CMP #1            Function 1?
2165              BNE .4
2170              LDA (DCHR),Y
2175              CMP LOC
2180              BNE .1
2185 .3           LDA #SPACE
2190              RTS
2195 .4           CMP #2            Function 2?
2200              BNE .5
2205              LDA (DCHR),Y
2210              CMP LOC
2215              BEQ .1
2220              BNE .3
2225 .5           CMP #3            Function 3?
2230              BEQ .3
2235              JMP .1
2240
2245
2250 *------------------------------------- Filter parameter data
2255
2260 CHG0         .BS 8
2265 CHG1         .BS 8
2270 CHG2         .HS C0808040800000E0
2275 CHG3         .HS C1818141810101E1
2280 CHG4         .HS C0808040010000E0
2285 CHG5         .HS 02020202C00040E0
2290 CHG6         .HS 0000000000000000
2295 CHG7         .HS C0808040800000E0
2300 CHG8         .BS 8
2305 CHG9         .BS 8
2310
2315 *------------------------------------------------------------
2320
2325 DEL0         .BS 8
2330 DEL1         .BS 8
2335 DEL2         .BS 8
2340 DEL3         .HS C0A0C0A000A0C0C0
2345 DEL4         .BS 8
2350 DEL5         .BS 8
2355 DEL6         .BS 8
2360 DEL7         .BS 8
2365 DEL8         .BS 8
2370 DEL9         .BS 8
2375
2380 *-------------------------------FILTER STATUS 1=ON
2385
2390 FSTAT        .HS 00            FILTER #0
2395              .HS 0000000001000000
2400
2405 *------------------------------ FILTER PARM LOCATIONS
2410
2415 FLT.LOC      .DA DEL0
2420              .DA CHG0
2425              .DA DEL1
2430              .DA CHG1
2435              .DA DEL2
2440              .DA CHG2
2445              .DA DEL3
2450              .DA CHG3
2455              .DA DEL4
2460              .DA CHG4
2465              .DA DEL5
2470              .DA CHG5
2475              .DA DEL6
2480              .DA CHG6
2485              .DA DEL7
2490              .DA CHG7
2495              .DA DEL8
2500              .DA CHG8
2505              .DA DEL9
2510              .DA CHG9
2515
2520 *------------------------------------- Print screen prompts
2525
2530 PROMPT0      LDA OLDTRK
2535              STA TRACK
2540              LDA OLDSCT
2545              STA SECTOR
2550 PROMPT       LDA #21
2555              JSR FIND.BASE.ADDR
2560              LDY #0
2565 .1           LDA PROMPT1,Y
2570              STA (BASE1),Y
2575              INY
2580              CPY #3
2585              BCC .1
2590              LDA SLOT
2595              LSR
2600              LSR
2605              LSR
2610              LSR
2615              ORA #$B0
2620              STA (BASE1),Y
2625              INY
2630 .2           LDA PROMPT1,Y
2635              STA (BASE1),Y
2640              INY
2645              CPY #8
2650              BCC .2
2655              LDA DRIVE
2660              ORA #$B0
2665              STA (BASE1),Y
2670              INY
2675 .3           LDA PROMPT1,Y
2680              STA (BASE1),Y
2685              INY
2690              CPY #12
2695              BCC .3
2700              LDA TRACK
2705              JSR PRINT.HEX.OR.DECIMAL
2710 .4           LDA PROMPT1,Y
2715              STA (BASE1),Y
2720              INY
2725              CPY #18
2730              BCC .4
2735              LDA SECTOR
2740              JSR PRINT.HEX.OR.DECIMAL
2745 .5           LDA PROMPT1,Y
2750              STA (BASE1),Y
2755              INY
2760              CPY #24
2765              BCC .5
2770              LDA VOLUME
2775              JSR PRINT.HEX.OR.DECIMAL
2780 .6           LDA PROMPT1,Y
2785              STA (BASE1),Y
2790              INY
2795              CPY #30
2800              BCC .6
2805              LDA CRSVAL
2810              JSR PRINT.HEX.OR.DECIMAL
2815 .7           LDA PROMPT1,Y
2820              STA (BASE1),Y
2825              INY
2830              CPY #35
2835              BCC .7
2840              LDA FLTNUM
2845              ORA #$B0
2850              STA (BASE1),Y
2855              INY
2860 .8           LDA PROMPT1,Y
2865              STA (BASE1),Y
2870              INY
2875              CPY #37
2880              BCC .8
2885              LDX EDIT.MODE.FLAG
2890 .9           LDA EDIT.MODE.TEXT,X
2895              STA (BASE1),Y
2900              INY
2905              INX
2910              CPY #40
2915              BCC .9
2920              RTS
2925 *------------------------------------------------------------
2930 EDIT.MODE.TEXT
2935
2940              .HS 0805180113030 90E
2945              .HS 16060C130C2F03
2950
2955 PROMPT1
2960
```

```
2965                .HS  130CBAA0A0      SL
2970                .HS  0412BAA0A0      DR
2975                .HS  14BAA0A0A0A0    T
2980                .HS  13BAA0A0A0A0    S
2985                .HS  16BAA0A0A0A0    V
2990                .HS  02BAA0A0A0A0    B
2995                .HS  06A0A0A0A0A0    F
3000
3005 *--------------------------------------------------------------
3010
3015 SET.HEX.OR.DEC
3020
3025                LDX #1
3030                CPX HEX.OR.DEC.FLG
3035                BNE .1
3040                DEX
3045 .1             STX HEX.OR.DEC.FLG
3050                JMP PROMPT
3055 *--------------------------------------------------------------
3060 SWT.DRV        LDX #1
3065                CPX DRIVE
3070                BNE .1
3075                INX
3080 .1             STX DRIVE
3085                JMP PROMPT
3090 *--------------------------------------------------------------
3095 FSET           LDA LOC+1
3100                SEC
3105                SBC #$B0
3110                STA FLTNUM
3115                JMP PRINT.SCREEN.DATA
3120 *--------------------------------------------------------------
3125 DEC.SCT        DEC SECTOR
3130                BPL IOJMP
3135                LDX MAXSCT
3140                DEX
3145                STX SECTOR
3150 *--------------------------------------------------------------
3155 DEC.TRK        DEC TRACK
3160                BPL IOJMP
3165                LDX MAXTRK
3170                DEX
3175                STX TRACK
3180 IOJMP          JSR CALLIO
3185                JMP PRINT.SCREEN.DATA
3190 *--------------------------------------------------------------
3195 INC.SCT        INC  SECTOR
3200                LDX SECTOR
3205                CPX MAXSCT
3210                BCC IOJMP
3215                LDX #0
3220                STX SECTOR
3225 *--------------------------------------------------------------
3230 INC.TRK        INC  TRACK
3235                LDX TRACK
3240                CPX MAXTRK
3245                BCC IOJMP
3250                LDX #0
3255                STX TRACK
3260                BEQ IOJMP           ...ALWAYS
3265
3270 *-------------------------CURSOR MOVEMENT ROUTINE
3275
3280 LEFT           JSR FIND.CURRENT.LINE
3285                DEC CRSVAL
3290                JMP CRS1
3295 *--------------------------------------------------------------
3300 RIGHT          JSR FIND.CURRENT.LINE
3305                INC CRSVAL
3310                JMP CRS1
3315 *--------------------------------------------------------------
3320 UP             JSR FIND.CURRENT.LINE
3325                LDA CRSVAL
3330                SEC
3335                SBC #13
3340                BCS .2
3345                CMP #$FC
3350                BCC .1
3355                SBC #14
3360 .1             ADC #4
3365 .2             STA CRSVAL
3370 CRS1           JSR PRINT.OLD.LINE
3375                JSR PRINT.NEW.LINE
3380                JSR PRTCRS
3385                LDX EDFLG
3390                BNE PARSE2
3395                RTS
3400
3405 *--------------------------------------------------------------
3410 DOWN           JSR FIND.CURRENT.LINE
3415                LDA CRSVAL
3420                CLC
3425                ADC #13
3430                BCC .2
3435                CMP #4
3440                BCS .1
3445                ADC #14
3450 .1             ADC #$FB
3455 .2             STA CRSVAL
3460                JMP CRS1
3465
3470 SET.HEX.EDIT
3475
3480                LDA #0
3485 SETMODE        STA EDIT.MODE.FLAG
3490                JMP PROMPT
3495
3500 SET.ASCII.EDIT
3505
3510                LDA #3
3515                BNE SETMODE         ...Always
3520
3525
3530 *--------------------------------------------------------------
3535 PARSE          LDA TRACK
3540                STA OLDTRK
3545                LDA SECTOR
3550                STA OLDSCT
3555 PARSE2         LDX SPECIAL.FUNCTION
3560                BEQ .2
3565                DEX
3570                STX SPECIAL.FUNCTION
3575 .1             LDA KEY
3580                BPL .1
3585                JSR PRINT.SCREEN.DATA
3590 .2             JSR INKEY
3595                LDX #$FD
3600 .3             INX
3605                INX
3610                INX
3615                LDA VALID.CMND.TABLE,X
3620                BEQ .4
3625                CMP LOC
3630                BNE .3
3635                INX
3640                LDA VALID.CMND.TABLE+1,X
3645                PHA
3650                LDA VALID.CMND.TABLE,X
3655                PHA
3660 .4             RTS                 Bad Command
3665
3670 *--------------------------------------------------------------
3675
3680 VALID.CMND.TABLE
3685                .HS C9              I
3690                .DA UP-1
3695                .HS CA              J
3700                .DA LEFT-1
3705                .HS CB              K
3710                .DA RIGHT-1
3715                .HS CD              M
3720                .DA DOWN-1
3725                .HS 88              <-
3730                .DA LEFT-1
3735                .HS 95              ->
3740                .DA RIGHT-1
3745                .HS AC              ,
3750                .DA DEC.TRK-1
3755                .HS AE              .
3760                .DA INC.TRK-1
3765                .HS B1              1
3770                .DA FSET-1
3775                .HS B2              2
3780                .DA FSET-1
3785                .HS B3              3
3790                .DA FSET-1
3795                .HS B4              4
3800                .DA FSET-1
3805                .HS B5              5
3810                .DA FSET-1
3815                .HS B6              6
3820                .DA FSET-1
3825                .HS B7              7
3830                .DA FSET-1
3835                .HS B8              8
3840                .DA FSET-1
3845                .HS B9              9
3850                .DA FSET-1
3855                .HS BC              <
3860                .DA DEC.TRK-1
3865                .HS BE              >
3870                .DA INC.TRK-1
3875                .HS C1              A
3880                .DA SET.ASCII.EDIT-1
3885                .HS C4              D
3890                .DA SWT.DRV-1
3895                .HS C8              H
3900                .DA SET.HEX.EDIT-1
3905                .HS CC              L
3910                .DA DEC.SCT-1
3915                .HS CE              N
3920                .DA INC.SCT-1
3925                .HS D5              U
3930                .DA SET.HEX.OR.DEC-1
3935                .HS B0              0
3940                .DA FILES-1
3945                .HS 00              EOT
3950 *--------------------------------------------------------------
3955 CTRLMV         LDX EDIT.MODE.FLAG
3960                BEQ .4              HEX EDIT.
3965                CMP #CTRL.I
3970                BNE .1
3975                LDA #$40
3980                STA OFFSET
3985                LDA #6              INVERSE
3990                JMP SETMODE
3995 .1             CMP #CTRL.F
4000                BNE .2
4005                LDA #$80
4010                STA OFFSET
4015                LDA #9              FLASHING
4020                JMP SETMODE
4025 .2             CMP #CTRL.N
4030                BNE .3
4035                LDA #0
4040                STA OFFSET
4045                LDA #3              NORMAL
4050                JMP SETMODE
4055 .3             CMP #CTRL.L
4060                BNE .4
4065                LDA #$20
4070                STA OFFSET
4075                LDA #12             Lower Case
4080                JMP SETMODE
4085 .4             CMP #RETURN
4090                BNE .5
4095                JMP RIGHT
4100 .5             CMP #CTRL.U
4105                BNE .6
4110                JMP RIGHT
4115 .6             CMP #CTRL.H
4120                BNE .7
4125                JMP LEFT
4130 .7             CMP #CTRL.Q
4135                BNE .8
4140                JMP UP
4145 .8             CMP #CTRL.Z
4150                BNE .9
4155                JMP DOWN
4160 .9             LDX FIRST
4165                BNE .10
4170                RTS
4175 .10            CMP #CTRL.D
4180                BNE .12
4185                LDX CRSVAL
4190 .11            LDA BUFFER+1,X
4195                STA BUFFER,X
4200                INX
4205                BNE .11
4210                JMP PRINT.SCREEN.DATA
4215 .12            CMP #CTRL.A
4220                BNE .14
4225                LDX #$FE
4230                DEC CRSVAL
4235 .13            LDA BUFFER,X
```

```
4240                STA BUFFER+1,X
4245                DEX
4250                CPX CRSVAL
4255                BNE .13
4260                INC CRSVAL
4265                JMP PRINT.SCREEN.DATA
4270 .14            LDX #1
4275                STX EDFLG
4280                RTS
4285
4290 *--------------------------------------------------------------
4295
4300 EDIT           LDX #$FF
4305                STX FIRST
4310                INX
4315                STX EDFLG           0 = EDIT ON
4320                LDX FLTNUM
4325                STX ON.OFF
4330                LDA FSTAT,X
4335                BNE .1
4340                STA FLTNUM
4345 .1             JSR PRINT.SCREEN.DATA
4350 .2             JSR PRINT.NEW.LINE
4355 .3             JSR PRTCRS
4360                JSR INKEY
4365                CMP #ESCAPE
4370                BNE .5
4375
4380                LDX ON.OFF
4385                STX FLTNUM
4390                LDX #1
4395                STX EDFLG           EDIT OFF
4400                LDA EDIT.MODE.FLAG
4405                BEQ .4
4410                LDA #3
4415                STA EDIT.MODE.FLAG
4420 .4             RTS
4425
4430 * CHECK FOR HEX OR ASCII EDIT
4435
4440 .5             LDX EDIT.MODE.FLAG
4445                BNE .8
4450
4455 * HEX EDIT ROUTINE
4460
4465                CMP #SPACE
4470                BCS .6
4475                JSR CTRLMV
4480                LDX #$FF
4485                STX FIRST
4490                INX
4495                STX EDFLG
4500                JMP .2
4505 .6             JSR CKHEX
4510                CMP #16
4515                BCS .3
4520                INC FIRST
4525                BNE .7
4530                LDX CRSVAL
4535                STA BUFFER,X
4540                JMP .2
4545 .7             STA LOC
4550                LDX CRSVAL
4555                LDA BUFFER,X
4560                ASL
4565                ASL
4570                ASL
4575                ASL
4580                ORA LOC
4585                STA BUFFER,X
4590                JSR FIND.CURRENT.LINE
4595                INC CRSVAL
4600                JSR PRINT.OLD.LINE
4605                LDA #$FF
4610                STA FIRST
4615                JMP .2
4620
4625 * ASCII EDIT ROUTINE
4630
4635 .8             CMP #$A0
4640                BCS .10
4645                LDX USE.CTRL.CHARS
4650                BEQ .14
4655                JSR CTRLMV
4660                LDX #0
4665                CPX EDFLG
4670                BNE .9
4675                JMP .2
4680 .9             STX EDFLG
4685                JMP .14
4690 .10            LDX OFFSET
4695                CPX #$20            LOWER CASE?
4700                BNE .11
4705                CMP #$C1
4710                BCC .14             < "A"
4715                CMP #$DB
4720                BCS .14             > or = "["
4725                BCC .13             ...always
4730 .11            CMP #$C0
4735                BCS .12             > = "@"
4740                ADC OFFSET
4745 .12            CLC
4750 .13            ADC OFFSET
4755 .14            LDX CRSVAL
4760                STA BUFFER,X
4765                JSR FIND.CURRENT.LINE
4770                INC CRSVAL
4775                JSR PRINT.OLD.LINE
4780                JMP .2
4785
4790 *--------------------------------------------------------------
4795
4800 CKHEX          CMP #$B0            "0"
4805                BCC .3              < "0"
4810                CMP #$C7
4815                BCS .3              > "F"
4820                CMP #$BA
4825                BCC .2              < "10"
4830                CMP #$C1
4835                BCC .3              >= "A"
4840                SBC #7
4845 .2             AND #$0F
4850 .3             RTS
4855
4860 *-------------------------------Flashing cursor routine
4865
4870 NOPRESS        LDX CRSVAL
4875                LDA BUFFER,X
4880                PHA
4885                LDA #$20
4890                JSR WAIT.FOR.KEY
4895                LDA #SPACE
4900                JSR WAIT.FOR.KEY
4905                LDX #1
4910                STX KEYFLG
4915                PLA
4920                JSR WAIT.FOR.KEY
4925 INKEY          LDX #0
4930                STX KEYFLG
4935 OUTKEY         LDA KEY
4940                BPL NOPRESS
4945                STA STROBE
4950                STA LOC
4955                STA LOC+1
4960                LDX #1
4965                STX KEYFLG
4970                RTS
4975
4980 *--------------------------------------------------------------
4985
4990 WAIT.FOR.KEY
4995
5000                LDX CRSVAL
5005                STA BUFFER,X
5010                JSR PRINT.NEW.LINE
5015                LDA #60
5020 .1             TAX
5025 .2             LDY KEY
5030                BMI .3
5035                DEX
5040                BNE .2
5045                SBC #1
5050                BNE .1
5055 .3             RTS
5060
5065 *--------------------------------------------------------------
5070
5075 PRTCRS         LDA #21
5080                JSR FIND.BASE.ADDR
5085                LDY #30
5090                LDA CRSVAL
5095                JSR PRINT.HEX.OR.DECIMAL
5100                LDX SPACES
5105                BEQ .7
5110 .6             LDA #SPACE
5115                STA (BASE1),Y
5120                INY
5125                DEX
5130                BNE .6
5135 .7             RTS
5140
5145 *--------------------------------------------------------------
5150
5155 PRINT.HEX.DEC
5160
5165                LDA BYTE
5170
5175 PRINT.HEX.OR.DECIMAL
5180
5185                PHA
5190                LDX HEX.OR.DEC.FLG
5195                BNE PRINT.DECIMAL
5200                STX SPACES
5205
5210 *--------------------------------------------------------------
5215
5220 PRINT.HEX.BYTE
5225
5230                LDA #$A4
5235                STA (BASE1),Y
5240                INY
5245                PLA
5250                PHA
5255                LSR
5260                LSR
5265                LSR
5270                LSR
5275                ORA #$B0
5280                CMP #$BA
5285                BCC .1
5290                ADC #$06
5295 .1             STA (BASE1),Y
5300                INY
5305                PLA
5310                AND #$0F
5315                ORA #$B0
5320                CMP #$BA
5325                BCC .2
5330                ADC #$06
5335 .2             STA (BASE1),Y
5340                INY
5345                STY CH
5350                RTS
5355 *--------------------------------------------------------------
5360
5365 PRINT.DECIMAL
5370
5375                LDX #2
5380                STX SPACES
5385                LDX #$B0
5390                PLA
5395                CMP #100
5400                BCC .2
5405 .1             INX
5410                SBC #100
5415                CMP #100
5420                BCS .1
5425                DEC SPACES
5430                PHA
5435                TXA
5440                STA (BASE1),Y
5445                INY
5450                LDX #$B0
5455                PLA
5460 .2             CMP #10
5465                BCC .4
5470 .3             INX
5475                SBC #10
5480                CMP #10
5485                BCS .3
5490                DEC SPACES
5495 .4             PHA
5500                LDA SPACES
5505                CMP #2
5510                BEQ .5
```

```
5515              TXA
5520              STA (BASE1),Y
5525              INY
5530 .5           PLA
5535              ORA #$B0
5540              STA (BASE1),Y
5545              INY
5550              RTS
5555
5560 *-------------------------------------------------
5565
5570 FILES        LDX #21
5575              STX WNDBTM
5580              LDX #0
5585              STX SPACES
5590              INX
5595              STX SPECIAL.FUNCTION
5600              JSR HOME
5605              INX
5610              JSR PRBLANK
5615              LDA BUFFER+1
5620              JSR HEX2
5625              LDA BUFFER+2
5630              JSR HEX2
5635              JSR CR.LF
5640              JSR CR.LF
5645              LDX #$0B
5650 .1           LDY #2
5655              JSR CR.LF
5660              JSR SPCOUT
5665 .2           LDA BUFFER,X
5670              JSR HEX2
5675              INX
5680              DEY
5685              BNE .2
5690              LDA BUFFER,X
5695              INX
5700              ROL
5705              PHA
5710              BCC .3
5715              LDA #STAR
5720              JSR COUT
5725              JMP .4
5730 .3           JSR SPCOUT
5735 .4           LDY #0
5740              PLA
5745              LSR
5750              BEQ .6
5755 .5           INY
5760              LSR
5765              BCC .5
5770 .6           LDA TYPE,Y
5775              JSR COUT
5780              JSR SPCOUT
5785              LDY #30
5790              STY HCOUNT
5795 .7           LDA BUFFER,X
5800              STA LOC
5805              LSR
5810              LSR
5815              LSR
5820              LSR
5825              LSR
5830              TAY
5835              LDA (CFLT),Y
5840              AND #$F0
5845              CLC
5850              ADC LOC
5855              CMP #CTRL.AT      change
5860              BMI .8            control
5865              CMP #SPACE        characters
5870              BPL .8            to a
5875              LDA #PERIOD       period.
5880 .8           JSR COUT
5885              INX
5890              DEC HCOUNT
5895              BNE .7
5900              INX
5905              INX
5910              BNE .1
5915              DEX
5920              STX BUFPNTR
5925              RTS
5930 TYPE         .AS -"TIABSRAB"
5935
```

```
5940 *-------------------------------- Filter used by BASIC
5945
5950 ASCPRINT     LDA BYTE
5955              CMP #$FF
5960              BNE .1
5965              LDA #$A0
5970              STA BYTE
5975 .1           LSR
5980              LSR
5985              LSR
5990              LSR
5995              LSR
6000              TAY
6005              LDA ASCFD,Y
6010              CLC
6015              ADC BYTE
6020              JMP COUT
6025 ASCFD        .HS C080804040000C0
6030
6035 HEX0         LDA #$A4          "$"
6040              JSR COUT
6045 HEXPRINT     LDA BYTE
6050 HEX2         JSR PRHEX
6055 SPCOUT       LDA #SPACE
6060              JMP COUT
6065
6070 *-------------------------------- Print Hex or Decimal
6075
6080 HXBYTE       LDX HEX.OR.DEC.FLG
6085              BEQ HEX0          0 = HEX
6090              LDX BYTE
6095              LDA #0
6100              JSR LINPRT
6105              JMP SPCOUT
6110
6115 *-------------------------------------------------
6120
6125              .HS 00
6130 STOP         .HS 0000
6135
6140 *-------------------------------------------------
```

Hexdump with checksums

```
0800:00 35 08 0A 00 89 3A 97    $647E
0808:3A BA 22 41 43 45 57 52    $4222
0810:49 54 45 52 22 C3 34 31    $DF96
0818:29 22 42 49 4E 41 52 59    $0D9C
0820:20 43 48 45 43 4B 53 55    $6302
0828:4D 20 47 45 4E 45 52 41    $2730
0830:54 4F 52 22 00 91 08 14    $F25A
0838:00 A2 35 3A BA 22 4E 41    $DB70
0840:4D 45 20 4F 46 20 46 49    $EEC8
0848:4C 45 20 54 4F 20 53 41    $90F4

0850:56 45 20 3D 22 3B 3A B0    $12BB
0858:31 30 30 30 3A 41 24 D0    $B4CC
0860:4E 4D 24 3A 81 41 D0 32    $ADDB
0868:C1 E3 28 41 24 29 C8 31    $977F
0870:3A 41 31 24 D0 E8 28 41    $48FE
0878:24 2C 41 C9 31 29 3A AD    $B46E
0880:EA 28 41 24 2C 41 2C 31    $A3D3
0888:29 D1 CF 22 2C 22 C4 82    $6F26
0890:00 13 09 1E 00 A2 31 30    $D4DF
0898:3A BA 22 53 54 41 52 54    $C13F

08A0:2C 45 4E 44 20 3D 22 3B    $EED0
08A8:3A B0 31 30 30 30 3A 81    $5997
08B0:41 D0 32 C1 E3 28 4E 4D    $4E45
08B8:24 29 C8 31 3A 4E 31 24    $93EB
08C0:D0 E8 28 4E 4D 24 2C 41    $B82F
08C8:C9 31 29 3A AD 28 EA 28    $EEFF
08D0:4E 4D 24 2C 41 2C 31 29    $DD8F
08D8:CF D0 22 30 22 CD EA 28    $1276
08E0:4E 4D 24 2C 41 2C 31 29    $61D6
08E8:D1 D0 22 39 22 29 CE 28    $86BD
08F0:EA 28 4E 4D 24 2C 41 2C    $C908
08F8:31 29 CF D0 22 41 22 CD    $AD32
0900:EA 28 4E 4D 24 2C 41 2C    $1237
0908:31 29 D1 D0 22 46 22 29    $403C
0910:C4 82 00 42 09 28 00 51    $CD11
0918:D0 41 3A 42 41 D0 32 35    $2B09
0920:32 3A B0 31 30 32 30 3A    $1D84
0928:4E 31 24 D0 E9 28 4E 4D    $93DB
0930:24 2C E3 28 4E 4D 24 29    $77BA
0938:C9 51 29 3A B0 31 30 32    $E956

0940:30 00 8F 09 32 00 97 3A    $5B53
0948:BA 3A BA 22 04 4E 4F 4D    $CFB3
0950:4F 4E 43 22 3A BA 22 04    $B956
0958:4D 4F 4E 49 4F 22 3A BA    $2C8F
0960:22 04 4F 50 45 4E 22 41    $6682
0968:24 3A BA 22 04 44 45 4C    $A529
0970:45 54 45 22 41 24 3A BA    $7E4E
0978:22 04 4F 50 45 4E 22 41    $1413
0980:24 3A BA 22 04 57 52 49    $9AD2
0988:54 45 22 41 31 24 00 A9    $3CB3

0990:09 3C 00 8C 36 30 38 3A    $A5D7
0998:BA 22 04 43 4C 4F 53 45    $8427
09A0:22 3A BA 22 04 46 50 22    $D23B
09A8:00 EF 09 E8 03 B9 35 31    $94D9
09B0:2C 31 39 30 3A 8C 36 34    $F77C
09B8:38 37 34 3A 4E 4D 24 D0    $9E70
09C0:22 22 3A 81 41 D0 35 31    $8E83
09C8:32 C1 37 36 38 3A AD E2    $4360
09D0:28 41 29 D1 CF 31 34 31    $0074
09D8:C4 4E 4D 24 D0 4E 4D 24    $2AE4

09E0:C8 E7 28 E2 28 41 29 C9    $E747
09E8:31 32 38 29 3A 82 00 F5    $6A8C
09F0:09 F2 03 B1 00 0E 0A FC    $0D5C
09F8:03 AD E3 28 4E 31 24 29    $CBF8
0A00:D1 31 CE E3 28 4E 31 24    $E0A8
0A08:29 CF 34 C4 AC 00 37 0A    $4AB0
0A10:06 04 AD E3 28 4E 31 24    $B87E
0A18:29 D1 34 C4 4E 31 24 D0    $8735
0A20:E8 28 22 30 30 30 30 22    $9092
0A28:2C 34 C9 E3 28 4E 31 24    $D5C0

0A30:29 29 C8 4E 31 24 00 7C    $D645
0A38:0A 10 04 44 4E D0 30 3A    $CC4E
0A40:81 41 D0 31 C1 34 3A 44    $4532
0A48:47 D0 E5 28 EA 28 4E 31    $8524
0A50:24 2C 41 2C 31 29 29 3A    $E0DE
0A58:AD EA 28 4E 31 24 2C 41    $C37A
0A60:2C 31 29 CF 22 39 22 C4    $E8F8
0A68:44 47 D0 E6 28 EA 28 4E    $F3D8
0A70:31 24 2C 41 2C 31 29 29    $3C88
0A78:C9 35 35 00 C3 0A 1A 04    $F924

0A80:44 4E D0 44 4E C8 D3 28    $CB22
0A88:31 36 CC 28 34 C9 41 29    $7143
0A90:CA 44 47 29 3A 82 3A B9    $F9C5
0A98:42 41 2C 44 4E C9 D3 28    $72B8
0AA0:44 4E CB 32 35 36 29 CA    $1BD3
0AA8:32 35 36 3A B9 42 41 C8    $04B1
0AB0:31 2C 44 4E CB 32 35 36    $D7F7
0AB8:3A 42 41 D0 42 41 C8 32    $4E50
0AC0:3A B1 00 00 00 4E CD 08    $3E58
0AC8:52 94 00 00 41 00 83 20    $68A4

0AD0:00 00 00 41 80 14 06 95    $3722
0AD8:00 00 41 B1 10 7E 94 00    $12F0
```

```
0AE0:00 4E B1 04 43 94 00 00  $6187
0AE8:51 00 83 00 00 00 00 42  $E870
0AF0:41 89 00 00 00 00 44 4E  $7D64
0AF8:8D 11 00 00 00 44 47 00  $27B2
0B00:00 00 00 00 09 30 C9 3A  $C71B
0B08:90 0D E9 39 4C 17 0B 09  $4CEC
0B10:B0 C9 BA 90 02 69 06 91  $7678
0B18:28 E6 28 68 29 0F A6 E4  $341F

0B20:EC 34 08 D0 0B 09 30 C9  $2BBA
0B28:3A 90 0D E9 39 4C 38 0B  $9C01
0B30:09 B0 C9 BA 90 02 69 06  $0453
0B38:91 28 E6 28 E6 E4 A6 E4  $43D2
0B40:F0 08 CE 6C 08 AD 6C 08  $3752
0B48:D0 91 AE 6F 08 F0 1B A9  $44B2
0B50:A0 91 28 A6 25 E0 13 D0  $9A21
0B58:11 A9 A0 91 26 91 28 E6  $3EFF
0B60:28 91 28 C8 C0 04 D0 F3  $0C9A
0B68:91 28 60 A9 00 85 E1 85  $AB78

0B70:E2 A0 C8 B9 F2 B3 F0 0B  $A006
0B78:0A 90 FB E6 E1 D0 F9 E6  $6CA9
0B80:E2 D0 F5 88 D0 ED A2 0F  $9AED
0B88:BD 9D 0B 20 ED FD CA D0  $D05C
0B90:F7 A6 E1 A5 E2 20 24 ED  $EAA3
0B98:A9 8D 20 ED FD 60 A0 BD  $E59B
0BA0:A0 C5 C5 D2 C6 A0 D3 D2  $24F5
0BA8:CF D4 C3 C5 D3 A0 AD 2B  $C3AE
0BB0:08 85 E0 4A 4A 4A 4A 4A  $8E79
0BB8:A8 B1 E9 AA 29 F0 18 65  $A268

0BC0:E0 85 E0 8A 29 0F D0 03  $83A4
0BC8:A5 E0 60 C9 01 D0 09 B1  $8305
0BD0:E7 C5 E0 D0 F3 A9 A0 60  $D157
0BD8:C9 02 D0 08 B1 E7 C5 E0  $602E
0BE0:F0 E6 D0 F1 C9 03 F0 ED  $D1A9
0BE8:4C C8 0B 00 00 00 00 00  $F062
0BF0:00 00 00 00 00 00 00 00  $F0C2
0BF8:00 00 00 C0 80 80 40 80  $50C2
0C00:00 00 E0 C1 81 81 41 81  $9B2C
0C08:01 01 E1 C0 80 80 40 01  $490A

0C10:00 00 E0 02 02 02 02 C0  $1330
0C18:00 40 E0 00 00 00 00 00  $9370
0C20:00 00 00 C0 80 80 40 80  $C350
0C28:00 00 E0 00 00 00 00 00  $4330
0C30:00 00 00 00 00 00 00 00  $B3B0
0C38:00 00 00 00 00 00 00 00  $4330
0C40:00 00 00 00 00 00 00 00  $B3B0
0C48:00 00 00 00 00 00 00 00  $4330
0C50:00 00 00 C0 A0 C0 A0 00  $7390
0C58:A0 C0 C0 00 00 00 00 00  $B3B0

0C60:00 00 00 00 00 00 00 00  $4330
0C68:00 00 00 00 00 00 00 00  $B3B0
0C70:00 00 00 00 00 00 00 00  $4330
0C78:00 00 00 00 00 00 00 00  $B3B0
0C80:00 00 00 00 00 00 00 00  $4330
0C88:00 00 00 00 00 00 00 00  $B3B0
0C90:01 00 00 00 00 3B 0C EB  $DB73
0C98:0B 43 0C F3 0B 4B 0C FB  $CFA7
0CA0:0B 53 0C 03 0C 5B 0C 0B  $ECEE
0CA8:0C 63 0C 13 0C 6B 0C 1B  $0C66

0CB0:0C 73 0C 23 0C 7B 0C 2B  $AC2E
0CB8:0C 83 0C 33 0C AD 2C 08  $3740
0CC0:8D 1A 08 AD 2D 08 8D 1B  $5DC3
0CC8:08 A9 15 20 8F 0A A0 00  $6B18

0CD0:B9 72 0D 91 28 C8 C0 03  $010C
0CD8:90 F6 AD 17 08 4A 4A 4A  $695F
0CE0:4A 09 B0 91 28 C8 B9 72  $788D
0CE8:0D 91 28 C8 C0 08 90 F6  $0D45
0CF0:AD 18 08 09 B0 91 28 C8  $837A
0CF8:B9 72 0D 91 28 C8 C0 0C  $66C9

0D00:90 F6 AD 1A 08 20 D1 10  $6E2B
0D08:B9 72 0D 91 28 C8 C0 12  $8547
0D10:90 F6 AD 1B 08 20 D1 10  $684D
0D18:B9 72 0D 91 28 C8 C0 18  $69C4
0D20:90 F6 AD 24 08 20 D1 10  $0FA5
0D28:B9 72 0D 91 28 C8 C0 1E  $B82F
0D30:90 F6 AD 34 08 20 D1 10  $1E16
0D38:B9 72 0D 91 28 C8 C0 23  $44F2
0D40:90 F6 AD 35 08 09 B0 91  $6134
0D48:28 C8 B9 72 0D 91 28 C8  $36A3

0D50:C0 25 90 F6 AE 6E 08 BD  $5371
0D58:63 0D 91 28 C8 E8 C0 28  $0D65
0D60:90 F5 60 08 05 18 01 13  $6F98
0D68:03 09 0E 16 06 0C 13 0C  $3245
0D70:2F 03 13 0C BA A0 A0 04  $BEF4
0D78:12 BA A0 A0 14 BA A0 A0  $ADC0
0D80:A0 A0 13 BA A0 A0 A0 A0  $ACAE
0D88:16 BA A0 A0 A0 A0 02 BA  $48FC
0D90:A0 A0 A0 A0 06 A0 A0 A0  $6724
0D98:A0 A0 A2 01 EC 2F 08 D0  $24A7

0DA0:01 CA 8E 2F 08 4C C9 0C  $563F
0DA8:A2 01 EC 18 08 D0 01 E8  $96E0
0DB0:8E 18 08 4C C9 0C A5 E1  $3C14
0DB8:38 E9 B0 8D 35 08 4C 29  $9D5D
0DC0:0A CE 1B 08 10 13 AE 70  $B118
0DC8:08 CA 8E 1B 08 CE 1A 08  $2027
0DD0:10 07 AE 71 08 CA 8E 1A  $460A
0DD8:08 20 90 08 4C 29 0A EE  $26F5
0DE0:1B 08 AE 1B 08 EC 70 08  $2725
0DE8:90 EF A2 00 8E 1B 08 EE  $B3AE

0DF0:1A 08 AE 1A 08 EC 71 08  $B5D6
0DF8:90 DF A2 00 8E 1A 08 F0  $1A38
0E00:D8 20 A3 0A CE 34 08 4C  $1C1A
0E08:29 0E 20 A3 0A EE 34 08  $8C42
0E10:4C 29 0E 20 A3 0A AD 34  $5101
0E18:08 38 E9 0D B0 08 C9 FC  $9865
0E20:90 02 E9 0E 69 04 8D 34  $FC0A
0E28:08 20 CF 0A 20 CC 0A 20  $7624
0E30:B3 10 AE 6B 08 D0 32 60  $AE20
0E38:20 A3 0A AD 34 08 18 69  $C427

0E40:0D 90 08 C9 04 B0 02 69  $FD24
0E48:0E 69 FB 8D 34 08 4C 29  $78D7
0E50:0E A9 00 8D 6E 08 4C C9  $0B63
0E58:0C A9 03 D0 F6 AD 1A 08  $4C90
0E60:8D 2C 08 AD 1B 08 8D 2D  $273B
0E68:08 AE 72 08 F0 0C CA 8E  $E579
0E70:72 08 AD 00 C0 10 FB 20  $0357
0E78:29 0A 20 83 10 A2 FD E8  $CC82
0E80:E8 E8 BD 95 0E F0 0D C5  $977A
0E88:E0 D0 F4 E8 BD 96 0E 48  $A0DA

0E90:BD 95 0E 48 60 C9 12 0E  $C69A
0E98:CA 00 0E CB 09 0E CD 37  $3495
0EA0:0E 88 00 0E 95 09 0E AC  $F571
0EA8:CC 0D AE EE 0D B1 B5 0D  $D54C
0EB0:B2 B5 0D B3 B5 0D B4 B5  $768F
0EB8:0D B5 B5 0D B6 B5 0D B7  $599F

0EC0:B5 0D B8 B5 0D B9 B5 0D  $71B0
0EC8:BC CC 0D BE EE 0D C1 58  $D1F5
0ED0:0E C4 A7 0D C8 50 0E CC  $CD42
0ED8:C0 0D CE DE 0D D5 99 0D  $D5C1

0EE0:B0 3F 11 00 AE 6E 08 F0  $C13A
0EE8:38 C9 89 D0 0A A9 40 8D  $23E1
0EF0:69 08 A9 06 4C 53 0E C9  $B6F3
0EF8:86 D0 0A A9 80 8D 69 08  $D6BC
0F00:A9 09 4C 53 0E C9 8E D0  $97FA
0F08:0A A9 00 8D 69 08 A9 03  $B3C3
0F10:4C 53 0E C9 8C D0 0A A9  $F89C
0F18:20 8D 69 08 A9 0C 4C 53  $76AF
0F20:0E C9 8D D0 03 4C 0A 0E  $96B6
0F28:C9 95 D0 03 4C 0A 0E C9  $78C9

0F30:88 D0 03 4C 01 0E C9 91  $D487
0F38:D0 03 4C 13 0E C9 9A D0  $2B41
0F40:03 4C 38 0E AE 6A 08 D0  $2F6C
0F48:01 60 C9 84 D0 0F AE 34  $1797
0F50:08 BD 01 09 9D 00 09 E8  $EDFA
0F58:D0 F7 4C 29 0A C9 81 D0  $C9C4
0F60:17 A2 FE CE 34 08 BD 00  $D538
0F68:09 9D 01 09 CA EC 34 08  $849A
0F70:D0 F4 EE 34 08 4C 29 0A  $B78E
0F78:A2 01 8E 6B 08 60 A2 FF  $4116

0F80:8E 6A 08 E8 8E 6B 08 AE  $763B
0F88:35 08 8E 30 08 BD 8B 0C  $9541
0F90:D0 03 8D 35 08 20 29 0A  $413D
0F98:20 CC 0A 20 B3 10 20 83  $82D5
0FA0:10 C9 9B D0 16 AE 30 08  $0BC3
0FA8:8E 35 08 A2 01 8E 6B 08  $4097
0FB0:AD 6E 08 F0 05 A9 03 8D  $CF37
0FB8:6E 08 60 AE 6E 08 D0 4A  $BF81
0FC0:C9 A0 B0 0F 20 E4 0E A2  $F1E6
0FC8:FF 8E 6A 08 E8 8E 6B 08  $74AA

0FD0:4C 98 0F 20 54 10 C9 10  $C314
0FD8:B0 C1 EE 6A 08 D0 09 AE  $1B9F
0FE0:34 08 9D 00 09 4C 98 0F  $C226
0FE8:85 E0 AE 34 08 BD 00 09  $3B7B
0FF0:0A 0A 0A 0A 05 E0 9D 00  $37CD
0FF8:09 20 A3 0A EE 34 08 20  $7598
1000:CF 0A A9 FF 8D 6A 08 4C  $49E0
1008:98 0F C9 A0 B0 18 AE 33  $82F3
1010:08 F0 2F 20 E4 0E A2 00  $F459
1018:EC 6B 08 D0 03 4C 98 0F  $B962

1020:8E 6B 08 4C 42 10 AE 69  $1029
1028:08 E0 20 D0 0A C9 C1 90  $02B6
1030:11 C9 DB B0 0D 90 08 C9  $7FE7
1038:C0 B0 03 6D 69 08 18 6D  $B821
1040:69 08 AE 34 08 9D 00 09  $F7A8
1048:20 A3 0A EE 34 08 20 CF  $F079
1050:0A 4C 98 0F C9 B0 90 10  $1611
1058:C9 C7 B0 0C C9 BA 90 06  $2DF0
1060:C9 C1 90 04 E9 07 29 0F  $7C5F
1068:60 AE 34 08 BD 00 09 48  $AE70

1070:A9 20 20 9A 10 A9 A0 20  $4D4E
1078:9A 10 A2 01 8E 6F 08 68  $FEC1
1080:20 9A 10 A2 00 8E 6F 08  $5D69
1088:AD 00 C0 10 DC 8D 10 C0  $E05C
1090:85 E0 85 E1 A2 01 8E 6F  $83E7
1098:08 60 AE 34 08 9D 00 09  $0C2B
```

continued on page 51

An inside look at disk formats with

DiskView

By Charles Haight

This program is called DiskView. DiskView is a mini "nibbler." It will read the raw nibbilized data from a disk without regard to disk format.

This means data can be viewed on a nonstandard format disk (copy-protected) as easily as from a normal DOS formatted disk. With DiskView, a nonstandard disk can be examined to see what was changed. Often these changes are minor and a similar change can be made to your DOS. This would allow use of DiskEdit to read that disk.

To understand these changes lets examine the data pattern on a normal DOS 16 disk.

DOS formats a track by first writing a unique byte called a "sync byte." This byte (normally $FF) allows the Disk II hardware to synchronize with the data on the disk. DOS then writes an address field, some more sync bytes and the data field. At this time the data field is full of $00s. DOS goes on to write sixteen sets of address and data fields on each track. These sets of address and data fields are called sectors.

The following is a normal address field for 3.3 DOS:

D5AA96FFFEAABBAEAAFBEFDEAAEB

It can be broken down into:

Start of address	D5 AA 96
Volume number	FF FE
Track	AA BB
Sector	AE AA
Checksum	FB EF
End of address	DE AA EB

The volume, track, sector and checksum are in a 4+4 coded format. This means that 4 bits in each byte are actual data. The first byte is rotated left and logically AND-ed with the second byte to recover the data.

The data field consists of:

Start of data	D5 AA AD
Encoded data	(341 bytes)
Checksum	(1 byte)
End of data	DE AA EB

The data field is encoded in a 2+6 format. Six bits of each byte are valid data.

The basic structure of 3.2 DOS is similar to 3.3 DOS with these notable exceptions:

1. When initializing a disk, DOS 3.2 does not write a blank data sector. Instead it just writes enough $FFs to fill the space a data sector would use. Trying to read a track/sector that has never been written to will always generate I/O errors.

2. The data is encoded in a 3+5 format which requires 410 bytes to encode 256 data bytes. This is one reason why there are only 13 sectors.

More on Diskview

The format of DiskView is similar to DiskEdit. A full screen of hexadecimal bytes is displayed with the status prompts at the bottom of the screen. The buffer extends from $2000 to $4000 hex which is large enough to ensure reading in an entire track. The slot, drive and track are selectable. Half-tracks can be accessed by appending a ".5" to the track number. The commands are:

D - change the drive
L - read last track (steps by half tracks)
N - read next track (steps by half tracks)
P - print screen contents
R - read the current track
S - change the slot
T - select a track or half track
X - exit to basic
→ - increment buffer
← - decrement buffer

Type in the program and save it to disk. Be especially careful with the data statements. When those values are poked into memory they become a machine language subroutine that is the heart of the program. Run the program. When the COMMAND prompt flashes, press the R key. The screen will fill with hex bytes that show the data stored on the disk.

CAUTION: Utility Nibbler is DOS dependent. It calls directly into DOS to step the drive motor. DOS 3.3 and 48K of memory are needed. This program can be used to read 13 or 16-sector disks or any other Apple disk, but it will only run under a 48K Apple 3.3 DOS.

```
10 TEXT : HOME : IN# 0: PR# 0:
   LOMEM: 16384: POKE 1144,90:
   GOTO 90
20 KY% = PEEK ( - 16384): IF KY%
   < 128 THEN 20
30 POKE - 16368,0: RETURN
40 FOR X = 1 TO 40: PRINT "-";:
   NEXT : RETURN
50 GOSUB 60: POKE 781,0: POKE
   1144,90: POKE TR%,0: CALL
   IO%: POKE 781,255: POKE
   TR%,TK%: CALL IO%: RETURN
60 VTAB 23: HTAB 2: INVERSE :
   PRINT "SLOT";: HTAB 10:
   PRINT "DRIVE";: HTAB 19:
   PRINT "TRACK";: NORMAL
70 VTAB 23: HTAB 7: PRINT PEEK
   (S1%) / 16;: HTAB 16: PRINT
   PEEK (DR%) - PEEK (S1%);:
   HTAB 25: PRINT "     "B$B$
   B$B$ PEEK (TR%) / 2
80 RETURN
90 GOSUB 540
100 IN% = PEEK (CT%): VTAB 21:
   HTAB 32: PRINT "PAGE "IN% -
   31: GOSUB 60: VTAB 23: HTAB
   30: CALL - 868: FLASH :
   PRINT ">COMMAND<": NORMAL :
   GOSUB 20
110 IF KY% = 210 THEN GOSUB 480
120 IF KY% = 211 THEN GOSUB 390
130 IF KY% = 216 THEN GOSUB 410
140 IF KY% = 212 THEN GOSUB 420
150 IF KY% = 199 THEN GOSUB 270
160 IF KY% = 196 THEN GOSUB 230
170 IF KY% = 208 THEN GOSUB 290
180 IF KY% = 136 THEN GOSUB 250
190 IF KY% = 149 THEN GOSUB 370
200 IF KY% = 204 THEN GOSUB 490
210 IF KY% = 206 THEN GOSUB 510
220 GOTO 100
230 VTAB 23: HTAB 30: INVERSE :
   PRINT G$"SET DRIVE";: HTAB
   10: FLASH : PRINT "DRIVE";:
   NORMAL : HTAB 16: PRINT " "
   CHR$ (8);: GET A$:DR = VAL
   (A$): IF DR < 1 OR DR > 2
   THEN 230
240 POKE DR%, PEEK (S1%) + DR:
   GOTO 50
250 IN% = IN% - 1: IF IN% < 32
   THEN IN% = 32
260 POKE CT%,IN%: CALL MV%:
   RETURN
270 PRINT G$: IF G$ = CHR$ (7)
   THEN G$ = "": RETURN
280 IF G$ = "" THEN G$ = CHR$
   (7): RETURN
290 VTAB 23: HTAB 30: FLASH :
   PRINT G$">PRINTER<";:
   NORMAL
300 PR# 1
310 BUFFER% = PEEK (CT%) * 256
320 PRINT : PRINT "TRACK "TK%
```

```
330 FOR X = 0 TO 255 STEP 13:
    FOR Y = 0 TO 12: POKE NM%,
    PEEK (BUFFER% + X + Y): CALL
    HX%: PRINT " ";: NEXT Y:
    PRINT
340 IF PEEK ( - 16384) = 155
    THEN 360
350 NEXT X
360 PR# 0: POKE - 16368,0:
    RETURN
370 IN% = IN% + 1: IF IN% > 63
    THEN IN% = 63
380 POKE CT%,IN%: CALL MV%:
    RETURN
390 VTAB 23: HTAB 30: INVERSE :
    PRINT G$"NEW SLOT?";: HTAB
    2: FLASH : PRINT "SLOT";:
    NORMAL : HTAB 7: PRINT " "
    CHR$ (8);: GET A$:KY% = VAL
    (A$): IF KY% < 1 OR KY% > 7
    THEN 390
400 POKE S1%,KY% * 16: POKE
    S2%,KY% * 16: GOTO 240
410 TEXT : HOME : POKE 33,33:
    CALL 1002: END
420 C$ = "": VTAB 23: HTAB 30:
    INVERSE : PRINT "SET
    TRACK";: HTAB 19: FLASH :
    PRINT G$"TRACK";: NORMAL :
    PRINT "     " CHR$ (8) CHR$
    (8) CHR$ (8);: GET A$:C$ =
    C$ + A$: PRINT A$;: GET
    A$:C$ = C$ + A$: IF A$ = CHR$
    (13) THEN 460
430 PRINT A$;
440 GET A$:C$ = C$ + A$: PRINT
    A$;
450 IF A$ = "." THEN GET A$:C$ =
    C$ + A$: PRINT A$;
460 KY = VAL (C$): IF KY < 0 OR
    KY > 35 THEN 420
470 TK% = KY * 2
480 POKE CT%,32: VTAB 23: HTAB
    30: FLASH : PRINT
    ">>>READ<<<"G$;: NORMAL :
    PRINT " ";: POKE TR%,TK%:
    GOSUB 70: CALL IO%: GOTO 60
490 TK% = TK% - 1: IF TK% < 0
    THEN TK% = 71
500 GOTO 480
510 TK% = TK% + 1: IF TK% > 71
    THEN TK% = 0
520 GOTO 480
530 STOP
540 FOR X = 768 TO 894: READ X%:
    POKE X,X%: NEXT X
550 DATA162,97,189,137,192,162,
    96,189,137,192,160,5,169,2
    55,32,168,252,136,16,248,1
    69,0,32,160,185,189,142,19
    2,169,0,133,30,169,32,133,
    31,162,96,160,0,189,140,19
    2,16,251,145,30,230,30,
    208,245,230,31,165,31,201,
    64,144,237
560 DATA189,136,192,169,1,133,
    37,32,34,252,169,0,133,36,
    133,30,169,13,133,31,162,1
    ,32,74,249,166,30,189,
    0,32,32,218,253,162,1,32,7
    4,249,230,30,240,7,198,31,
    208,235,76,75,3,32,156,252
    ,230,37,32,34,252,169,22
570 DATA 133,34,96,169,172,32,
    218,253,96
580 S1% = 774:S2% = 805:DR% =
    769:TR% = 789:MV% = 830:CT%
    = 856:B$ = CHR$ (8):G$ =
    CHR$ (7):IO% = 768:NM% =
    890:HX% = 889:DR = 1
590 GOSUB 40: VTAB 8: HTAB 10:
    PRINT "COPYRIGHT 1981 (C)":
    PRINT : HTAB 10: PRINT "ALL
    RIGHTS RESERVED": PRINT :
    HTAB 10: PRINT "HARDCORE
    COMPUTING": PRINT : HTAB 10:
    PRINT "P.O. BOX 44549":
    PRINT : HTAB 10: PRINT
    "TACOMA, WA 98444"
```

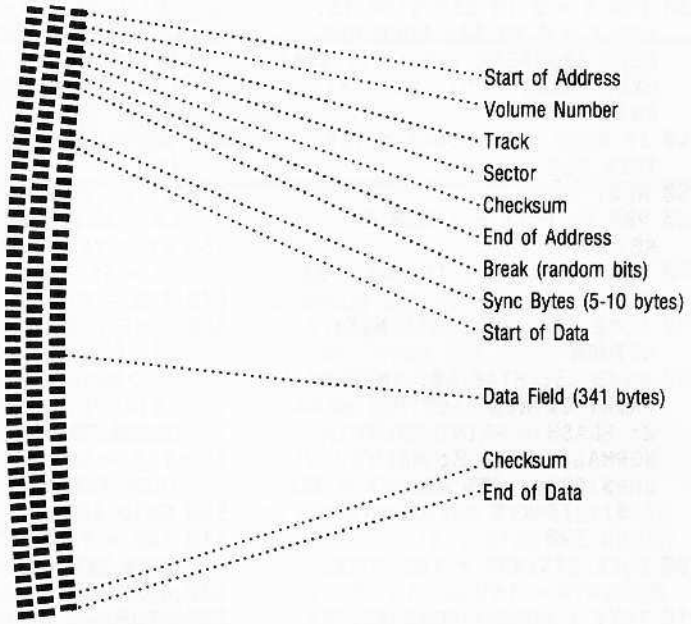
4 plus 4 Conversion Chart

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

```
600 VTAB 22: GOSUB 40: GOTO 60
```

Checksums for DiskView

```
10 - $23C0      210 - $001F      410 - $A1CC
20 - $5B85      220 - $0470      420 - $2481
30 - $E458      230 - $2FC5      430 - $A968
40 - $6CA2      240 - $E31D      440 - $8FDE
50 - $5C41      250 - $25D3      450 - $D7DC
60 - $C6C3      260 - $86BC      460 - $FD3C
70 - $EF9F      270 - $C22E      470 - $535E
80 - $6ABD      280 - $57E2      480 - $A082
90 - $026B      290 - $BFFD      490 - $ECAC
100 - $E564     300 - $10D1      500 - $6EC7
110 - $39D0     310 - $574E      510 - $BCAB
120 - $B076     320 - $2521      520 - $F92D
130 - $CEB8     330 - $E3CD      530 - $AFB1
140 - $A51A     340 - $F9DA      540 - $E804
150 - $F118     350 - $26AE      550 - $9AAF
160 - $6CAC     360 - $D2C2      560 - $01A1
170 - $FE0F     370 - $857E      570 - $835E
180 - $F494     380 - $1320      580 - $9F94
190 - $2B87     390 - $111E      590 - $0133
200 - $2BEC     400 - $1C3F      600 - $03C6
```



DOS Address and Data Mark Locations

DOS 3.2

	Read Locations HEX	Read Locations DECIMAL	Write Locations HEX	Write Locations DECIMAL
Start of Address....	B976 D5	47478 213	BEF5 D5	48885 213
	B980 AA	47488 170	BEFA AA	48890 170
	B98B B5	47499 181	BEFF B5	48895 181
End of Address.....	B9B2 DE	47538 222	BF29 DE	48937 222
	B9BC AA	47548 170	BF2E AA	48942 170
Start of Data.....	B908 D5	47368 213	B893 D5	47351 213
	B912 AA	47378 170	B898 AA	47256 170
	B91D AD	47389 173	B89D AD	47261 173
End of Data.....	B956 DE	47446 222	B8DE DE	47326 222
	B960 AA	47456 170	B8E3 AA	47331 170
Sync byte used during INITialization			BF38 FF	
Sync byte written before the Address Mark			BF73 FF	
Sync byte written before the Data Mark			B87E FF	

DOS 3.3

	Read Locations HEX	Read Locations DECIMAL	Write Locations HEX	Write Locations DECIMAL
Start of Address....	B955 D5	47445 213	BC7A D5	48250 213
	B95F AA	47455 170	BC7F AA	48255 170
	B96A 96	47466 150	BC84 96	48260 150
End of Address.....	B991 DE	47505 222	BCAE DE	48300 222
	B99B AA	47515 170	BCB3 AA	48307 170
Start of Data.....	B8E7 D5	47335 213	B853 D5	47187 213
	B8F1 AA	47345 170	B858 AA	47192 170
	B8FC AD	47356 173	B85D AD	47197 173
End of Data.....	B935 DE	47413 222	B89E DE	47262 222
	B93F AA	47423 170	B8A3 AA	47267 170
Sync byte written before the Address Mark			BC60 FF	
Sync byte written before the Data Mark			B83E FF	

DOS 3.2 Legal Bytes

HEX	DEC	HEX	DEC	HEX	DEC
AA	170	BF	191	EE	238
AB	171	D5	213	EF	239
AD	173	D6	214	F5	245
AE	174	D7	215	F6	246
AF	175	DA	218	F7	247
B5	181	DB	219	FA	250
B6	182	DD	221	FB	251
B7	183	DE	222	FD	253
BA	186	DF	223	FE	254
BB	187	EA	234	FF	255
BD	189	EB	235		
BE	190	ED	237		

DOS 3.3 Legal Bytes

HEX	DEC	HEX	DEC	HEX	DEC
96	150	BA	186	E6	230
97	151	BB	187	E7	231
9A	154	BC	188	E9	233
9B	155	BD	189	EA	234
9D	157	BE	190	EB	235
9E	158	BF	191	EC	236
9F	159	CB	203	ED	237
A6	166	CD	205	EE	238
A7	167	CE	206	EF	239
AA	170	CF	207	F2	242
AB	171	D3	211	F3	243
AC	172	D5	213	F4	244
AD	173	D6	214	F5	245
AE	174	D7	215	F6	246
AF	175	D9	217	F7	247
B2	178	DA	218	F9	249
B3	179	DB	219	FA	250
B4	180	DC	220	FB	251
B5	181	DD	221	FC	252
B6	182	DE	222	FD	253
B7	183	DF	223	FE	254
B9	185	E5	229	FF	255

Deprotecting disks with

Super IOB

By Ray Darrah

Requirements:

An Apple][plus
Disks that need to be modified

As dedicated Hardcore COMPUTIST readers will recall, the IOB program is a simple BASIC program that performs softkeys. IOB stands for Input Output control-Block. It is a list of parameters used by the Read Write Track Sector (RWTS) subroutine.

In the course of time, HARDCORE COMPUTING (old series) and HARDCORE COMPUTIST have published several IOB programs (or IOB modifications). These were useful not only for copying different types of disks but for configuring the program to different machines.

Presented here, is an advanced version of the original IOB program. We're calling it "Super IOB." Included are the most useful subroutines from all the previous IOB programs. Here are some of the new features:

1) The controller isn't spread throughout the program.
2) Half tracks can be accessed.
3) Super IOB is self-configuring.
4) Incorrectly numbered tracks can be copied.
5) The controller performs sector modifications DURING the copy process.
6) A range of seven tracks are read at one time to cut down the disk swaps on single drive systems.
7) Super IOB can do everything MUFFIN PLUS and DEMUFFIN PLUS can.
8) Automatic error trapping is now included.

Using the Super IOB Program

Start by entering the Applesoft listing, then

SAVE SUPER IOB

Next, enter the hexdump and

BSAVE IOB.OBJ0,A$300,L$5C

A third file is required in order to copy disks that have been protected with a 13 sector format.

RWTS.13

To read the protected DOS 3.2 disks, Super IOB uses an image of the 3.2 RWTS. By performing a swap of the image with the RWTS currently in memory, diskettes with different formats can be accessed.

Use BOOT13 from the system master disk to get DOS 3.2 into your 3.3 machine.

Once DOS 3.2 is booted up, all you have to do is BSAVE the RWTS.

BSAVE RWTS.13,A$B800,L$800

What it Does

Super IOB de-protects disks by pushing the RWTS to its upper most limits. Because of this, it only works on disks with sectors somewhat resembling normal DOS. Before a disk can be "Softkeyed", the protection scheme must be determined. The easiest way to do this is to use a program (like "The CIA," 'Bag of Tricks" or "DiskView") which allow you to discover the difference between normal sectors and the ones on the intended disk.

Once the protection has been discovered, all that needs to be done is the insertion of a controller program (lines 1000 through 9999) into Super IOB. Here is a list of the protection schemes Super IOB was designed to Softkey:

1) Altered data, address, prologue, or epilogue marks.
2) Strangely numbered sectors or tracks.
3) Modified RWTS (with same entry conditions).
4) Half tracks for any of the above.
5) Thirteen or 16 sector format for any of the above.

The following is a brief description of each protection scheme and how it relates to Super IOB:

Altered marks

A technique used on a lot of the earlier disks is DOS mark alterations. DOS puts certain reserved bytes on the disk (during INITalization) so it can tell where a sector (and other valuable information) begins. For example, a normal 16-sector disk has the bytes: D5 AA AD, designating the start of the data field which contains the 256 bytes of data in encoded form. When a standard RWTS tries to find a sector, it looks for these marks. If they are not found (either because they don't exist or they have been changed to something else) DOS returns with the dreaded I/O ERROR.

The sequences of the four reserved-byte marks (start of address, end of address, start of data, end of data) are handled by subroutines in Super IOB. These subroutines simply change the marks the RWTS looks for or modify the RWTS so that it doesn't look for them at all (depending upon the mark).

Strangely Numbered Sectors

Sometimes the numbers on the disk which tell the RWTS what sector is currently passing under the read/write head are tampered with. These disks are easily softkeyed with Super IOB. The controller simply reads, using the strange sector numbers.

This works because the RWTS compares the sector number found on the disk with the one the controller is looking for (even if it is higher than 15). Later, when writing, standard sector numbers are used. Thus de-protecting the disk!

Modified RWTS

Often, the disk-protectors will rearrange and/or modify the standard RWTS subroutine. When this happens, all one has to do is make a controller program which reads, using the strange RWTS, then swaps with a normal RWTS and writes the information back out.

Since the RWTS of a protected disk will be modified to read any altered DOS marks, this is a good method to use if you are unable to determine what they have been changed to.

Anatomy of a controller

Before we attempt to write a controller, let's look at the format of a controller. Here is an explanation of the subroutines (and sub-programs) in the Super IOB program that are at the controller's disposal.

Start up

Lines 10-60

The first few lines indentify the program. Line 60, however, sets HIMEM and LOMEM so that they fit the memory usage requirements (see memory map, following). It then goes to "CONFIGURATION TIME."

Initial IOB setup

Line 80

This subroutine is normally GOSUBed via "TOGGLE READ / WRITE." Its purpose is to reset the buffer page and set the drive number, slot number and volume number to the disk to be accessed next.

R/W sector

Line 100-110

This subroutine is GOSUBed directly from the controller. It reads or writes (depending upon CD) at the specified track and sector.

Move S phases

Lines 130-140

Moves the disk drive head by the number of phases specified by S; one phase equals one half-track. It is capable of moving in either direction up to 128 phases (or 64 tracks). When moving the head, this routine doesn't let the RWTS know that the head has been moved. Therefore, this subroutine makes it possible to copy disks that have track mismarkings. Care should be taken when moving a great number of phases that PH + S isn't greater than 255 or less than 0, otherwise an error will occur.

Ignore chechsums & end marks

Line 170 (16 sector RWTS)
Line 270 (13 sector RWTS)

These routines do a few POKEs into their corresponding RWTS. The final result is that the RWTS no longer looks for epilogue marks or checksums when searching for a sector.

Altered address marks

Line 190 (16 sector RWTS)
Line 290 (13 sector RWTS)

These modify the RWTS (via POKE) so that it looks for a different sequence of address prologue marks. The decimal values of the marks to look for should be stored in DATA statements in the "DATA FOR MARKS" area.

Altered data marks

Line 210 (16 sector RWTS)
Line 310 (13 sector RWTS)

These are the same as the previous subroutine except for DATA prologue marks.

Normalizer

Lines 230-240 (16 sector RWTS)
Lines 330-340 (13 sector RWTS)

This restores the values in the RWTS subroutine that are messed up by the three previous routines. This routine should be called just before writing, when using only one RWTS (assuming of course that one of the previous routines was called before reading).

Exchange RWTS

Line 360

This is the standard swap RWTSs routine. It exchanges the RWTS at $1900 with the one at $B800, which is the normal residing place for an RWTS. To tell the swap routine, (which is invoked by a CALL 832) what to exchange, a few POKEs must be executed. They are:

POKE 253, start of first location
POKE 255, start of second location
POKE 224, number of pages (a standard RWTS is eight pages long)

Format disk

Lines 380-410

Formats the target disk. It is meant to be used before the Softkey operation begins (and is GOSUBed by "CONFIGURATION TIME") but can be called by the controller should the need arise.

Print track & sector

Line 430

This is the subroutine that puts the current track and sector number at the top of the screen during the softkey operation.

Center message

Line 450

Centers a message (contained in A$) at the current VTAB position and RETURNs.

Print message and wait

Line 470

This routine uses "CENTER MESSAGE" to print the intended message at a VTAB of 11 and then it prints "PRESS ANY KEY TO CONTINUE." After this, it waits for a key to be pressed and RETURNs.

Toggle Read/Write

Lines 490-530

This routine toggles the state of CD (from ReaD to WRite and vice versa) and prints the current mode in flashing letters at the very top of the screen. In addition, if the user has only one drive, it asks him to swap disks. It then exits via "INITIAL IOB SETUP." Thus making the sector buffer ready for the next operation.

Controller

Lines 1000-9999

These are the line numbers set aside for the controller. This area should have all of the controller and subroutines (sector edits and the like). Before using this, please see the memory map that follows.

Configuration time

Lines 10000-10090

This routine asks the user which slots and drive numbers to use for the various disks. It also formats the target disk if the user so desires.

Get slot and drive#

Lines 10110-10130

Used by "CONFIGURATION TIME" to get SLot and DriVe information.

Get a key

Lines 10150-10170

Used by "GET SLOT AND DRIVE#" to wait for the appropriate drive or slot number to be typed.

Disk error

Lines 10190-10270

This is the normal error-trapping routine. If a disk error occurs, this routine will print the error message, otherwise, it will assume the error is in the controller and the program will crash (CALL 834).

Data for marks

Lines 62010-63999

These line numbers should contain the appropriate data (if any) required for any altered mark routine.

Note: In the above line number description, line numbers consisting of REMs have ommitted. They may be excluded (although it isn't recommended) when typing the program in.

Now that you have an idea of the subroutines, note how the following variables relate to them. While examining this table, it would be a good idea to observe the BASIC that makes up the previously listed subroutines. This will give you a good idea of how things are accomplished in Super IOB.

A - general temporary usage, scrambled by "MOVE S PHASES."

A$ - holds message to pass to the user via "CENTER MESSAGE" and "PRINT MESSAGE AND WAIT," scrambled by "TOGGLE READ / WRITE."

A1,A2,A3 - scrambled by any "ALTERED ADDRESS MARKS" or "ALTERED DATA MARKS" routine, they are READ from DATA statements and POKEd into the appropriate RWTS subroutine to change the marks it looks for.

B$ - altered only during configuration.

BF - buffer full, holds the status of the sector buffer, set to 1 if the buffer is either full or empty and to 0 if neither; changed only by "R/W SECTOR."

BUF - buffer location, holds the address where the RWTS is expecting to find the page number of the sector; used by "INITIAL IOB SETUP" and "R/W SECTOR." A (PEEK(BUF)-1)*256 will return the address of byte zero in the last read sector.

CD - command code, used by the controller and "TOGGLE READ / WRITE," holds the current RWTS command code; only POKEd in by "INITIAL IOB SETUP" (see RD, WR, and INIT)

CMD - Command code location, holds the address where the RWTS is expecting to find the previously stated command code; used by "INITIAL IOB SETUP." A POKE CMD,CD will change the IOB command.

D1 - drive 1, set during configuration to the drive number of the source drive; used by "TOGGLE READ/WRITE".

D2 - drive 2, same as above except for target drive.

DOS - Disk Operating System, the number of sectors to read or write; initialized to 16.

DRV - drive location, holds the address where the RWTS is expecting to find the drive number of the drive to be accessed; used by "INITIAL IOB SETUP" to change the IOB drive number. A PEEK(DRV) will return the drive last accessed.

DV - current drive, used by "INITIAL IOB SETUP," 'TOGGLE READ/WRITE" and "MOVE S PHASES;" holds the drive number of the drive to be accessed next.

ERR - error code, used by "DISK ERROR" to determine the error that has just occured.

INIT - initialize command code, a CD = INIT will set the command code to format the diskette.

IO - Input/Output location, normally holds a 768 (set during configuration); CALLed by "R/W SECTOR" to induce the RWTS subroutine. To use a relocated RWTS, the controller must have a IO = IO + 42 statement.

MB - maximum buffer page, holds the last page of memory for the sector buffer; used by "R/W SECTOR," initialized (during configuration) to 151 and should be changed to 130 only when a 13-sector disk is read or written.

OVL - old volume location, a PEEK(OVL) will return the volume number of the previously accessed (via "R/W SECTOR") diskette.

PH - current phase, if "MOVE S PHASES" is referenced (by the controller), this variable must contain the disk arms' current phase number (PH = 2 * TK).

RD - read command code, a CD = RD will set the command to read the disk.

S - step, used to tell "MOVE S PHASES" how may phases to step through (-120 to 120).

S1 - slot 1, set to the slot number of the source drive during configuration; used by "TOGGLE READ/WRITE."

S2 - slot 2, same as above except for target drive.

SCT - sector number location, holds the address where the RWTS is expecting to find the sector to be accessed; used by "R/W SECTOR" to tell the RWTS which sector is to be read or written. A PEEK(SCT) will return the last accessed sector number.

SLT - slot number location, holds the address where the RWTS is expecting to find the slot number of the disk to accessed next; used by "INITIAL IOB SETUP." A PEEK(SLT) will return the last accessed disk's slot number.

SO - slot number, used by "TOGGLE READ/WRITE" and "INITIAL IOB SETUP;", holds the slot number of the disk to be accessed next.

ST - sector number, used by the controller to tell "R/W SECTOR" what sector number is to be read or written next.

TK - track number, used by the controller to tell "R/W SECTOR" what track is to be accessed next.

TRK - track number location, holds the memory location where the RWTS is expecting to find the track to be accessed. A PEEK(TRK) will return the last accessed track number.

VL - volume number, used by the controller to tell "TOGGLE READ / WRITE" (which passes it to "INITIAL IOB SETUP") the volume number of the disk to be accessed next.

VL$ - altered only by "FORMAT DISK."

VOL - volume number location, holds the memory location where the RWTS is expecting to find the volume to be accessed. A PEEK(VOL) will return the volume number last used by the controller.

WR - write command code. A CD = WR will set the command to write.

Memory Usage

Before actually looking at some controllers, let's say a few words about memory usage.

Following, is a memory allocation table for the various parts of Super IOB. It is extremely important to stay within the boundaries when writing a controller. Otherwise, horrible things might happen (the least of which would be the production of an incorrect copy).

$0800.$18FF (2048-6399) intended for the Applesoft part of Super IOB.
$1900.$20FF (6400-8447) space allocated for a relocated RTWS (RWTS.13 or RWTS.16)
$2100.$26FF (8448-9983) BASIC variable space.
$2700.$96FF (9984-38655) used for the sector buffer

First, notice the amount of space available for the BASIC program. The Super IOB program as listed (with all REMs), ends about 1200 bytes short of the final designated location. This means that the controller (and all DATA statements) must fit into this 1K area. In view of the space requirement, the end of program should be checked by typing:
PRINT PEEK(175) + PEEK(176) * 256
before a new controller is used.

If it has exceeded the 6399 limit, I suggest DELeting all subroutines not referenced by the controller and all REM lines until it fits within the allocated space.

However, if the program does NOT use a relocated RWTS, then the extra 2K allocated for an RWTS can be used for the BASIC. In this situation, the end of the program should only be checked with very long controllers, since 3K ought to be enough for any softkey operation.

Secondly, observe the 1534 bytes for variables. This should be enough space for the simple softkey procedure. It is impossible to allocate more memory for variables and use a relocated RWTS file. If you find that you need more memory and the program does not use RWTS.16 or RWTS.13, then the LOMEM: 8448 statement in line 60 may be ommitted. This will allocate what isn't used (by the BASIC program) of the 2K area reserved for the relocated RWTS as variable space.

Never omit the "HIMEM:" statement! This could cause variables to overflow into the sector buffer, thus making a faulty copy.

Finally, with all this new knowledge we are ready to scrutinize some sample controller programs. Keep in mind that protection schemes can be used with one another. Therefore, a more sophisticated controller for Super IOB will probably be required for most softkeys. Even so, devoloping new controllers isn't difficult.

Standard Controller

```
1000  REM STANDARD CONTROLLER
1010 TK = 0:ST = 0:LT = 35:CD = WR
1020 T1 = TK: GOSUB 490
1030  GOSUB 430: GOSUB 100:ST = S
      T + 1: IF ST < DOS THEN 1030
1040  IF BF THEN 1060
1050 ST = 0:TK = TK + 1: IF TK <
      LT THEN 1030
1060  GOSUB 490:TK = T1:ST = 0
1070  GOSUB 430: GOSUB 100:ST = S
      T + 1: IF ST < DOS THEN 1070
1080 ST = 0:TK = TK + 1: IF BF =
      0 AND TK < LT THEN 1070
1090  IF TK < LT THEN 1020
1100  HOME : PRINT : PRINT "DONE
      WITH COPY": END
```

Here is how the standard controller works:

Unique Variables

The following variables are used by the controller exclusively. Other variables used by the controller are for interaction with various subroutines in Super IOB.

LT - this variable holds the last track to be accessed (it is the last track plus one). For example, if line 1010 were to have an LT = 15 (instead of LT = 35) then it would only copy tracks 0-14.

T1 - holds the track number (TK) for the transition of read to write and vice versa.

Line explanation

1000 - identifies controller.

1010 - initializes variables.
TK = 0 - sets the starting track to zero.
ST = 0 - sets the starting sector to zero.
LT = 35 - sets the last track to 34.
CD = WR - sets command code to write (WR).

1020 - The read routine. It begins by saving the current track number and, then, gets the source disk.
1030 - prints the current track and sector number, reads in the sector and increments the sector number. If it is less than DOS (in this case 16) then it reads another sector.
1040 - if the sector buffer is full, it goes to the write routine.
1050 - resets the sector number to zero and increments the track number. If it is not past the last track, it reads the new track.
1060 - this is the beginning of the write routine. It gets the write drive and starts at the previously saved track (T1), sector zero.
1070 - prints the current track and sector number, writes the sector to the disk and increments the sector number. If it is not finished with this track, it writes another sector.
1080 - resets the sector number and increments the track number. If the sector buffer isn't empty and it's not past the last track, it writes another track.
1090 - if it is not done duplicating the disk (i.e, not past last track), it reads some more tracks.
1100 - tells user that everything is OK and that the disk is copied.

Even though this controller only copies normal DOS 3.3 disks, I recommend saving it anyway. This controller is the basic (pun intended) building block for more complex controllers.

Load the original Super IOB program

LOAD SUPER IOB

Type in the controller listed above.
Save this new program

SAVE IOB.STANDARD.CON

You now have the capability (I'm sure you did before) to copy a regular diskette. Because you probably don't think this is so exciting, we'll move on to the de-protection of Castle Wolfenstein. I chose this game because its controller is a simple example of what a few modifications to the standard controller can accomplish.

Castle Wolfenstein Controller

```
1000  REM CASTLE WOLFENSTEIN CONT
      ROLLER
1010 TK = 3:ST = 0:LT = 35:MB = 1
      30:CD = WR:DOS = 13
1020 T1 = TK: GOSUB 490: GOSUB 36
      0
1030  GOSUB 430: GOSUB 100:ST = S
      T + 2: IF ST < DOS * 2 THEN
      1030
1040  IF BF THEN 1060
1050 ST = 0:TK = TK + 1: IF TK <
      LT THEN 1030
1060  GOSUB 490:TK = T1:ST = 0: GOSUB
      360
1070  GOSUB 430: GOSUB 100:ST = S
      T + 1: IF ST < DOS THEN 1070
1080 ST = 0:TK = TK + 1: IF BF =
      0 AND TK < LT THEN 1070
1090  IF TK < LT THEN 1020
1100  HOME : PRINT "EVERYTHING O.
      K. NO DOS ON COPY": END
10010  PRINT  CHR$ (4)"BLOAD RWTS
      .13,A$1900"
```

Castle Wolfenstein uses "Strangely Numbered Sectors" as its protection scheme. Luckily, they aren't so strange that a complex algorithm is needed to calculate the next number. Instead, they are merely even-numbered DOS 3.2 sectors (0-24).

When 13-sector DOS gets these sector numbers, it doesn't accept them and returns with I/O error. But the 13-sector RWTS doesn't care about the actual number on the sector, as long as it matches up with the sector number you want to access. Thus, all one has to do is read with the strange sector numbers and write with the normal ones.

Here is a line-by-line explanation of the differences that make this controller sucessful:

1000 - identifies controller.
1010 - start at track three (bypass DOS tracks) and set MB and DOS to their 13-sector values.
1020 - since we want to use RWTS.13 to read with, swap it in.
1030 - counts from 0 to 24 by two's.
1060 - swaps the normal RWTS back into its original location for the write ahead.
1100 - tells the user that the copy has no DOS on it.
10010 - BLOADs the 13-sector RWTS at $1900.

As noted in line 10010, once the copy has been made there will be no DOS on the de-protected version. This isn't a problem as long as you don't boot with it.

Super IOB BASIC program

```
10 REM ************************
20 REM **       SUPER IOB       **
30 REM **    BY RAY DARRAH      **
40 REM ************************
50 REM SET HIMEM BELOW BUFFER AND
    SET LOMEM ABOVE THE BLOADED RWTS
60 LOMEM: 8448: HIMEM: 9983: GOTO
    10010
70 REM INITIAL IOB SETUP
80 POKE BUF,39: POKE DRV,DV: POKE
    VOL,VL: POKE SLT,SO * 16: RETURN
90 REM R/W SECTOR
100 BF = 0: POKE TRK,TK: POKE
    SCT,ST: POKE CMD,CD: CALL IO:
    POKE BUF, PEEK (BUF) + 1: IF
    PEEK (BUF) = > MB THEN BF = 1
110 RETURN
120 REM MOVE S PHASES
130 POKE 49289 + SO * 16 + DV,0:
    POKE 49289 + SO * 16,0: A = PH -
    INT (PH / 4) * 4: POKE 1144,128
    + A: POKE 811,128 + S + A: POKE
    813,SO * 16: CALL 810: POKE
    49288 + SO * 16,0: PH = PH + S:
    IF PH < 0 THEN PH = 0
140 RETURN
150 REM 16 SECTOR RWTS ALTERATIONS
160 REM IGNORE CHKSUM & END MARKS
170 POKE 47405,24: POKE 47406,96:
    POKE 47497,24: POKE 47498,96:
    RETURN
180 REM ALTERED ADDRESS MARKS
190 READ A1,A2,A3: POKE 47445,A1:
    POKE 47455,A2: POKE 47466,A3:
    RETURN
200 REM ALTERED DATA MARKS
210 READ A1,A2,A3: POKE 47335,A1:
    POKE 47345,A2: POKE 47356,A3:
    RETURN
220 REM NORMALIZER
230 POKE 47405,208: POKE 47406,19:
    POKE 47497,208: POKE 47498,183:
    POKE 47445,213
240 POKE 47455,170: POKE 47466,150:
    POKE 47335,213: POKE 47345,170:
    POKE 47356,173: RETURN
250 REM 13 SECTOR RWTS ALTERATIONS
260 REM IGNORE CHKSUM & END MARKS
270 POKE 47530,24: POKE 47531,96:
    POKE 47438,24: POKE 47439,96:
    RETURN
280 REM ALTERED ADDRESS MARKS
290 READ A1,A2,A3: POKE 47478,A1:
    POKE 47488,A2: POKE 47499,A3:
    RETURN
300 REM ALTERED DATA MARKS
310 READ A1,A2,A3: POKE 47368,A1:
    POKE 47378,A2: POKE 47389,A3:
    RETURN
320 REM NORMALIZER
330 POKE 47530,208: POKE 47531,183:
    POKE 47438,208: POKE 47439,19:
    POKE 47478,213
340 POKE 47488,170: POKE 47499,181:
    POKE 47368,213: POKE 47378,170:
    POKE 47389,173: RETURN
350 REM SWAP RWTS AT $1900 WITH THE
    ONE AT $B800
360 POKE 253,25: POKE 255,184: POKE
    224,8: CALL 832: RETURN
370 REM FORMAT DISK
380 A$ = "VOLUME NUMBER FOR
    COPY=>254": HOME: GOSUB 450:
    HTAB32: INPUT "";VL$: VL = VAL
    (VL$): IF VL$ = "" THEN VL = 254
390 IF VL > 255 OR VL < 0 THEN 380
400 POKE CMD,INIT: SO = S2: DV = D2:
    A$ = "INSERT BLANK DISK IN SLOT
    " + STR$ (S2) + ", DRIVE " + STR$
    (D2): GOSUB 470
410 GOSUB 80: HOME: A$ =
    "FORMATING": FLASH: GOSUB 450:
    NORMAL: CALL IO: VL = 0: RETURN
420 REM PRINT TRACK & SECTOR#
430 VTAB 3: HTAB 10: PRINT
    "TRACK=>"TK SPC( 2)"SECTOR=>"ST
    SPC(2): RETURN
440 REM CENTER MESSAGE
450 HTAB 21 - LEN (A$) / 2:
    PRINTA$;: RETURN
460 REM PRINT MESSAGE AND WAIT
470 HOME: VTAB 11: GOSUB 450:
    VTAB13: A$ = "PRESS ANY KEY TO
    CONTINUE": GOSUB 450: WAIT
    -16384,128: GET A$: RETURN
480 REM TOGGLE READ/WRITE
490 CD = (CD = 1) + 1: IF CD = RD
```

```
THEN A$ = "INSERT SOURCE DISK.": SO
    = S1: DV = D1: GOTO 510
500 A$ = "INSERT TARGET DISK.": SO =
    S2: DV = D2
510 IF D1 = D2 AND S1 = S2 THEN GOSUB
    470: HOME
520 VTAB 1: HTAB 1: PRINT SPC(39);:
    FLASH: A$ = "READING": IF CD =
    WR THEN A$ = "WRITING"
530 GOSUB 450: NORMAL: GOTO 80
10000 REM CONFIGURATION TIME
10010 REM BLOAD RWTS HERE
10020 IF PEEK (768) * PEEK (769) =
    507 THEN 10060
10030 HOME: A$ = "* SUPER IOB *":
    GOSUB 450: PRINT: PRINT: A$ =
    "CREATED BY RAY DARRAH": GOSUB
    450
10040 VTAB 10: A$ = "INSERT SUPER
    IOB DISK": GOSUB 450: PRINT:
    PRINT: PRINT: A$ = "PRESS ANY
    KEY TO CONTINUE": GOSUB450: WAIT
    - 16384,128: GET A$
10050 PRINT: PRINT CHR$ (4)"BLOAD
    IOB.OBJ0,A$300"
10060 TK = ST = VL = CD = DV = SO:RD
    = 1:WR = 2:INIT = 4: ONERR GOTO
    10220
10070 IO = 768: SLT = 779: DRV =
    780: VOL = 781: TRK = 782: SCT
    =783: BUF = 787: CMD = 790: OVL
    =792
10080 HOME: DOS = 16:MB = 151
10090 VTAB 8: PRINT: A$ =
    "ORIGINAL": S2 = 6: D2 = 1:
    GOSUB 10140: S1 = S2: D1 = D2
10100 PRINT: PRINT: PRINT: D2 =(D2 =
    1) + 1: A$ = "DUPLICATE
": GOSUB 10140
10110 A$ = "FORMAT BACK UP FIRST? N"
    + CHR$ (8): HOME: VTAB12: GOSUB
    450: GET A$: IF A$ = "Y" THEN
    GOSUB 380
10120 HOME: A$ = "INSERT DISKS IN
    PROPER DRIVES.": GOSUB 470:
    HOME: GOTO 1000
10130 REM GET SLOT AND DRIVE#
10140 GOSUB 450: PRINT: PRINT: PRINT
    TAB( 10)"SLOT=>"S2
    SPC(8)"DRIVE=>"D2;
10150 HTAB 16: B$ = "7": GOSUB
    10180: S2 = VAL (A$)
10160 HTAB 32: B$ = "2": GOSUB
    10180: D2 = VAL (A$): RETURN
10170 REM GET A KEY
10180 GET A$: IF (A$ < "1" OR A$ >
    B$) AND A$ < > CHR$ (13) THEN
    10180
10190 IF A$ = CHR$ (13) THEN A$ =
    CHR$ ( PEEK ( PEEK (40) + PEEK
    (41) * 256 + PEEK (36)) - 128)
10200 PRINT A$;: RETURN
10210 REM DISK ERROR
10220 ERR = PEEK (222): IF ERR >15
    AND ERR < 254 THEN POKE216,0:
    CALL 822
10230 IF ERR = 254 THEN PRINT "TYPE
    AGAIN PLEASE:": PRINT: RESUME
10240 IF ERR = 255 THEN STOP
10250 IF ERR = 0 THEN A$ =
    "INITIALIZATION ERROR"
10260 IF ERR = 1 THEN A$ = "WRITE
    PROTECTED"
10270 IF ERR = 2 THEN A$ = "VOLUME
    MISMATCH ERROR"
10280 IF ERR = 4 THEN A$ = "DRIVE
    ERROR"
10290 IF ERR = 8 THEN A$ = "READ
    ERROR"
10300 VTAB 20: GOSUB 450: PRINT CHR$
    (7): END
62000 REM DATA FOR MARKS
```

Super IOB Checksums

```
10    - $BADD
20    - $9B13
30    - $4D3B
40    - $AD92
50    - $C899
60    - $1FBA
70    - $0061
80    - $835F
90    - $E171
100   - $AD0E
110   - $57B6
120   - $8472
130   - $617E
140   - $0F1F
150   - $F1B3
160   - $C59A
170   - $6DEC
180   - $56EA
190   - $D2AC
200   - $1EEF
210   - $C7D5
220   - $7B7E
230   - $F7E4
240   - $596A
250   - $50B9
260   - $7DB0
270   - $AD47
280   - $E373
290   - $4B8B
300   - $FFE7
310   - $4DD1
320   - $4DA3
330   - $C76F
340   - $01F0
350   - $F0AE
360   - $5452
370   - $C2A5
380   - $8A57
390   - $65AE
400   - $15FA
410   - $9A03
420   - $FF36
430   - $713A
440   - $0A35
450   - $76B5
460   - $51E2
470   - $CCA2
480   - $7AD0
490   - $EEB8
500   - $3A54
510   - $5FC8
520   - $D7BE
530   - $A4CF
10000 - $7C7C
10010 - $EBB3
10020 - $255F
10030 - $97DE
10040 - $7273
10050 - $5F85
10060 - $3118
10070 - $95D0
10080 - $B791
10090 - $399B
10100 - $1951
10110 - $E187
10120 - $A960
10130 - $0F5E
10140 - $A8AD
10150 - $3AAD
10160 - $7CEA
10170 - $A58F
10180 - $28E4
10190 - $2E00
10200 - $64EA
10210 - $056B
10220 - $7BA4
10230 - $DB00
10240 - $DB1F
10250 - $03D9
10260 - $7F7D
10270 - $214C
10280 - $CDCD
10290 - $EE7B
10300 - $B2DB
62000 - $C89F
```

Source Code

```
1000 * ------------------------------
1010 * Super IOB machine routines
1020 *
1030 * BY RAY DARRAH
1040 * ------------------------------
1050
1060 RWTS.B800   .EQ  $03D9        ENTRY POINT T
O RWTS @$B800
1070 INVOKERROR  .EQ  $D412        ROUTINE THAT
CAUSES BASIC TO DO THE ERROR CONTAINED IN X
1080 RWTS.1900   .EQ  $1E00        ENTRY POINT T
O THE RWTS AT $1900
1090 SEEKABS     .EQ  $B9A0        ENTRY POINT T
O THE SEEKABS ROUTINE AT $B800
1100 BAS.ERR     .EQ  222          ;BASIC ON ERR
 ERROR CODE
1110 SWFRM       .EQ  $FC          ;EXCHANGE FR
OM PARAMTER
1120 SWTO        .EQ  $FE          ;EXCHANGE RW
TS 'TO' PARAMETER
1130 PAGES       .EQ  $E0          ;NUMBER OF PA
GES OF MEMORY TO EXCHANGE
1140             .OR  $0300        STARTS AT PAG
E THREE
1150             .TF  IOB.OBJ0
1160
1170 * ------------------------------
1180 *              CALL RWTS              *
1190 * ------------------------------
1200
1210 IO          LDA /TABLETYP     ENTRY POINT
FOR CALING THE RWTS THROUGH BASIC
1220             LDY #TABLETYP     A,Y POINT TO
 THE IOB TABLE
1230             JSR RWTS.B800     GO TO THE RW
TS AT $B800
1240             BCS DOS.ERR       IF THE CARRY
SET THEN CAUSE BASIC ERROR
1250             RTS               OTHERWISE, AL
L IS WELL SO RETURN
1260 TABLETYP    .HS  01           TYPE OF TABLE
(1=IOB)
1270 SLT         .HS  60           SLOT
1280 DRV         .HS  01           DRIVE
1290 VOL         .HS  00           VOLUME
1300 TRK         .HS  00           TRACK
1310 SCT         .HS  00           SECTOR
1320 DCTPTR      .DA  DCT          POINTER TO TH
E DEVICE CHARACTERISTICS TABLE
1330 BUFFERLO    .HS  00           ALWAYS MAKE
LSB OF BUFFER POINTER ZERO!
1340 BUF         .HS  27           SECTOR BUFFE
R PAGE POINTER
1350 NOTHING     .HS  00           NOT USED
1360 BYTCOUNT    .HS  00           BYTE COUNT FO
R PARTIAL SECTOR (0=256 BYTES)
1370 CMD         .HS  00           COMMAND COD
E (0=SEEK)
1380 RWTS.ERR    .HS  00           ERROR CODE T
HA THE RWTS.B800 RETURNS WITH
1390 OVL         .HS  00           VOLUME NUMB
ER OF LAST ACCESSED DISK
1400 OLDSLT      .HS  60           SLOT PREVIOUS
LY ACCESSED
1410 OLDDRV      .HS  01           DRIVE PREVIOU
LSY ACCESSED
1420 DCT         .HS  00           DEVICE TYPE O
F DEVICE CHARACTERISTICS TABLE
1430 PHASES      .HS  01           PHASES-1 PER
TRACK, (0 OR 1)
1440 MOTORCNT    .HS  EFD8         MOTOR-ON TIM
E COUNT
1450 DOS.ERR     LDA RWTS.ERR      DOS HAS HAD A
N ERROR, GET THE ERROR CODE
1460             LSR               DIVIDE IT BY 16
1470             LSR
1480             LSR
1490             LSR
1500             TAX               TRANSFER IT T
O X SO BASIC WLL INDUCE THE FALSE ERROR CODE
1510             JMP INVOKERROR    CAUSE A BAS
IC ERROR
1520
```

```
1530 * ------------------------------
1540 *          MOVE THE DISK ARM     *
1550 * ------------------------------
1560
1570 MOVPHASES  LDA #$00        ROUTINE TO SE
T UP THE REGISTERS BEFORE CALLING SEEKABS
1580            LDX #$00        X AND A HAVE
DUMMY NUMBERS THAT WILL BE POKED INTO BY
 "MOVE S PHASES"
1590            JMP SEEKABS
1600
1610 * ------------------------------
1620 *       CAUSE ERROR IN CONTROLLER  *
1630 * ------------------------------
1640
1650 BASICERR   LDX BAS.ERR     BASIC HAS MAD
E AN ERROR SO CAUSE THE ERROR NUMBER AT 222
1660            JMP INVOKERROR
1670 * ------------------------------
1680 *          POP OFF RETURN        *
1690 * ------------------------------
1700 POP        PLA             ROUTINE TO PO
P OFF ONE RETURN (BASIC) ADDRESS
1710            TAY
1720            PLA
1730            LDX BAS.ERR+1   GET WHAT THE
 STACK WOULD BE IF THE GOSUB WASN'T THERE
1740            TXS             PUT THAT AS
THE STACK POINTER
1750            PHA
1760            TYA             RESTORE THE
LAST RETURN ADDRESS
1770            PHA
1780            RTS
1790
1800 * ------------------------------
1810 *          EXCHANGE RWTS's       *
1820 * ------------------------------
1830
1840            LDY #0          ;ZERO LSB's
1850            STY SWFRM       ;AND HAVE Y
AT ZERO FOR START
1860            STY SWTO
1870 MOVE.PAGE  LDA (SWFRM),Y   ;GET A BYTE
1880            PHA             ;AND SAVE IT
1890            LDA (SWTO),Y    ;GET THE BYTE
 WHERE THE SAVED ONE GOES
1900            STA (SWFRM),Y   ;AND STORE I
T WHERE THE SAVED ONE WAS
1910            PLA             ;GET THE SAVE
D BYTE
1920            STA (SWTO),Y    ;AND STORE IT
 WHERE IT GOES
1930            INY             ;DONE WITH A
PAGE
1940            BNE MOVE.PAGE   ;NO KEEP WOR
KING ON IT
1950            INC SWFRM+1     ;GET NEXT MSB
1960            INC SWTO+1
1970            DEC PAGES       ;DECREMENT T
HE NUMBER OF PAGES TO MOVE
1980            BNE MOVE.PAGE   ;IF NOT DONE
, MOVE ANOTHER PAGE
1990            RTS             ;FINISHED, RTS
```

Super IOB hexdump

```
0300: A9 03 A0 0A 20 D9 03 B0 $BD35
0308: 16 60 01 60 01 00 00 00 $9CF5
0310: 1B 03 00 27 00 00 00 00 $4320
0318: 00 60 01 00 01 EF D8 AD $55A7
0320: 17 03 4A 4A 4A 4A AA 4C $B42B
0328: 12 D4 A9 00 A2 00 4C A0 $8038
0330: B9 A6 DE 4C 12 D4 68 A8 $6E1C
0338: 68 A6 DF 9A 48 98 48 60 $FDD9
0340: A0 00 84 FC 84 FE B1 FC $3777
0348: 48 B1 FE 91 FC 68 91 FE $AAB9

0350: C8 D0 F3 E6 FD E6 FF C6 $921F
0358: E0 D0 EB 60             $3160
```

A quick and easy way to

Unlock Hyperspace Wars

By Robb Canfield

Requirements:

48K A][+ with Applesoft in ROM
MUFFIN
HYPERSPACE WARS
A Blank Diskette

Hyperspace Wars is published by Continental Software (copyright 1980) and consists of two games on a single disk:

1) 48K TREK, a text "Star Trek" type of strategy arcade game.

2) 3-D Space Battle, one of the pioneers in "real-view" space arcades. In this game, the player is shown an "out-the-window" view in which stars move and aliens abound, zooming towards and away from the player.

The following sections will first explain the technique used to unlock Hyperspace Wars, and then provide a step-by-step method that will make it easy for anyone to do.

Hyperspace Wars is on a DOS 3.2 disk. In order to run on DOS 3.3 systems, the game requires either the BASICS disk or the BOOT13 program from the master disk.

The unlocking technique

To discover what locking method was used, Hyperspace Wars was booted and the HELLO program was loaded.

The next step was to enter the monitor and then compare the DOS in memory to a normal 3.2 DOS. To do this, the Hyperspace Wars DOS was moved down to $4800, using the monitor move command:

$4800<B800.BFFFM.

Then a 3.2 disk that had been upgraded to boot on either a 3.2 or 3.3 system was placed in the drive. After the disk had booted, the two DOSs' were compared.

The only differences found were in the address marks and in the translate tables. Hyperspace Wars uses a data mark of D6 instead of the normal DOS 3.2 mark of D5. The translate table had two bytes switched (24 and 60). Another byte was changed from D6 to D5.

It was also necessary to clear the checksum value. Since MUFFIN has a 3.2 image of the DOS Read-Write Track/Sector (RWTS) in it (starting at $1900), all that was needed was to make a few changes and, voila an unlocked (normalized) 3.3 version of the game.

You can do it

An easy guide for normalizing Hyperspace Wars resulted from the methods discussed above.

1) Boot the 3.3 master disk to insure that there is a good 3.3 DOS in memory.

2) Clear the usable memory and make sure you are in Applesoft by typing:

FP

3) Initialize a disk with the HELLO as the HELLO program. Enter:

INIT HELLO

Set this disk aside for use in step 8.

4) BLOAD MUFFIN from your system master disk.

BLOAD MUFFIN

5) Enter the monitor by typing:

CALL -151

6) And make the following changes to MUFFIN (press return after each line):

1A08:D6
1A76:D6
1A63:18
1BD5:60 24
1DA6:D5

The first two changes are to the address marks, and the third change clears the checksum. The last three bytes change the Read-Translate table.

7) From the monitor, run MUFFIN by entering:

803G

8) Use the initialized blank disk from step 3 as the target disk to put your copy of Hyperspace Wars on.

9) When asked for the file name, enter an equal sign (=) and copy over the existing HELLO name.

You now have a normalized copy of Hyperspace Wars.

Taking a peek at

Boot Code Tracing

By Mycroft

Requirements:

Knowledge of machine language
Multi-Disk Catalog by Sensible Software
Initialized blank disk with HELLO program deleted

If you have a little knowledge of machine language programming, and a good measure of perseverance, you can defeat the locking scheme used in a large group of programs and capture them on standard DOS. These are the kinds of programs that boot and run with no subsequent disk access. Most games and many utility and business programs fall into this category.

THE THEORY

No matter what locking scheme is used, the disk must boot on a standard Apple.

If we could somehow step through the boot process, get everything loaded, and then stop just before starting the program, we would be able to save the whole thing and run it under standard DOS.

The Apple boot process starts with boot 0 in the disk controller ROM. This short machine language program BLOADS track 0, sector 0 (containing boot 1 of the disk being booted) at locations $800 through $8FF, and then jumps to $801 to execute boot 1.

Boot 1 reads boot 2, and the process continues through successive boot stages until finally the main program is loaded and run.

ABOUT RESET

The Reset routine is at $FF59 in the monitor ROM. It performs a function similar to pushing the reset button. If called, a Reset cycle is performed and any executing program will be stopped.

GETTING STARTED

To illustrate this unlocking procedure, the program Multi-Disk Catalog III, by Sensible Software, will be used. Even though this program has been around for a while, it is an excellent and useful utility.

Dozens of other programs which use a virtually identical boot sequence were examined. They can be unlocked using this same technique with only a few changes (try one of your own single-load programs if you don't have Multi-Disk Catalog).

Have an initialized slave diskette ready and make sure there is a writeprotect tab on the disk you are trying to unlock.

THE UNLOCKING PROCESS

Turn on your Apple with the disk drive empty, and push reset to stop the drive. This will keep the Apple's memory clear. *NOTE: Commands will be on a separate line and printed exactly as you should enter them. Press the return key after each line. If a command has already been listed, it will be referred to but not listed again.*

Now insert the locked disk and enter the monitor.

CALL-151

MODIFYING BOOT 0

Since boot 0 is in ROM on the controller card, it cannot be directly modified. The solution is to move it to RAM, using the monitor's Memory Move routine so that it can be changed.

The new location should be some place in RAM where the boot will not be overwritten in any of the successive boot stages. Memory just below DOS is usually safe since many locked programs use only slightly modified versions of normal DOS.

Always move the boot 0 code to a page boundary that corresponds to the slot used by your controller card. (IE. for slot 6 - $9600, $8600, $7600, etc. or for slot 5 - $9500, $8500, etc.) The reason for this is the boot 0 code contains a routine which finds the slot where your controller card resides. It does this by calling a return code in the F8 ROM and extracting the return address from the stack to locate the page boundary. The boot 0 code itself is relocatable (will run anywhere in memory).

THE FIRST STEP

Assuming your disk controller card is in slot 6, the code for boot 0 starts at $C600 and extends through $C6FA. Move the boot 0 code from the controller card to page $96 in memory.

9600<C600.C6FFM

Change the exit jump from boot 0 at $96F8 to point to $9801.

96FA:98

Change memory at $9801 to point to the reset routine.

9801:4C 59 FF

Run the boot 0 code.

9600G

The drive will start, and in a second or two there should be a beep. The monitor prompt will appear on the screen. Turn off the drive by typing:

C0E8

This process will be repeated for each successive boot stage. Whenever there is an exit jump to a new code section, put in a jump to reset to stop it. Don't worry about what the code is doing at this point. Look mainly for exit jumps.

THE BOOT 1 CODE

Examine the boot 1 code.

801L

This code relocates itself to memory page two, loads boot 2, and finally jumps to $301 at $841.

Move this code so it can be modified.

9800<800.8FFM

Fix the exit jump to go to $9301.

9843:93

Change memory at $9301 to point to the reset routine.

9301:4C 59 FF

One other byte in boot 1 must be changed so that our modified code is executed properly.

9805:98

Run the code at $9600 again and stop the drive when you get the monitor prompt.

BOOT 2

The next stage of the boot normally starts at $301. Move this code to $9300.

9300<300.3FFM

Take a look at the disassembled listing of the code beginning at $9301.

The exit jump from this stage can be seen at $9343, but it is disguised as an indirect jump through page 0 location $3E.

If you examine the code in this boot stage beginning at $931F, you will find that this indirect jump is used repeatedly to go to $25D, but ultimately the indirect jump address is changed to go to the next boot stage. This change occurs at $933A-$9341.

The final jump is determined by the byte stored in memory location $3CC, which the program increments by 1 before executing the final indirect jump.

Look at the byte at location $3CC.

$3CC

You will find that it contains the value $36 (other programs commonly use $B6). This is the high byte of the jump-to address (the low byte has value $00). The program increments this value by 1, so the final JMP

address is $3700.

The boot should do all the jumps when it is going to $25D, but stop before it makes the final jump to $3700.

AN INDIRECT JUMP

Zero page location $3E contains the value $5D for all the indirect jumps except the final one. We only need to see if this value changes and, if it does, stop the boot. This short subroutine can handle the indirect jump.

9000:A9 5D C5 3E D0 03 4C 5D
9008:02 4C 59 FF

The source code for this routine would look like:

```
9000-A9 5D     LDA  #$5D    Load value.
9002-C5 3E     CMP  $3E     Same?
9004-D0 03     BNE  $9009   No, go RESET
9006-4C 5D 02  JMP  $025D   Yes, go on.
9009-4C 59 FF  JMP  $FF59   Jump RESET.
```

Change the boot code to jump to this subroutine.

9343:4C 00 90

Run the boot 0 code.

BOOT 3

After the beep, stop the drive and examine the code beginning at $3700. The next exit jump is at $3747, and is a jump to $1B03. Change the jump to point to reset.

3747:4C 59 FF

WRITING TO ROM

Now for the sneaky part. The code that was moved to memory page $93 ($9300) was responsible for reading this portion of the boot. But since $3747 was just changed, it must not be overwritten when the boot starts over again. To avoid that happening, change byte $93CC, so that a dummy write is done by letting it 'write" to the ROM (read only memory)!

93CC:D0

Also change the bytes at $9315 and $933E to reference this location instead of $3CC.

9315:93
933E:93

The "write" for the next boot stage will begin at $D000, and is ineffectual except to keep the drive running and in the proper read mode.

Change the subroutine we put in at $9000 to go to the modified next stage.

9009:4C 00 37

Run boot 0 again. When you hear the beep, the drive will stop by itself. You're almost finished.

Start listing the program at $1B03, looking for the next major exit jump. You should find it at $1C25. It is a jump to $1E54. Change $1C25 to point to reset.

1C25:4C 59 FF

Run the code at $1B03.

1B03G

List the code beginning at $1E54. There is an immediate jump to $9D84. List from $9D84.

LANGUAGE CARD?

At $9DE4 and $9DE7 are two indirect jumps, through $9D5E and $9D5C respectively.

A careful examination of the code, beginning at $9D84, reveals that the first indirect jump is taken by systems equipped with language cards (RAM cards), and the second for those without.

No matter, the second indirect address will ultimately be jumped to whichever system you have. To find out what it is, change the indirect jump at $9DE7 to point to reset.

9DE7:4C 59 FF

Run the code at $9D84.

9D84G

When you hear the beep, examine the bytes at $9D5C and $9D5D.

9D5C.9D5D

The screen will display (low byte first) the address indirectly jumped to as $33D5. Begin listing from $33D5, and you should find the next exit jump at $34BC. It goes to $00FD. Change the jump at $34BC to point to reset.

34BC:4C 59 FF

Run the code at $33D5.

33D5G

The disk drive will start, and the last segment of the program will be loaded. If everything worked correctly, you should hear a beep, the drive will stop, and the screen will be filled with garbage.

THE PROGRAM START

Normally, the program would next jump via the page zero location which was just changed at $34BC to the start of the Multi-Disk Catalog main program. Find the destination of the next jump by examining the code at $00FD.

00FDL

To make it difficult, the software protectors have put one last obstacle in your path. $00FD takes an indirect jump through page zero locations $4E and $4F to the start of the program. Unfortunately, an examination of these locations to find out where the jump goes isn't possible because they are changed when a Reset cycle is executed.

Examine the code at $348F.

348FL

The bytes at locations $4E and $4F are set from $33C0 and $33C1, respectively. Examine these locations.

33C0.33C1

The starting address (low byte first) of the main program is $1294.

The program occupies memory from $800-$18FF, $5000-$5CFF, and $9D00-$BFFF. (Find this out by scrolling through memory to identify program statements and data, often a trial and error process. If there is too much, no real harm is done, but too little and the program will not run.)

The last step is to capture the program under normal DOS.

MOVING THE MEMORY

Warm booting a slave diskette will overwrite memory locations $800-$8FF and $9600-$BFFF, but everything from $900-$95FF will be unaffected. Move the 'lower" part of the program ($800-$1800) up and out of the way of the boot, and put it adjacent to the "middle" part.

3F00<800.18FFM

Move the "top" part of the program down.

5D00<9D00.BFFFM

Add this relocation routine so that when the program is BRUN everything will be put back in the proper place.

3ED0:00 00 A9 5D 85 3D A9 7F
3ED8:85 3F A9 9D 85 43 20 F3
3EE0:3E A9 3F 85 3D A9 4F 85
3EE8:3F A9 08 85 43 20 F3 3E
3EF0:4C 94 12 A0 FF 84 3E C8
3EF8:84 3C 84 42 20 2C FE 60

The source code for this routine looks like:

```
3ED2-A9 5D      LDA  #$5D    Set up
3ED4-85 3D      STA  $3D       address
3ED6-A9 7F      LDA  #$7F    data to
3ED8-85 3F      STA  $3F       move top
3EDA-A9 9D      LDA  #$9D    part of
3EDC-85 43      STA  $43       program.
3EDE-20 F3 3E   JSR  $3EF3   Call move.
3EE1-A9 3F      LDA  #$3F    Do it again
3EE3-85 3D      STA  $3D       for the
3EE5-A9 4F      LDA  #$4F      bottom
3EE7-85 3F      STA  $3F       part of the
3EE9-A9 08      LDA  #$08      program.
3EEB-85 43      STA  $43
3EED-20 F3 3E   JSR  $3EF3   Call move
3EF0-4C 94 12   JMP  $1294   Run Program
3EF3-A0 FF      LDY  #$FF    Get things
3EF5-84 3E      STY  $3E       ready
3EF7-C8         INY            before
3EF8-84 3C      STY  $3C     calling the
3EFA-84 42      STY  $42       monitor
3EFC-20 2C FE   JSR  $FE2C     move
3EFF-60         RTS            routine.
```

THE FINAL TEST

Finally, remove the protected disk from the drive and replace it with the normal DOS (slave) disk. Warm boot it by typing:

6 CTRL-P

Do not type "CTRL P", just hold the ctrl key down and tap the "P" key, then release the ctrl key and press return.

When the boot is complete, save the code as a binary or "B" type file.

BSAVE MDC,A$3ED2,L$412E

You now have an unlocked program that will BRUN normally, or it can be customized as you see fit.

Try this procedure with your other 'one-shot" load programs. You will probably be surprised at how often it works, with a little sleuthing. That is where the perseverance part comes in.

The zero page locations that are changed by the reset routine are:

$20-$2B, $31, $33-$3F, $40-$49 and $4E-$4F.

Introduction to 'Parms'

This is our most complete and updated list of parameters for the four leading bit-copy programs: Locksmith, Nibbles Away, Copy II + and Back-It-Up.

Although the instructions for the use of these parms are part of their documentation, additional assistance is offered in the articles accompanying each parm list.

If any particular software package is not included in the parms list of your favorite bit copier, it can either be copied without parms, or no parms have been received by us for that package.

To save space, the names of the publishers of the programs (whose copy parms are found in any of the lists that follow) have been abbreviated.

Table of Abbreviations of Publishers

AC . . . . . Apple Computer
AD . . . . . Anthro-Digital
AI . . . . . Adventure International
AV . . . . . Avante Garde
ARN . . . . . Action-Research Northwest
ARS . . . . . ARS Publications
ART . . . . . Artsci
ARW . . . . . Artworks
AST . . . . . Apple Software Technology
AVH . . . . . Avalon Hill
AW . . . . . Addison Wesley
BC . . . . . Budgeco
BES . . . . . Bulls Eye Software
BP . . . . . Beamon Porter
BS . . . . . Broderbund Software
BUS . . . . . Business Solutions
CBS . . . . . CBS Software
CAI . . . . . Computer Advanced Ideas
CC . . . . . Cavalier Computer
CES . . . . . CE Software
CDS . . . . . ????
CP . . . . . California Pacific
CPS . . . . . Counter Point Software
CS . . . . . Computer Solutions
CTS . . . . . Contentinental Software
CW . . . . . Compuware
CX . . . . . Cedex
DAT . . . . . Data Transforms
DLM . . . . . DLM
DM . . . . . Data Most
DS . . . . . Datasoft
DSS . . . . . Decision Support Software
DY . . . . . Dynacomp
EAI . . . . . Educational Activities Inc.
EC . . . . . Educational Courseware
EIN . . . . . Einstein
EP . . . . . Epyx
EU . . . . . Eureka
EW . . . . . Eduware
FC . . . . . Frontier Computing
GB . . . . . Gebelli
HAL . . . . . HAL
HOB . . . . . Hobar
HOW . . . . . Howardsoft
HN . . . . . Hayden
HT . . . . . High Technology
ICP . . . . . Image Computer Products
IDSI . . . Innovative Design Software Inc.
IN . . . . . Insoft
INS . . . . . Instant Software
ISM . . . . . ISM
KL . . . . . Krell
KN . . . . . Kensington
L10 . . . . . Level Ten
LC . . . . . Learning Company
LJK . . . . . LJK Enterprises
LNS . . . . . Lightning Software
LOD . . . . . Logidisque
LTS . . . . . Lotus
MAG . . . . . Micro Applications Group
MF . . . . . Micro Fun
MH . . . . . Megahaus
MIS . . . . . Microsoft
ML . . . . . Micro Lab
MM . . . . . Micromax
MS . . . . . Mind Systems
MSP . . . . . Micro-Sparc
MU . . . . . Muse
MWD . . . . . Micro Ware Distributors
MWS . . . . . Midwest Software
OD . . . . . Odesta
OR . . . . . Origin Systems
PBS . . . . . Personal Business Systems
PDS . . . . . Picadilly Software
PEN . . . . . Penguin Software
PDI . . . . . Program Design Inc.
PH . . . . . Phoenix
PIC . . . . . Picadilly
QS . . . . . Quality Software
SAA . . . . . State of the Art Accounting
SAM . . . . . Howard W. Sams
SDL . . . . . Systems Design Lab
SEN . . . . . Sensible Software
SFS . . . . . Sof/Sys Inc.
SIR . . . . . Sir-Tech
SL . . . . . Sub Logic
SMI . . . . . Smith Micro Software
SNT . . . . . Sentient Software
SOL . . . . . Sierra On-Line
SPC . . . . . Software Publishing Corp.
SPN . . . . . Spinnaker Software
SPT . . . . . Spectrum
SRS . . . . . Sirius Software
SSI . . . . . Strategic Simulations
SSP . . . . . Sterling Swift Publishers
SSM . . . . . Transcend
SVS . . . . . Silicon Valley Systems
SW . . . . . Stoneware
SY . . . . . Synergistic Software
TER . . . . . Terrapin
TKS . . . . . Turnkey Software
TSR . . . . . TSR
ULS . . . . . Ultrasoft
UNK . . . . . Unknown publisher
USA . . . . . USA
VC . . . . . Virtual Combinatics
VCP . . . . . Visicorp
VER . . . . . Versa
VOY . . . . . Voyager
VX . . . . . Videx
XPS . . . . . XPS

The Compleat Guide to

Locksmith Parameters

Locksmith is a product of Omega Microware, Inc.

This article describes user-changeable Locksmith parameters and program logic.

Note: This article is of a highly technical nature and is intended primarily for the advanced user of Locksmith.

BACKGROUND

When Locksmith was first introduced in January 1981, it would copy almost all disks with no special instructions from the user. Only a few disks required parameter changes. Alas, those good old days are gone forever. Instead of providing the user with a better back-up policy, software vendors decided to escalate the battle by developing more complicated (and, in some cases, bizarre) protection techniques. Because of the many different techniques now in use, it is likely that many disks will require some input from the user in the form of parameter changes.

OVERVIEW

Locksmith copies disks by reading a track, performing analysis on the data and writing the track to the copy disk. Reading and writing are fairly straightforward functions.

The analysis of the track data is by far the most difficult task and must provide for flexibility. Many analysis routines (algorithms) are provided within Locksmith. Each algorithm performs a specific function relating to the analysis of track data. By changing parameters, the user may select, disable or change the execution order of algorithms. Parameters may also be used to define values to be used by individual algorithms.

ALGORITHMS

The algorithms are numbered from 0 to $23 (all values are in hex). New algorithms may be added in future versions of Locksmith. During track analysis, algorithms are selected sequentially from a table of algorithm numbers located from PARM 4C-80. As algorithms are selected from this table during analysis, they are displayed on the screen as two-digit hex numbers in inverse video.

Algorithm 00 indicates a null algorithm, which can replace algorithm numbers in the table the user wants to disable. An FF entry in this table indicates the end of the algorithms to perform.

Currently, the algorithm table contains four separate algorithm sequences, each one terminated by an FF entry. The starting point of the algorithm sequence to be used is defined by PARM 25. This parameter contains the index into the algorithm table to be used as the first algorithm of a sequence. For example, if PARM 25 = 00, the algorithm sequence would start at PARM 4C. If PARM 25 = 10, the algorithm sequence would start at PARM 5C. The section of algorithm table starting at PARM 71 is selected as an algorithm sequence start (instead of PARM 4C) when synchronized tracks are chosen.

Algorithms, in addition to performing their specialized function, can return a flag to indicate success or failure. It is possible to indicate an algorithm is to be performed only if the previous algorithm failed. This may be done by setting the high-order bit of the algorithm number within the algorithm table. For example, an entry of A1 indicates that algorithm 21 is to be performed only if the previous algorithm failed.

DESCRIPTION OF ALGORITHMS

The following is a list of algorithm numbers and the parameters which affect them.

ALG 00 (This algorithm doesn't do much of anything.)

ALG 01 (Consecutive nibbles to self-sync) Changes normal nibbles to self-sync nibbles based on: finding (PARM 10) consecutive nibbles in the range (PARM 34) to (PARM 35), inclusive. For example, if PARM 10 = 0C, PARM 34 = FE and PARM 35 = FF, then algorithm 01 would search for sequences of length 0C nibbles with values FE through FF and set them to self-sync.

ALG 02 (Invalids to self-sync) Sets invalid nibbles (those with three or more consecutive zero bits) to self-sync.

ALG 03 (Standardize self-sync) Sets all self-sync to (PARM 33), which must have high-order bit clear.

ALG 04 (Loner self-sync to normal) Sets consecutive self-sync strings less than or equal to (PARM 3C) to normal.

ALG 05 (Glitch remover) Sets consecutive normal nibbles of length less than or equal to (PARM 12) to self-sync.

ALG 06 (Sets self-sync by marker pattern match) Searches for pattern specified by (PARM 44-4B) and sets the previous (PARM 40) nibbles to self-sync. Values of 00 within the pattern are 'don't care" and always match.

ALG 07 (Extend bit shifted self-sync) Extends self-sync strings backwards, using the table at (PARM 86-A5). This table contains nibble value sequences frequently found to be self-sync.

ALG 08 (Reserved for future use.)

ALG 09 (Trackstart after longest gap) Sets trackstart to first normal after longest string of self-sync (gap).

ALG 0A (Minimum length self-sync) Extends self-sync strings backwards to minimum length of (PARM 2C).

ALG 0B (Sets self-sync by self-sync pattern match) Sets self-sync based on multiple-byte pattern match. Pattern is defined at (PARM 81-85) and is terminated with a 00 value.

ALG 0C (Shortens all gaps) Shorten all gaps (consecutive strings of self-sync) by (PARM 41) nibbles if the string length is greater than or equal to (PARM 16).

ALG 0D (2 of 3 gap merge) Merges first and second gaps (by setting to self-sync, nibbles between them) if three gaps are found within (PARM 26) nibbles. (The gaps merged are usually the gaps after a data field.)

ALG 0E (Trackstart after first self-sync) Sets trackstart to first normal after the first string of self-sync.

ALG 0F (Shortens longest gaps) Shortens the longest gap if longer than (PARM 2C) by (PARM XX) nibbles. Repeat this procedure (PARM YY) times. XX = 27 (or 29 if synchronized). YY = 28 (or 2A if synchronized).

ALG 10 (Reserved for future use.)

ALG 11 (Sets failure flag) Same as algorithm 00, but sets the failure flag.

ALG 12 (Trackstart by marker pattern match) Sets trackstart to the first sequence to match pattern at (PARM 44-4B) (see ALG 06).

ALG 13 (Center of gaps to normal) Leaving eight self-sync at the start and at the end of a gap, sets self-sync in the center of the gap to normal.

ALG 14 (Bit-translate to self-sync) Using the bit table at (PARM D9-E8), translates nibbles corresponding to a one-bit to self-sync. Bits in the table represent values for nibbles in the following order: 80,81,82, ... FC,FD,FE,FF

ALG 15 (Reserved for future use.)

ALG 16 (Reserved for future use.)

ALG 17 (Track-end and compare) This

algorithm searches for a repeat of the track-start beginning at (PARM 1D) pages beyond the current track-start. A repeat of the track-start is determined by matching (PARM 1E) number of nibbles. If the track size is greater than (PARM 1B) pages, an error 2 status code will be issued.

Once a track-end is chosen, the first two track images are compared, nibble for nibble. If an unequal nibble compare occurs, a look-ahead of up to (PARM 13) nibbles is performed, looking for self-sync.

If self-sync is found, the compare failure is ignored. If no self-sync is found during this look-ahead, a counter is incremented for the compare-failure, and this count is checked against (PARM 14), which must not be exceeded, or an error 4 status code is issued immediately.

The third track image is then used as a tie-breaker to determine which of the first or second track images is correct. The exact position in the third track image is found by first finding the approximate location in the third image (by using track length), backing up (PARM 11) nibbles, and pattern-matching (PARM 32) number of nibbles, while searching through the next (PARM 31) number of nibbles. The first image is corrected by the tie-breaker nibble. This algorithm returns a success/fail flag.

ALG 18-1F (PARM modifier) These algorithms are used to modify PARMs dynamically. The table at (PARM B6-D8) consists of several sequences of PARM modifier entries. Each PARM modifier entry consists of a pair of bytes. The first byte defines the PARM number, and the second byte defines the new PARM value. The end of a sequence is indicated by a 00 entry for PARM number, and a new sequence begins with the next byte. Algorithm 18 invokes the first sequence of parameter modifier entries, algorithm 19 invokes the second sequence, etc. Using these algorithms, parameters may be automatically changed and restored during analysis. The defaults for these algorithms are currently set as follows:

ALG 18 sets 13-sector PARMs.

ALG 19 sets 16-sector PARMs.

ALG 1A sets misc. PARMs.

ALG 1B sets nibble-counting PARMs.

ALG 20 goes to Nibble Buffer address. This algorithm is used in conjunction with the Nibble Editor. It prompts the user for an address to go to, and the Nibble Editor cursor is immediately placed at that location. (See "Invoking Algorithms from the Nibble Editor.")

ALG 21 (Sets error code 1) Issues an error 1 status code. It is usually placed in the algorithm table with the high-order bit set to cause it to execute only when the previous algorithm fails.

ALG 22 (Backs up trackstart to front of gap) Moves the trackstart pointer backwards to the beginning of the preceding gap.

ALG 23 (Sets trackstart to longest normal) Sets trackstart to the first nibble of the longest sequence of normal nibbles.

Printer Control PARMs

(Parm 2D) Specifies the printer slot, and (PARM 2E) is set to 00 if Locksmith is not to generate >CR< at the end of a line, or left at 01 if >CR<s are to be generated.

Maximum Error Count PARMs

(Parm 01),(PARM 02), and (PARM 04) are used to specify the number of errors allowed for error codes 1, 2 and 4 in automatic error retry mode. If increments of ½ tracks are used, (PARM 09),(PARM 0A) and (PARM 0C) are used instead.

Nibble-Counting PARMs

Some protected disks use a technique known as nibble counting. This technique is based on the fact that all Apple disk drives run at slightly different speeds, and even the speed of one specific disk drive varies slightly over time. Disks which are protected by this method count the nibbles on a given track and record this unique number somewhere else on the disk. When the disk is booted by the user, the nibble count on the track in question is checked against the correct value. Simply copying the track will almost always write a different number of nibbles due to disk drive speed variation.

Locksmith will preserve nibble counts on any track requested. After the track is written to the copy disk, the nibbles are counted and compared to the original count to be preserved. The difference is shown as a four-digit hex number preceded by < or > to indicate to the user which way to adjust the count manually.

The count may be adjusted in one of two ways. Either the disk speed adjustment pot (inside the disk drive) can be turned in the direction indicated by the < or > arrows (see user manual regarding disk speed adjustment), or limited adjustment can be done from software without adjusting the disk speed.

To adjust the nibble count from software, press either < or > as indicated, and wait until the speaker begins beeping. The speaker will beep rapidly once for each nibble that the track is being shortened or lengthened. Then press the return key (or any key other than < and >) and allow the nibble count routine to test the track again. When the nibble count is within the tolerance value specified by PARM 37 (normally 00), the track will be considered copied correctly. This technique may seem cumbersome, but it is the only way in which a track may be copied while preserving the nibble count.

There are three parameters which are used when nibble-count preservation is desired. Setting (PARM 36) to 01 turns on nibble-counting. The nibble-count tolerance value, (PARM 37), specifies how close to the original disk the copy must be. When nibble-counting, the track-end pointer is moved up by (PARM E9) pages before writing.

PARMs Used for Synchronizing

(Parm 22) specifies the track*2 to sync with. This is normally 00 but may be set to any track. (Parm 1F) is the length of the nibble sequence to sync with, and (PARM A6-B5) contain the pattern to match when attempting to sync on the sync-track. Values of 00 within the pattern are "don't care" and always match. (PARM 23) and (PARM 24) are values which can be used to adjust the accuracy of the sync-track routine. They are normally equal, and can be adjusted by increasing the value of one with respect to the other.

PARMs Used to Control Writing

(Parm 20) contains the lead-in self-sync nibble value. (Parm 2F-30) (default is $1A00) number of these lead-in self sync nibbles are written before track data is written, with the exception of synchronized track writing, which is preceded by (PARM 23) lead-in self-sync nibbles. The number of framing bits (1 or 2) is contained in (PARM 21). This places the proper number of trailing zero-bits after self sync. (Parm 2B) contains the number of the algorithm to be used to shorten the track after an over-write is detected by verify readback failure.

Other PARMs

(Parm 38) is the number of nibbles to test during verify readback. (Parm 39), if set non-zero, shows the hi-res screen during analysis, to provide a graphic representation of analysis. (Parm 3A) is used during disk certify. It specifies the maximum size of the track-end glitch. (Parm 3B), when set to 01, causes the Nibble Editor to be entered for every track before analysis.

Debug Parameter

(PARM 00) is a special parameter intended for use during Locksmith debugging. When this PARM is set to 11, certain debugging options are enabled.

They are:

1. Inspector entry is allowed even without a resident RWTS.

2. The Nibble Editor is entered automatically without prompting the user for a track to read. This allows the previous track to be examined.

Invoking algorithms from the Nibble Editor.

With debug PARM set (PARM 00 = 11), the Nibble Editor is sensitive to two

additional commands. These are ctrl S and ctrl A. Ctrl S invokes Locksmith track-analysis for the track currently in the Nibble Buffer. Ctrl A first allows the user to change parameters by entering the parameter modifier, and after the user has indicated the end of parameter changes with a >CR<, it prompts the user for algorithm number.

The user-entered algorithm number is executed immediately, and control is returned to the Nibble Editor. In this way, the user can dynamically test the effects of specific Locksmith algorithm sequences when attempting to copy unknown disks. Algorithm 00 can be specified if no processing is to be done.

Algorithm 20 is very useful within the Nibble Editor to rapidly go to a specific address within the Nibble Buffer.

A2-FS1 (Flight Simulator) (SL)
00-21 BY 1.5
07-08 BY 1
9.5

Alternate method
00
1.5-21 BY 1.5.....44:DB 45:AB 46:BF 40:20 4E:00 54:12
07-08 BY 1
9.5

A2-PB1 (Night Mission) (SL)
00
01-15.....44:DB 45:AB 46:BF 40:20 4E:00 54:12
Write protect before running.

AIRSIM-1 (MS)
s 00-02 BY 1.....A8:B5 78:00 79:12 47:FF
s 09-22 BY 1
s 03-08 BY 1.....79:0E (08 ERROR IS O.K.)
Write protect before running.

AKALABETH (CP)
s 00-18 BY 1.....44:DD

ALIEN RAIN (BS)
s 00-0E BY 1

ALIEN TYPHOON (BS)
S 00-0E BY 1

Alternate method
00
01-05 BY 1.....46:AD
06-0E BY 1.....44:DE

APPLE LOGO (AC)
00-22
01.....4C:1B 57:00 E9:02 34:FF 50:00 51:00 52:00 53:00
Uses nibble count.

APPLEOIDS (CP)
00-22
03.5
21.5

APPLE PANIC (BS)
00-0D
Alternate method
00
01-0C.....44:DD

APPLE PERSONAL FINANCE MANAGER (AC)
00-22.....10:04 16:40 46:96 51:00 53:0B 54:12 81:CF 82:F3 83:FC

APPLE PRINT USING (UNK)
s 00-23 BY 1

APPLE III BUSINESS GRAPHICS (AC)
s 00-22 BY 1.....18:50 19:00 40:04 46:96 75:00 76:00 77:00 78:00 79:12

APPLE WORLD (USA)
00-23

APPLE-WRITER II (AC)
00-22.....46:96 54:12

APPLE-WRITER III (AC)
s 00-22 BY 1

AUTOBAHN (SRS)
00
S 04-06 BY 1.....74:00
S 09.5-0C.5 BY 1

Alternate method
00
04-06.....74:00
09.5-0C.5

BAG OF TRICKS (QS)
00
01-14.....40:10 44:D6 53:00

BANK STREET WRITER (BS) (use MODE 2)
1-19.....4D:0B 4E:02 4F:01 50:06 1:05 52:0D 53:07 58:19 9:06 5A:1A 5B:FF 40:07 44:A5 45:96 46:BF BD:04 BE:44 BF:45 C0:D5 C1:46 C2:D6 C3:D0 C4:44 C5:A5 C6:45 C7:96 C8:46 C9:BF CA:00
1A-22.....44:EE 45:EF 46:F5 40:80 58:FF 4D:00
00

BASIC MAILER (ART)
00-22.....4F:0B

BATTLE OF SHILOH (SS)
00
01-22.....4F:0B

BEER RUN (SRS)
s 00.....18:20 19:00 46:96 4D:00 4E:00 52:00 53:00 54:12 57:00 40:20
s 01.5-0D.5 BY 1

Alternate method
00
01.5-0D.5 BY 1.....72:00 73:00 77:00 78:00 79:12 7C:00 40:20 19:00 44:DD 45:AD 46:DA

BORG (SRS)
s 00.....18:20 19:00 40:20 44:DD 45:AD 46:DA 72:00 73:00 77:00 78:00 79:12 7C:00
s 01.5-0B.5 BY 1
s 0D-20 BY 1

Alternate method
00.....18:20 19:00 40:20 4D:00 4E:00 4E:00 52:00 53:00 54:12 57:00 72:00 73:00 77:00 78:00 79:12 7C:00 44:DD 45:AD 46:DA
s 01.5-0C.5 BY 1
s 0D-20 BY 1

BPI BUSINESS ACCOUNTING (AC)
00-22.....81:AD 82:FB 83:E6 84:FF 40:08 16:08 41:FF 19:00 58:0B 59:FF

Alternate method
00-22.....19:00 21:02 58:19 59:06 5A:1A 5B:FF BD:44 BE:E6 BF:45 C0:FF C1:40 C2:01 C4:44 C5:D5 C6:45 C7:AA C8:40 C9:04 CA:00

BRAIN SURGEON (UNK)
00-22
1B.....4C:1B 57:00 E9:02 D2:00

Alternate method
00-22
04.....4C:1B 57:00 E9:02 D2:00

BUDGE'S SPACE ALBUM (CP)
00-0B

BUG ATTACK (CC)
00-13 (0E-13 Errors may occur)
1E.....4C:1B 57:00 E9:02
Uses nibble count.

CANNONBALL BLITZ (SOL)
00-22.....46:96 54:12 53:00
03-0F.....4C:1B 57:00 E9:02
Uses nibble count.

CARTELS AND CUTTHROATS (SS)
00
01-22.....4F:0B

Alternate method
s 00.....46:96
s 01-23 BY 1.....44:AB 45:AB 79:12

CASTLE WOLFENSTEIN (MU)
s 00-22 BY 1.....46:B5 79:12

Alternate method
s 00-22 BY 1

COMPUTER AIR COMBAT (SS)
00-22.....25:19 65:00 6B:00

COMPUTER AMBUSH (SS)
00
01-22.....4F:0B

Alternate method
00-22.....25:19 65:00 6B:00

COMPUTER CONFLICT (SS)
00
01-22.....4F:0B

COMPUTER FOOSBALL (SRS)
00.....18:20 19:00 46:96 4D:00 4E:00 52:00 53:00 54:12 57:00 40:20
s 1.5-9.5
1A
(For errors on trk 9.5 use manual retry till no error)

COMPUTER NAPOLEONICS (SS)
00
01-22.....4F:0B

COMPUTER QUARTERBACK (SRS)
00-22.....25:19 65:00 60:00

CONGLOMERATES COLLIDE (RO)
00-22
1B.....36:01

CONGO (SN)
00-22.....46:96 4D:00 4E:00 21:02
26:06 51:00

CONTEXT CONNECTION (CON)
00-22.....19:01 21:02 58:19 59:06
5A:1A 5B:FF BD:44 BE:EB
BF:45 C0:FD C1:40 C2:01
C4:44 C5:D5 C6:45 C7:AA
C8:40 C9:04 CA:00

COPTS AND ROBBERS (SRS)
00.....18:20 19:00 46:96 4D:00
4E:00 52:00 53:00 54:12
57:00 40:20
s 01.5-0F.5 BY 1.....72:00 73:00 77:00 78:00
79:12 7C:00 40:20 19:00
44:DD 45:AD 46:DA

CRANSTON MANOR (SOL)
00-22
18.....4C:1B 57:00 E9:02
Uses nibble count.

Alternate method
00-22
18.....53:00 44:D5 45:FE 4C:1B
57:00
Uses nibble count.

CRISIS MOUNTAIN (SY)
00-22.....1B:19 1D:18 1E:30 40:02
44:00 45:00 46:EB 47:AF
4E:00 51:00 52:00

CROSSFIRE (SOL)
00-22
01.....4C:1B 57:00 E9:02
Uses nibble count.

CROSSWORD MAGIC (L&S)
s 00-22 BY 1.....46:96 75:00 76:00 77:00
78:00 79:12 4B:AA

CRUNCH, CRUMBLE & CHOMP (EAS)
s 00-22 BY 1

CYBERSTRIKE (SRS)
00
s 03-0B BY 1
s 11-1C BY 1

Alternate method
00
s 04-0B BY 1.....46:F5 79:12
s 11-1C BY 1.....46:B5

Alternate method
00
s 04-0B BY 1.....46:F5 79:12
11-1C.....46:B5

CYBORG (SN)
00-22.....47:FF 48:F8 4D:00 4E:00
51:00 40:04

CYTRON MASTERS (UNK)
00-22.....25:19 65:00 6B:00

DARK FOREST (SRS)
s 00-22 BY 1

Alternate method
00.....18:20 19:00 46:96 4D:00
4E:00 52:00 53:00 54:12
57:00 40:20
s 02-22 BY 1.....72:00 73:00 77:00 78:00
79:12 7C:00 40:20 19:00
44:D5 45:AA 46:AF

DATADEX (IU)
s 00-02 BY 1.....79:12 46:96
s 03.5
s 05-22 BY 1

DATA FACTORY 5.0 (ML)
s 00-23 BY 1.....46:96 79:12 71:19 75:00
76:00 77:00 78:00

DATA PLAN (UNK)
s 00-22 BY 1

DATA REPORTER (SY)
00-22.....4D:00 46:96 54:12

DB MASTER & UTILITIES (SW)
00-05
06.5-22.5 BY 1

Alternate method
00-05
06.5-22.5 BY 1
Write protect before running.

DB MASTER & UTILITIES v3.2
00
s 01-05 BY 1
06.5-21.5 BY 1
22.5.....4D:00 46:96 54:12

DEAD LINE (IC)
00-22.....46:96 40:14

DESK TOP PLAN II (VCP)
00-22.....19:01 21:02 58:19 59:06
5A:1A 5B:FF BD:44 BE:EB
BF:45 C0:FD C1:40 C2:01
C4:44 C5:D5 C6:45 C7:AA
C8:40 C9:04 CA:00

DISK LIBRARY (INS)
00-22.....40:09 53:00 16:77 46:96
47:AA 48:AA 4B:AA 54:12
21:02

DISK ORGANIZER II (SEN)
00
s 02-04 BY 1

s 0A-0B BY 1
01.....4C:1B 57:00 E9:04
Uses nibble count.

DISK RECOVERY (SEN)
00
s 02-16 BY 1

Alternate method
00
s 02-04 BY 1
s 0A-0B BY 1

DRAGON GAMES (EAI)
00-22
04.....4C:1B 57:00 E9:02 D2:00

ELECTRIC DUET (IN)
00-22.....40:08 16:08 41:FF 19:00
81:DE 82:AA 58:0B 59:FF

EPOCH (SRS)
00.....18:20 19:00 46:96 4D:00
4E:00 52:00 53:00 54:12
57:00 40:20
s 01.5-0F.5 BY 1.....72:00 73:00 77:00 78:00
79:12 7C:00 40:20 19:00
44:D5 45:AA 46:DA

ESCAPE FROM ARCTURUS (SY)
00-22.....4D:00

Alternate method
s 00-22 BY 1.....4D:00

ESCAPE FROM RUNGISTAN (SRS)
s 00-21 BY 1.....36:01
Uses nibble count.

Alternate method
00-22
s 22.....46:96

EXECUTIVE SECRETARY (PBS)
01.5-21.5 BY 1
00-22 BY 1

Alternate method
00-22.....46:96 54:12

EXPEDITER (SOL)
00-22
03 & 1F.....4C:1B 57:00 E9:02
Uses nibble count.

Alternate method
00-22
03 & 1F.....4C:1B 57:00 E9:02 D2:00

Alternate method
00-22
03 & 1F.....4C:1B D2:00
APPLY PATCH NC30 (version 4.0 only)

Alternate method
00-22
03 & 1F.....4C:1B 57:00 E9:02
(version 4.1 only)

FACEMAKER (SPN)
0-22.....21:02 40:04 52:00 53:00
58:0B 59:FF 81:AA 82:9B
83:FC

FALCONS (PDS)
00
01.5-04.5 BY 1.5....18:20 34:AA
44:DF 45:AD 46:FE
05.5
07-0A BY 1
0B.5-0E.5 BY 1.5
10-12 BY 1
13.5-14.5 BY 1
16-19 BY 1.5
1A-1B.5 BY 1.5

Alternate method
00
01.5-04.5 BY 1.5....18:40 19:00 34:AA 40:40
44:DF 45:AD 46:00 4E:00
4D:00 52:00 53:00
05.5
07-0A BY 1
0B.5-0E.5 BY 1.5
10-12 BY 1
13.5-14.5 BY 1
16-19 BY 1.5
1A-1B.5 BY 1.5

Alternate method
s 00
s 01.5-19.5 BY 1

FINANCIAL CONTROLLER (UNK)
s 00-22 BY 1

FIREBIRD (GS)
00.....18:20 19:00 46:96 4D:00
4E:00 52:00 53:00 54:12
57:00 40:20
s 01.5-0B.5 BY 1....72:00 73:00 77:00 78:00
79:12 7C:00 40:20 19:00
44:DD 45:AD 46:DA

GALACTIC SAGA I (EMPIRE) (BS)
s 00-23 BY 1

GALACTIC SAGA II (TRADER) (BS)
00-23

GALACTIC SAGA IV (BS)
00.....18:50 19:00 40:20 46:96 4D:00 4E:00 52:00 53:00 54:12 57:00
01-22.....44:D5 45:AA 46:B5

GALAXY WARS (BS)
s 00-12 BY 2

GAMMA GOBLINS (SRS)
00.....18:20 19:00 46:96 4D:00 4E:00 52:00 53:00 54:12 57:00 40:20
s 01.5-0D.5 BY 1....72:00 73:00 77:00 78:00 79:12 7C:00 40:20 19:00 44:DD 45:AD 46:DA

GENETIC DRIFT (BS)
00.....18:50 19:00 40:20 46:96 4D:00 4E:00 52:00 53:00 54:12 57:00
01-03 BY 1.....44:BB 45:B5 46:BB
04.5-06 BY 1.5
07.5-0B.5 BY 1
0D.....44:D4 45:D5 46:BB
0E.5-12.5 BY 1.....44:AD 45:B5 46:DE

GOBBLER (SOL)
00-22.....4E:00
03.....4C:1B D2:00 45:DB 4E:01 34:FF 54:12 52:00

GOLD RUSH (SN)
00-22.....46:96 4D:00 4E:00 21:02 26:06 51:00

GOLDEN MOUNTAIN (BS)
00
s 01-0D BY 2
02-0E BY 2

GORGON (SRS)
00.....54:12
s 01.5-0E.5 BY 1.....54:09

Alternate method
00.....18:20 19:00 46:96 4D:00 4E:00 52:00 53:00 54:12 57:00 40:20
s 01.5-0E.5 BY 1.....72:00 73:00 77:00 78:00 79:12 7C:00 40:20 19:00 44:DD 45:AD 46:DA

Alternate method
00
1.5-E.5 BY 1.....72:00 73:00 77:00 78:00 79:12 7C:00 40:20 19:00 44:DD

HADRON (SRS)
s 00.....18:20 19:00 46:96 4D:00 4E:00 52:00 53:00 54:12 57:00 40:20
s 01.5-0D.5 BY 1

Alternate method
00.....18:20 19:00 46:96 4D:00 4E:00 52:00 53:00 54:12 57:00 40:20
s 01.5-0E.5 BY 1.....72:00 73:00 77:00 78:00 79:12 7C:00 40:20 19:00 44:DD 45:AD 46:DA

HAYDEN ALIBI (HN)
00-02
03-22.....51:00 52:00 53:00 54:12 19:00 18:50 57:00 44:D4 46:B5
1B.....4C:1B E9:02
Uses nibble count.

HAYDEN APPLESOFT COMPILER (HN)
s 00-22 BY 1.....46:96 71:19 79:12
Errors on 10-1E O.K.
Very sensitive to drive speed.

HI-RES CRIBBAGE (SOL)
s 00-05 BY 1
06-22

HI-RES FOOTBALL (SOL)
s 00-05 BY 1
06-22

HI-RES GOLF (AG)
00-22 .....4E:00 46:B5 54:12

HI-RES SECRETS (AG)
00-22.....46:96 54:12 34:FB

HIRES SOCCER (SOL)
s 00-22 BY 1

HOME ACCOUNTANT (CTS)
00-22.....46:96 54:12

HYPER HEAD ON (BS)
s 00-12 BY 2

IMAGE PRINTER (SEN)
s 00-07 BY 1
s 09-22 BY 1
08.....4C:1B 57:00 E9:02 D2:00 44:FE 45:AB 54:12 50:00 51:00 52:00 53:00

INVOICE FACTORY (ML)
00-22.....46:96 54:12

JAWBREAKER (SOL)
00-22
03.....4C:1B 57:00 E9:01
Uses nibble count.

Alternate method
00-22
03.....34:FF 44:DF 45:EF 46:F7 50:00 51:00 52:00 53:00 54:12

LETTER PERFECT (LJK)
00-22.....44:00 45:D5 46:AA

MAD VENTURE (ML)
s 00-23 BY 1

MAGIC WINDOW (ART)
00-22.....4F:0B

MAGIC SPELLER (ART)
00-22.....4F:0B

MAGIC WORD (ART)
00-22.....4F:0B

MASTER DIAGNOSTICS PLUS (UNK)
00-22
04.....4C:1B 57:00 E9:02 D2:00

MASTERTYPE (LNS)
00-02
03-1A.....44:D4 54:12
1C-22

Alternate method
00-02
03-1A.....44:D4
1C-22

MICRO BASEBALL (SW)
00-04
s 05-22 BY 1

MICRO COURIER (MC)
00-22
1F.....81:97 82:EB 40:08 16:08 41:FF 19:00 58:0B 59:FF

Alternate Method
00-22.....46:96 54:12
1F.....81:97 82:EB 40:08 16:08 41:FF 19:00 58:0B 59:FF

MICRO TELEGRAM (MC)
00-22
1F.....81:97 82:EB 40:08 16:08 41:FF 19:00 58:0B 59:FF

MICROWAVE (CC)
00-22
11.....4C:1B 57:00 E9:02
Uses nibble count.

MILLIKEN MATH (ML)
00-22.....4C:18 46:B5 54:12 50:00 51:00 52:00 53:00

MISSLE DEFENCE (SOL)
s 00-22 BY 1

MISSION ASTEROID (SOL)
s 00-22 BY 1

MOUSKATTACK (SOL)
00-22.....46:96 54:12 53:00
23.....4C:1B 57:00 E9:02
Uses nibble count.

MONTY PLAYS MONOPOLY (IC)
00-05.....1E:0B

MULTI DISK CATALOG III (SEN)
s 00-02 BY 1
s 04-09 BY 1

Alternate method
s 00-22 BY 1

MYSTERY HOUSE (SOL)
s 00-22 BY 1

Alternate method
00-10
12-22

NIGHTMARE GALLERY (SY)
00-22.....46:96 54:12 51:00 4D:00 4E:00

OLYMPIC DECATHLON (MIS)
00-22

Alternate method
s 00-22 BY 1.....46:B5 A8:00 71:18 79:12

00-TOPOS (SN)
00-22.....32:88 01:06

Alternate method
00-22.....21:02

Alternate method
00-22.....4D:00 4E:00 21:02 2C:06 48:EE 49:FF

OPERATION APOCALYPSE (SS)
00-22.....25:19 65:00 6B:00

ORBITRON (SRS)
00.....18:20 19:00 46:96 4D:00
4E:00 52:00 53:00 54:12
57:00 40:20
s 01.5-0E.5 BY 1.....72:00 73:00 77:00
78:00 79:12 7C:00 40:20
19:00 44:DD 45:AD 46:DA

OUTPOST (SRS)
00.....18:20 19:00 46:96 4D:00
4E:00 52:00 53:00 54:12
57:00 40:20
s 01.5-0D.5 BY 1....72:00 73:00
77:00 78:00 79:12 7C:00
40:20 19:00 44:DD 45:AD
46:DA

Alternate method
00.....18:20 19:00 46:96 4D:00
4E:00 52:00 53:00 54:12
57:00 40:20
s 01.5-09.5 BY 1.....72:00 73:00
77:00 78:00 79:12 7C:00
40:20 19:00 44:DD 45:AD
46:DA

PADDLE GRAPHICS (SOL)
00-22
23.....36:01
Uses nibble count.

PALACE IN THUNDERLAND (ML)
00-22.....25:19

PEGASUS II (SOL)
00-22
03.....4C:1B 57:00 E9:02
Uses nibble count.

Alternate method
00-22.....4E:00
03.....4C:1B 57:00 E9:02
Uses nibble count.

PFS (SPC)
01-13
00.....40:08 41:FF 16:08 19:00
58:0B 59:FF 54:12 12:02
44:93 45:F3 46:FC 47:FF
81:93 82:F3 83:FC 84:FF
(00 error may occur)
Write-protect disk before running.

Alternate method
00-22.....10:04 16:40 46:96 51:00
53:0B 54:12 81:CF 82:F3
83:FC

PFS REPORTS (SPC)
00-13
00.....40:08 41:FF 16:08 19:00
58:0B 59:FF 54:12 12:02
44:93 45:F3 46:FC 47:FF
81:93 82:F3 83:FC 84:FF

Alternate method
00-22
02.....40:08 41:FF 16:08 19:00
58:0B 59:FF 54:12 12:02
44:93 45:F3 46:FC 47:FF
81:93 82:F3 83:FC 84:FF
(after copying write protect before running)

PHANTOMS FIVE (SRS)
00
02-1C.....44:DD

PHOTAR (STP)
s 00-22 BY 1

POOL 1.5 (IDSI)
s 00-15 BY 1
s 1E-21 BY 1

Alternate method
s 00-15 BY 1.....46:B5 79:12
s 1E-21 BY 1

Alternate method
00-15.....21:02
1E-21

PRESIDENT ELECT (SS)
00-22.....25:19 65:00 6B:00

Alternate method
00-22.....25:19 6B:00

PUCKMAN (UNK)
00.....54:12
01-0D.....54:09

PULSAR II (SRS)
s 00
s 1C.5-1D.5 BY 1
s 02-0C BY 1.....44:DD
s 13-19 BY 1
s 1A.5-1B.5 BY 1

QUICK LOADER (SEN)
00
s 02-11 BY 1

RASTER BLASTER (BC)
00.....44:AD 45:DE 53:00
s 05-11 BY 4
s 06-12 BY 4
s 07.5-0F.5 BY 4
s 01.5-03.5 BY 2

Alternate method
00.....46:96 54:12
s 05-11 BY 4.....44:AD 45:DE 46:00
72:00 73:00 75:00 78:00
79:12
s 06-12 BY 4
s 07.5-0F.5 BY 4
s 01.5-03.5 BY 2

RETROBALL (SOL)
00
04-06
09-0C
0E-10
12-14
17-1D
20-22.....4D:00 4E:00

RINGS OF SATURN (SL)
s 00-02 BY 1
03-22
s 05
s 09

SABOTAGE (SOL)
00-22
03.....4C:1B 57:00 E9:02
Uses nibble count.

SARGON II (HN)
00-1A.....19:00 54:12 47:FF 4C:18
48:FF 50:00 51:00 52:00
53:00

Alternate method
00-1A.....19:00 54:12

SCREENWRITER II (SOL)
00-22.....4D:00

SHATTERED ALLIANCE (SS)
00.....25:19 65:00
01-22.....4F:0B

Alternate method
00-22.....25:19

Alternate method
00.....4C:18 47:FF 53:0B 54:12
01-22.....44:D4 46:B5

SHOOT EM UP IN SPACE (UNK)
00-22.....25:19 65:00 6B:00

SINGA SHAPE MANAGER (UNK)
s 00-22 BY 1

SNAKEBYTE (SRS)
00.....18:20 19:00 46:96 4D:00
4E:00 52:00 53:00 54:12
57:00 40:20
s 01.5-0E.5 BY 1.....72:00 73:00 77:00
78:00 79:12 7C:00 40:20
19:00 44:DD 45:AD 46:DA

SNEAKERS (SRS)
00.....18:20 19:00 46:96 4D:00
4E:00 52:00 53:00 54:12
57:00 40:20
s 01.5-0D.5 BY 1....72:00 73:00 77:00
78:00 79:12 7C:00 40:20
19:00 44:DD 45:AD 46:DA

SNOGGLE (PUCKMAN) (BS)
00-09

Alternative method
00-0F
s 10.5-11.5 BY 1

Alternate method
s 00-09 BY 1

SOFTPORN ADVENTURE (SOL)
00-22
03.....4C:1B 57:00 E9:02
Uses nibble count.

SOUTHERN COMMAND (SS)
00-22.....25:19 6B:00 34:D5 35:AB

SPACE EGGS (SRS)
00
02-06
11-13
14-1A.....44:DD

SPACE QUARKS (BS)
00.....18:50 19:00 40:20 46:96
4D:00 4E:00 52:00 53:00
54:12 57:00
01-02.....44:AB 45:D4 46:AB
03.5-05.5 BY 1
07
09.....44:FE 45:DD 46:AF
0A.5-0B.5 BY 1.....44:AA 45:DE 46:BB
0D-15 BY 1

SPACE WARRIOR (BS)
00.....18:50 19:00 40:20 46:96
4E:00 52:00 53:00 54:12
57:00
02.5-03.5.....44:DF 45:AD 46:DE
05-08 BY 3
06.5
0A-10 BY 3

SPY'S DEMISE (PEN)
0-22 BY 1
(Errors on track 11-22 OK)

STAR BLASTER (PDS)
00
s 07-20.5 BY 1.5.....72:00 73:00 77:00 78:00 79:12 7C:00 40:20 19:00 44:DF 45:AD 46:DE

STAR CRUISER (SRS)
S 00-03 BY 3
s 05-0B BY 1
s 11-12 BY 1
s 04.....44:AA 45:DD 46:BB

STAR MINES (STP)
00
01-02.....46:AD
04-0A

STAR RAIDERS (USA)
00-05
(Error may occur on 05)

STAR THIEF (CC)
00-0E
22.....4C:1B 57:00 E9:02
Uses nibble count.

Alternate method
00-13
Errors may occur on 0E-13.
22.....4C:1B 57:00 E9:02
Uses nibble count.

SUPER APPLE BASIC (HN)
00-22
03
Uses extended retry.

SUPERSCRIBE II (SOL)
00-22
03.....4C:1B 57:00 E9:02
Uses nibble count.

Alternate method
00-22
03.....45:00 50:00

TAX PREPARER (HS)
00-22.....46:96 54:12 4C:19

THIEF (DM)
00-22.....83:FF 4F:0B 53:00
s 04-05 BY 1.....38:02 1E:02 19:00 12:01 7C:00

THRESHOLD (SOL)
00-22
01-23 BY
22.....4C:1B 57:00 E9:02
Uses nibble count.

Alternate method
00-22
01.....4C:1B 57:00 E9:02
Uses nibble count.

TIGERS IN THE SNOW (SS)
00-22.....25:19 65:00 6B:00

Alternate method
00-22.....25:19 6B:00

TIME ZONE (SOL)
s 00-04 BY 1
05-22
(Disk sides 1B to 6L, tracks 00-22.)

Alternate method
00-22
Use extended retry. (Sides 1B to 6L, tracks 00-22.)

TINY TROLL (UNK)
00-22
03.5-05 BY 1.5

TORPEDO FIRE (SS)
00
01-22.....4F:0B

TRANSEND 2 (SSM)
S 00-22

TRANSYLVANIA (PEN)
0-22 BY 2
1-21 BY 2.....44:D4 46:96

TWERPS (SRS)
00.....18:20 19:00 46:96 4D:00 4E:00 52:00 53:00 54:12 57:00 40:20
s 01.5-0E.5 BY 1.....72:00 73:00 77:00 78:00 79:12 7C:00 40:20 19:00 44:DD 45:AD 46:DA
1C.....4C:1B 57:00 E9:02 D2:00

Alternate method
00.....18:20 19:00 46:96 4D:00 4E:00 52:00 53:00 54:12 57:00 40:20
s 01.5-0E.5 BY 1.....72:00 73:00 77:00 78:00 79:12 7C:00 44:DD 45:AD 46:DA
s 1C

Alternate method
00.....18:20 19:00 46:96 4D:00 4E:00 52:00 53:00 54:12 57:00 40:20
s 01.5-0E.5 BY 1.....72:00 73:00 77:00 78:00 79:12 7C:00 40:20 19:00 44:DD 45:AD 46:DA
1A.....4C:1B 57:00 E9:02
Uses nibble count.

U-BOAT COMMAND (SY)
00-22.....4E:00 51:00 52:00 40:02 1E:30 1B:19 1D:18 44:00 45:00 46:EB 47:AF

Alternate method
00-22.....4E:00 51:00 52:00 40:02 1E:30 1B:19 1D:18 44:00 45:00 46:EB 47:AF 48:FB 49:EB

ULTIMA (CP)
00-22.....1E:0B

ULYSSES (SOL)
00-22
03.....4C:1B 57:00 E9:02
Uses nibble count. Disk side B tracks 00-22.

VISICALC (VCP)
00-22
NB ignore 01 error.

Alternate method
00-15
NB ignore 01 error.

VISICALC III (VCP)
s 00-22 BY 1

VISIDEX (VC)
00-22.....40:04 16:08 41:FF 19:00 58:0B 59:FF 81:AA 82:EB 83:FD 21:02

Alternate method
00-22.....40:04 16:08 41:FF 19:00 58:0B 59:FF 81:AA 82:EB 83:FD 21:02 46:96 54:12

VISIFILE (VCP)
00-22.....19:01 21:02 58:19 59:06 5A:1A 5B:FF BD:44 BE:EB BF:45 C0:EC C1:40 C2:01 C4:44 C5:D5 C6:45 C7:AA C8:40 C9:04 CA:00

Alternate method
00-22.....19:00 21:02 58:19 59:06 5A:1A 5B:FF BD:44 BE:EB BF:45 C0:EC C1:40 C2:01 C4:44 C5:D5 C6:45 C7:AA C8:40 C9:04 CA:00

VISISCHEDULE (VCP)
00-22.....40:04 16:08 41:FF 19:00 58:0B 59:FF 81:AA 82:EB 83:EC 21:02 46:96 54:12

Alternate method
00-22.....40:04 16:08 41:FF 19:00 58:0B 59:FF 81:AA 82:EB 83:EC

VISITERM (VCP)
00-22
06.....40:08 16:08 41:FF 19:00 58:0B 59:FF 81:AA 82:EB 83:FC

VISITREND/VISIPLOT (VCP)
00-22
07.....40:08 16:08 41:FF 19:00 81:DE 82:AA 58:0B 59:FF

WARP FACTOR (SS)
00
01-22.....4F:0B

WIZARD AND THE PRINCESS (SOL)
s 00-22 BY 1

WIZARDRY -1 (Proving Ground) (SIR)
00-09
0F-22
s 0A-0E BY 1.....36:01
Uses nibble count.
Write protect before running.

Alternate method
00.....36:01 21:02 46:96
Uses nibble count.
Write protect before running.
s 01-22 BY 1.....36:00

Locksmith Parameters continued on page 46

A step-by-step guide to making back ups using

Nibbles Away II Parameters

Nibbles Away II is a product of Computer Applications, Inc.

There are three basic steps to back up a diskette:

1. Locate the tracks which contain data.

2. Find the address marker for the sectors there.

3. Determine if any additional protection is used (this is the hard one!).

TRACK/BIT EDITOR

For most of the procedures below, a basic working knowledge of the track/bit editor (TBE) is required. For those who are not familiar with the TBE, an overall description and some examples are given below.

The examples are easier to understand if they are performed while reading the instructions, so boot Nibbles Away][and try them out to get a better understanding of what is going on.

Enter the TBE by selecting option T from the main menu. A large section of numbers will appear on the screen with two dashed lines at the top. The information between these lines is the status information. It shows such things as cursor position and track number. It is also the location where various prompts appear for certain functions. The numbers at the bottom are separated into two sections. On the left are the starting memory addresses for each line to the right.

Move the cursor around using I, J, K or M and watch the ADDR indicator in the status line. It will show exactly what memory address the value under the cursor represents.

The arrow keys change the area of memory that can be seen. They shift the view 256 bytes forward or backward at a time. The only really important thing to know for this discussion is how to use the arrow keys to move the viewing 'window" around in memory.

The semicolon (unshifted plus) and the dash (-) keys increment and decrement the track number in the status line.

Pressing R will cause drive one to read the data from the track indicated in the status line into memory. The bytes on the screen will change since different data has been read. Pressing the R key multiple times will result in different data being displayed. This is because Nibbles Away][starts reading at whatever point happens to be under the read/write head when the drive is turned on. The data is not actually different; it is just not loaded at the same memory location as it was previously.

Step 1: Locate the Tracks with Data

To begin, the track pointer should be set to track $00. Pressing R will read the track and show it on the screen. The arrow keys should be used to move the viewing "window" to start at $2000.

Now move forward and try to determine if this track contains valid data. Actually, track $00 must contain some data in order for the disk to boot, but we will be using this procedure on other tracks which do not necessarily contain data.

GAPS

The main thing which will identify a track as containing data is the presence of gaps. Gaps are sections of the same byte repeated several times. Normally they are made up of $FFs and are 6-20 bytes in length. To see what these look like, insert the System Master disk and read track $00 as described above.

Moving through the buffer with the arrow keys will reveal a large variety of values. Spaced among these should be sections of 6-20 FFs in a row, depending on the exact disk. Normally DOS 3.2 disks have larger gaps than DOS 3.3 disks. There should be many gaps, spaced so that one is seen about every other time the arrow keys are used to move forward or backward.

Note: A second, smaller (2-5 $FFs) gap may be seen following a large gap, with a small section of data in between. This is called the secondary gap. When referring to a gap here, the allusion will be to the primary gap, not the secondary one.

FULL/HALF TRACKS

Now try looking at other tracks on the disk. First look only at the full tracks (no ".5" on the end). All of them will be similar to track $00 in the appearance of the gaps. Try this several times to become comfortable with locating gaps on a given track.

Now read a half track (".5" on the end). Scan memory to locate some of the gaps. Since System Master disks do not use half tracks, the data which is seen is really "cross-talk." In other words, data was written on the full track, but the magnetic pattern spread out a bit, so some data is seen here. The telltale sign of this phenomenon is that the gaps will not all be the same. That is, they may contain one or more values which are inconsistent. This reveals that there is some data on the track but that it is not valid data. Take a look at other half tracks until half tracks and full tracks can be discerned by examining the gaps.

BLANK TRACKS

The next item to be able to identify is a blank track. To do this, insert a blank (noninitialized) disk into drive 1. Read any track on this disk and scan through the memory addresses. There will be no gaps, and many of the bytes will end in 0 (ie. $A0, $B0, $E0), which are not legal disk bytes. This means that the controller can find no valid data on the track. Some disks have portions of tracks which are not used, so always be sure to examine at least 24 screens full of information to make sure that there is no data at any point on the track.

The next tool for finding data is a result of the fact that valid data must be at least one track apart. In other words, if data is located on track 3.5, track 4 cannot have data and the next place where data can be is track 4.5. This is very helpful for finding tracks with data.

Note: If data is located on a given track, it is a good idea to look at the tracks one half track to either side to make sure that they look less valid than the track selected as the real one.

Now that valid data can be recognized, begin at track $00 and step towards track $22, checking each track to see if it appears to have data on it. Most disks have a pattern to the position of the data, and if that pattern can be figured out it may be possible just to check a few tracks to make sure and then go on to step 2. Otherwise, the data must be located one track at a time.

Most disks use the standard tracks (1,2,3,. . .22), but there are some which use half tracks and some which use track $23 (which cannot be read on all drives since Apple drives were not designed to go out that far).

When all tracks which contain some type of data are located, go on to step 2.

Step 2: Find the Address Markers

Now tell Nibbles Away][how to read the information on the tracks which have been found to contain valid data. This is done by going back to each of these tracks with the TBE and finding the address mark for each one. The address mark will be the first three bytes following the gap. To see this in operation, take a look at a track from the System Master disk. After each gap either D5 AA 96 for a DOS 3.3 master disk, or D5

AA B5 for a DOS 3.2 disk will be seen. These values should be noted alongside of each track number which contains data. Many times there will be only one, or maybe two, patterns for all tracks.

After this, these tracks can be copied. This is done by exiting the TBE (use Q) and then selecting M for the modifier menu. Then select B for back-up modifier. When asked USE ADDRESS MARK?, answer Y and then type in the address mark that was noted for the range of tracks to be copied. Simply press return to the rest of the questions and then return to the main menu.

Select N to enter Nibbles Away][, and answer Y to the question CHANGE DEFAULT OPTIONS? Use the return key to move to the START TRACK prompt, and then enter the first track to be copied. Press return and then type in the last track to be copied with the current address marker setting.

If the tracks in the specified range are not spaced at 1-track intervals, enter the interval at the TRACK INCREMENT prompt. Press return for the following questions, and begin the copy after inserting the disks (when prompted). After returning to the main menu, repeat the above procedure for each range of tracks which contains a different address marker.

Now comes the moment of truth! Try to boot the copy disk. (If the original had a write-protect tab, the copy should too!) If the copy boots, then all went successfully.

Step 3: Find Additional Protection

If the back-up did not work properly, there are a few things to look for.

1. Did all of the tracks which should have copied do so? This can be seen while the copy takes place as a Y or an N under that track status location. If some did not, then the address marker was probably not determined properly. If this is the case, go back to the TBE and try those tracks again.

2. If everything seemed to go well but the copy refuses to work (it might help to try the procedure again, maybe with the source and destination drives reversed, to make sure it was not a power glitch or other such occurrence which messed things up), the next step is to try the procedure with the synchronized copy option selected. Disks which use this method often make violent head movements during their boot procedure. This can be a clue to this type of protection.

Additional Information

On some DOS 3.3 diskettes, the gaps between the sectors are reduced in size. In some cases they can be as small as four or five bytes. When Nibbles Away][finds the beginning of a section of data, it normally adds eight bytes of sync just before the data. This will normally put sync bytes into the gap before the data, where it should be. However, if a disk has very small gaps, then the added sync can overwrite the end of the previous sector. The parameter FIX AMNT controls the number of sync bytes which are added, so this value can be reduced to prevent any data from being overwritten. The value that Nibbles Away][uses for the sync which it puts in is contained in the parameter FIX VALU. Normally this is a $7F, but it can be set to any desired value.

It should be noted that Nibbles Away][regards any data byte which has its high bit cleared to be a sync byte. So the $7F which is normally in this parameter means that a sync $FF is to be added. If the override standardizer option is selected, then Nibbles Away][will not add any bytes; it will simply convert the data which is present before a sector into sync without changing its value. This technique can also be used for disks whose gaps are very small.

LONG TRACKS

Another item to watch for is disks whose tracks appear to be very long. Some disk protection schemes put garbage on a portion of the track. When this garbage is read back, more bytes are read in than were written out. This causes the track to be longer than normal, and in some cases it becomes so long that the default parameters for Nibbles Away][cannot find the data properly.

DATA MIN/MAX

The parameters DATA MIN and DATA MAX control the minimum and maximum track lengths (in increments of 256 bytes) which Nibbles Away][will accommodate. The normal value of DATA MAX is $1D, but this can be set to a higher value, such as $25 if a track appears to be very long. Even though the track may read a large number of bytes, many of these will be removed by the nibble filter since they are garbage bytes. This will assure that the amount of data written back will not be too large to fit on the destination track.

When Nibbles Away][finds a sector of data, it looks ahead to find a second occurrence of the same pattern. This insures that the sector has been read and located correctly. On many disks, there is a primary section of data called the address field, and the actual data field follows. In between these is a small gap which often contains random information. This means that Nibbles Away][should only match the number of bytes which are found in the address field since the bytes in the gap may not read as the same value every time.

FIND MAX

The parameter FIND MAX controls the number of bytes which are checked during this procedure. The default value of $0C works in most cases, but some disks use a smaller address field which may require this parameter to be set to a smaller value. However, if this parameter is set too low, then Nibbles Away][may identify the match for a section of data whose first few bytes are the same, but which differ later on. Therefore, one should exercise caution when lowering this value.

How to make parameter changes

Listed below are the parameters to change in order to back up certain pieces of software which require more than the default values given with Nibbles Away][. If a number is listed within the "less than" (<) and "greater than" (>) signs, it corresponds to the number of the auto-load file which will perform the listed function. To use the auto-load files, see Chapter 6 in the Nibbles Away][Manual.

To back up a program, first find its name in the list of parameters. Directly across from the file name is the auto-load file to use if one exists. Remember that auto-load files are within the "less than" and "greater than" signs.

Directly below the name is a list of the tracks to copy and what parameter changes to make. If the letter 'S' appears to the far left of the track number, set the synchronization mode before copying these tracks. If the word BY is used, set the increment to this value; otherwise, use the default increment of 1. Parameters which are assigned values can be accessed under the control parameter modifier. The parameters ADDR and INS should be entered under ADDRESS MARK and INSERT MARK, respectively, in the back-up modifier.

When the word SECTMOD appears below, it means that a sector should be changed using the track/sector editor (TSE). Place the destination disk into drive 1; then perform the changes listed. The command format is:

SECTMOD [.F = nn,C = xx,S = yy,T = zz]
CHANGE ADDRESS A1 FROM A2 TO A3

The meaning of nn, xx, yy, zz and A1, A2, A3 are explained below:

nn-This will be either 13 or 16 and represents the disk format to be used. This should be set by selecting the 'O' option in the TSE, then pressing 'F' until the proper format is shown in inverse.

xx-This will be either on or off and should be set using the checksum option on the options page, as above ('C' to toggle).

yy-This is the sector to be read.

zz-This is the track to be read. (See Nibbles Away][Manual for details on how to set these.)

After setting these options, use the 'R' option to read the given sector into the buffer. Then change the information in the sector, following the conventions listed below:

A1-This is the location to be changed in the buffer.

A2-This is the old value.

A3-This is the new value.

If multiple changes are listed, they should be performed in sequence. After making changes to a sector, it should be written back to the disk with the W option.

Note: Parameters from Nibbles Away I may be used in Nibbles Away][. They must be entered using the name of the desired parameter listed in the Nibbles Away I Manual. Nibbles Away I parameters may not be entered under the global modifier in Nibbles Away][.

The following example shows a file that incorporates many changes. Step-by-step directions will be given on how to copy this program. Line numbers have been added to each line for reference. These numbers do not appear in the parameter list.

```
1) EXAMPLE FILE (XXX)
2) 0-5 . . . . . ADDR = D5 AA 96
3)       SECTMOD [F = 16,C = OFF,T = 00,S = 03]
4)       CHANGE ADDRESS 42 FROM 38 TO 18
5)       OVERIDE STANDARDIZER
6) S 6-9 BY 1.5 . . . . ADDR = DD AD DA
8)   11-22 BY 2 . . . . INS = AD FB E6 FF E6
9)                      SYNC SIZ = 0A
10)                     DATA MAX = 25
```

Line 1. The name of the program is "EXAMPLE FILE". The abbreviation of the company that markets the program is in parentheses (XXX).The name of the company can be found in the table of abbreviations.

Line 2. The tracks to copy are given, along with the parameter changes to make before copying the tracks. First set the ADDR parameter to D5 AA 96. These changes are made from the back-up menu. To get to the back-up menu first enter the modify parameter menu ('M' from the main menu); then press 'B'.

Line 3. Some special changes must be made to a certain sector (track 00, sector 3). Enter the TSE and use the 'O' command to select the following options:

F Set up for 16-sector.

C Turn the checksum flag off.

Track 00, sector 3 needs to be read. Type T, then 00, then S, then 3. Use the R command to read the proper track/sector.

Line 4. After the sector is in memory, change address 42 from 38 (what it originally was) to 18. Write the track back to the disk using the W command.

Line 5. Enter the back-up menu. From this menu answer yes (Y) to the question OVERIDE STANDARDIZER?

Line 6. Set the synchronized copy mode when copying tracks 6-9. The S means to make a synchronized copy. Also set the increment to 1.5.

Line 7. Before copying tracks 6-9, set the ADDR to DD AD DA.

Line 8. Copy tracks 11-22 with an increment of 2. Set INS to AD FB E6 FF E6. The changes to INS are made from the back-up menu.

Line 9. Set the SYNC SIZe to 0A. To change SYNC SIZes enter the modify parameter menu, and from this menu enter the control menu (C). Use the arrow keys to move the cursor to the SYNC SIZe (second column, third one down) and press the space bar to change this value

Line 10. The DATA MAX value is changed in the same manner as the SYNC SIZe.

ACCOUNTING SYSTEM (BPI) <15>
0-22 . . . . . ADDR=D5 AA 96
11-11 . . . . . INS=AD FB E6 FF E6
SYNC SIZ=0A

APPLE PANIC (BS)
0-D

APPLE WORLD (USA)
0- 23

APPLE WRITER /// (APC)
s 0-22

AUTOBAHN (SRS) <6>
s 0-0
s 4-6
s 9.5-C.5

A2-FS-1 (SBL) <13>
0-0
1.5-21 BY 1.5 . . . . . ADDR=DB AB BF
REDUCED ERROR CHECK
7-8 . . . . . REDUCED ERROR CHECK
9.5-9.5 . . . . . REDUCED ERROR CHECK

A2-PB1 PINBALL (SBL)
0-0 . . . . . ADDR=D5 AA 96
DATA MAX=25
1-15 . . . . . ADDR=DB AB BF

BEER RUN (SRS)
s 0-0 . . . . . ADDR=DD AD DA
DATA MAX=25
s 1.5-13.5
NOTE: Errors will begin to occur between track C.5 and track 13.5. This is normal.

BORG (SRS)
s 0-0 . . . . . ADDR=DD AD DA
s 1.5-B.5
s D-20

CANNONBALL BLITZ (OLS)
0-22 . . . . . ADDR=D5 AA 96
SECTMOD [F=16,C=ON,T=17,S=0E]
CHANGE ADDRESS CD FROM 49 TO 60

CASINO 21 (DM)
0-22 . . . . . ADDR=D5 AA 96
SECTMOD [F=16,C=OFF,S=03,T=00]
CHANGE ADDRESS 63 FROM 38 TO 18

CEILING ZERO (TKS)
0-2 . . . . . ADDR=D5 AA B5
3-11 . . . . . ADDR=D6 AA B5
INS=DE AA EB F9, SYNC
SIZ=0A

COPTS & ROBBERS (SRS)
Same as Beer Run

COUNTY FAIR (DM)
0-22 . . . . . ADDR=D5 AA B5

CRANSTON MANOR (OLS)
0-22 . . . . . ERASE DEST TRACKS

DARK FOREST (SRS)
s 0-0 . . . . . ADDR=DD AD DA
s 1-22 . . . . . ADDR=D5 AA A5
(Errors on 6-8 and last few tracks OK)

DB MASTER (OLD) (STW) <9>
0-5 . . . . . ADDR=D5 AA 96
6.5-22.5

DB MASTER (NEW) (STW) <19>
s 0-5 . . . . . ADDR=D5 AA 96
6.5-22.5

DEADLINE (IC) <DOS 3.3>
0-22 . . . . . ADDR=D5 AA 96

DESKTOP PLAN II (VCP) <16>
0-22 . . . . . ADDR=D5 AA 96
INS=AA EB FD
SYNC SIZ=0A, FIX AMNT=04

DUNG BEETLES (DS)
0-0 . . . . . ADDR=D5 AA B5
1-1 . . . . . ADDR=F5 F6 F7
4-22
SECTMOD [F=13,C=ON,T=00,S=01]
CHANGE ADDRESS 6D FROM 01 TO 7B
CHANGE ADDRESS 6E FROM 61 TO 69

ELIMINATOR (AI)
0-21 . . . . . ADDR=D5 AA 96
SECTMOD [F=16,C=OFF,T=03,S=0D]
CHANGE ADDRESS 2E FROM 20 TO EA
CHANGE ADDRESS 2F FROM 30 TO EA
CHANGE ADDRESS 30 FROM 72 TO EA

EPOCH (SRS) Same as Beer Run

ESCAPE (SBL) <DOS 3.3>
0-22 . . . . . ADDR=D5 AA 96

ESCAPE FROM ARCTURUS (SNS)
0-22 . . . . . ADDR=D5 AA 96
OVERIDE STANDARDIZER
OVERIDE NIBBLE FILTER

EXECUTIVE SECRETARY (PBS)
0-22 . . . . . ADDR=D5 AA 96
DOS 3.3

EXPEDITER][(OLS) <2>
0-22 . . . . . ADDR=D5 AA 96
ERASE DEST TRACKS

FIREBIRD (GS) <7>
s 0-0 . . . . . ADDR=DD AD DA
s 1.5-B.5

GAMMA GOBLINS (SRS)
s 0-0 . . . . . ADDR=DD AD DA
s 1.5-B.5
s D-D . . . . . ADDR=FF FF FF D5 AA EE
DATA MAX=30

GENETIC DRIFT (BS)
0-0 . . . . . ADDR=D5 AA B5
1-3 . . . . . ADDR=BB D5 BB
4.5-6 BY 1.5
7.5-B.5
D-D . . . . . ADDR=D4 D5 BB
E.5-12.5 . . . . . ADDR=AD B5 DE

GOBBLER (OLS) <1>
0-22 . . . . . ADDR=D5 AA B5
ERASE DEST TRACKS

GOLD RUSH (SS) <DOS 3.3>
0-22 . . . . . ADDR=D5 AA 96

GORGON (SRS)
s 0-0 . . . . . ADDR=DD AD DA
DATA MAX=25
s 1.5-C.5
s E.5-E.5
s D.5-D.5 . . . . . ADDR=D5 AA B5

GUARDIAN (CTS)
0-1 . . . . . ADDR=D5 AA B5
2-11 . . . . . ADDR=D6 AA B5
INS=DF AA EB F7, SYNC
SIZ=0A

HADRON (SRS) Same as Beer Run

HIRES ADV -1 (OLS) <DOS 3.2>
0-22 . . . . . ADDR=D5 AA B5

HIRES ADV -2 (OLS) <DOS 3.2>
0-22 . . . . . ADDR=D5 AA B5

HIRES CRIBBAGE (OLS) <20>
s 0-22.....ADDR=D5 AA B5

INTERNATIONAL GRAND PRIX (RBS)
0-C.....ADDR=FF FF FF AA

INVOICE FACTORY (ML) <DOS 3.3>
0-22.....ADDR=D5 AA 96

JABBERTALKY (MT) <DOS 3.3>
0-22.....ADDR=D5 AA 96

JAW BREAKER (OLS) <1>
0-22.....ADDR=D5 AA B5
ERASE DEST TRACKS

LETTER PERFECT (LJK) <DOS 3.2>
0-22.....ADDR=D5 AA B5

MASTER TYPE (LNS)
0-2.....ADDR=D5 AA B5
3-22.....ADDR=D4 AA B5
(ERROR ON $1B OK)
SECTMOD [F=13,C=OFF,S=03,T=00]
CHANGE ADDRESS 63 FROM 38 TO 18
SECTMOD [F=13,C=OFF,S=0A,T=02]
CHANGE ADDRESS 2E FROM 23 TO 2E

MICROWAVE (CC)
0-22.....ADDR=D5 AA 96
SECTMOD [F=16,C=ON,T=02,S=01]
CHANGE ADDRESS DA FROM A9 TO AD
CHANGE ADDRESS DB FROM 60 TO 03
CHANGE ADDRESS DC FROM 8D TO 81
CHANGE ADDRESS DD FROM 7E TO 60

MISSILE DEFENSE (OLS) <20>
s 0-22.....ADDR=D5 AA B5

MOUSKATTACK (OLS)
0-22.....ADDR=D5 AA 96
SECTMOD [F=16,C=ON,T=18,S=03]
CHANGE ADDRESS B1 FROM 49 TO 60

NEUTRONS (L10) <1>
0-22.....ADDR=D5 AA 96

ORBITRON (SRS)
s 0-0.....ADDR=DD AD DA
DATA MAX=25
s 1.5-E.5
F.5-F.5.....ADDR=FF B5 D5 AA

OUTPOST (SRS)
s 0-0.....ADDR=DD AD DA
s 1.5-9.5
B.5-B.5.....ADDR=D5 AA AD
DATA MAX=25

PADDLE GRAPHICS (OLS) <20>
0-23.....ADDR=D5 AA B5
s 2.....ADDR=D5 AA B5

PEEPING TOM (ML)
0-0.....ADDR=D5 AA B5
1-1.....ADDR=F5 AB BE
4-22
SECTMOD [F=13,C=ON,T=00,S=01]
CHANGE ADDRESS 6D FROM 01 TO 7B
CHANGE ADDRESS 6E FROM 60 TO 68

PEGASUS II (OLS) <1>
0-22.....ADDR=D5 AA B5
ERASE DEST TRACKS

PERSONAL FINANCE MGR. (SDS) <DOS 3.3>
0-22.....ADDR=D5 AA 96

PHOTAR (STP) <DOS 3.3>
0-22.....ADDR=D5 AA 96

POOL 1.5 (IDS)
0-15.....ADDR=D5 AA B5
1E-21
SECTMOD[F=13,C=OFF,T=0B,S=07]
CHANGE ADDRESS 6A FROM 8D TO 60
SECTMOD[F=13,C=OFF,T=00,S=03]
CHANGE ADDRESS 63 FROM 38 TO 18

PULSAR][(SRS) <18>
0-C
13-19
1A.5-1D.5

RASTER BLASTER (BC)
s 0-0.....ADDR=D5 AA 96
DATA MIN=18, DATA MAX=40
s 5-11 BY 4.....ADDR=AD DE, DATA MIN=13
s 6-12 BY 4
s 7.5-F.5 BY 4
s 1.5-3.5 BY 2

RICOCHET (MT) <DOS 3.3>
0-22.....ADDR=D5 AA 96

ROACH HOTEL (ML)
0-0.....ADDR=D5 AA B5
1-1.....ADDR=EE EA FE
4-22
SECTMOD [F=13,C=OFF,T=00,S=01]
CHANGE ADDRESS 75 FROM 01 TO 7B
CHANGE ADDRESS 76 FROM 61 TO 69

SNACK ATTACK (DM)
0-22.....ADDR=D5 AA B5
SECTMOD [F=13,C=OFF,S=03,T=00]
CHANGE ADDRESS 63 FROM 38 TO 18

SNAKE BYTE (SRS) Same as Beer Run

SNEAKERS (SRS)
s 0-0.....ADDR=DD AD DA
s 1.5-C.5
s D.5-D.5.....ADDR=D5 AA B5

SOFTPORN ADVENTURE 3.2 (OLS) <1>
0-22.....ADDR=D5 AA B5
ERASE DEST TRACKS

SOFTPORN ADVENTURE 3.3 (OLS) <2>
0-22.....ADDR=D5 AA 96
ERASE DEST TRACKS

SPACE QUARKS (BS)
0-0.....ADDR=D5 AA B5
1-2.....ADDR=FF DF DE
DATA MAX=25
3.5-5.5
7-9 BY 2
A.5-B.5
D-15

SPACE WARRIOR (BS)
0-0.....ADDR=D5 AA B5
DATA MAX=30
2.5-3.5.....ADDR=DF AD DE
5-8 BY 3
6.5-6.5
A-10 BY 3

STAR BLASTER (PDS)
0-0.....ADDR=D5 AA 96
7-20 BY 1.5.....ADDR=DF AD DE

STAR DANCE (USA) <DOS 3.2>
0-22.....ADDR=D5 AA B5

SUICIDE (PDS)
0-0.....ADDR=D5 AA B5
11.5-22 BY 1.5.....ADDR=DF AD DE

SWASHBUCKLER (DM)
0-22.....ADDR=D5 AA 96

TAX PREPARER (HS) <DOS 3.3>
0-22.....ADDR=D5 AA 96

THRESHOLD (OLS) <1>
0-22.....ADDR=D5 AA B5
ERASE DEST TRACKS

TIME ZONE V1.0 (OLS)
DISKS A-L
0-22.....ADDR=D5 AA 96
OVERIDE STANDARDIZER THEN DISK A
SECTMOD [F=16,C=ON,T=03,S=05]
CHANGE ADDRESS 5B FROM 4C TO 60
SECTMOD [F=16,C=ON,T=03,S=03]
CHANGE ADDRESS AB FROM A9 TO 60

TORPEDO FIRE (SSM)
0-22.....ADDR=D4 AA B7

TUNNEL TERROR (MS)
0-0.....ADDR=D5 AA B5
1-12.....ADDR=D6 AA B5
INS=DF AA D7 EB, SYNC SIZ=0A

TWERPS (SRS)
s 0-0.....ADDR=DD AD DA
s 1.5-E.5
1A-1A

ULYSSES & GOLDEN FLEECE (OLS) <2>
0-22.....ADDR=D5 AA 96
ERASE DEST TRACKS

VISICALC /// (APC)
s 0-22

VISICALC 3.3 (VCP)
0-0.....ADDR=D5 AA 96
2-22.....ADDR=D5 AA B5
(ERRORS TOWARD END OK)

VISIDEX (VCP)
0-22.....ADDR=D5 AA 96
INS=DE AA EB FD
SYNC SIZ=0A, FIX AMNT=04

VISIFACTORY (ML)
0-22.....ADDR=D5 AA 96
SECTMOD [F=16,C=OFF,T=00,S=03]
CHANGE ADDRESS 42 FROM 38 TO 18
SECTMOD [F=16,C=0FF,T=01,S=00]
CHANGE ADDRESS 84 FROM 4C TO AD
CHANGE ADDRESS 85 FROM 8E TO E9
CHANGE ADDRESS 86 FROM AE TO B7

VISIFILE (VCP)
0-22.....ADDR=D5 AA 96
INS=DE AA EB
SYNC SIZ=0A, FIX AMNT=04

VISISCHEDULE (VCP)
0-22.....ADDR=D5 AA 96
INS=DE AA EB
SYNC SIZ=0A, FIX AMNT=04

VISITERM (VCP)
0-22.....ADDR=D5 AA 96
INS=DE AA EB FC
SYNC SIZ=0A, FIX AMNT=04

VISITREND/VISIPLOT (VCP)
0-22.....ADDR=D5 AA 96
INS=DE AA EB
SYNC SIZ=0A, FIX AMNT=04

WORD HANDLER II (SVS)
0-0.....ADDR=D5 AA 96
11-22
1-C.....ADDR=FF DF DE

ZERO GRAV. PINBALL (AGC) <DOS 3.2>
0-22.....ADDR=D5 AA B5

Technical notes and making back ups using

Back-It-Up II + Parameters

Back-It-Up II + is a product of Sensible Software, Inc.

A great variety of schemes are used by software houses to copy-protect their diskettes. Two methods (synchronized tracks and bit insertion) are discussed below, along with instructions for copying disks that employ these methods.

Syncronized Tracks

Some software houses have greatly increased the accuracy of their synchronization requirements, which can be done if the tracks are not erased before each write cycle. To do this, change parms 0F and 10 to 01. Also, changing parm 0C to x8 improves sync accuracy (where x = any value).

Bit Insertion

It has become common practice to test for a nibble copy by writing one or more of the bit slip marks out at the self-sync timing rate. To copy these disks, change:

parm 11:00
parm 15:30 (or 60; try both if necessary)
parm 17:03
parm 2A:0A
parm 18-1A to whatever the address marks are as shown in the upper left-hand corner of the screen.

NOTE: Any time parm 11 is changed to 00, it is likely that you must also compress tracks.

The instructions for copying disks protected by bit insertion must be taken in order or they will not produce a workable copy.

Peeking at the Write Protect Tab

There are also some copy-protection methods which involve looking to see if the diskette is write-protected. If the diskette needs to be write-protected, this will be indicated at the end of the parm listing.

How to Read the Parameter List

Look for your program in the list. (The names are in alphabetical order.) On the far left-hand side of the listing is a letter which corresponds to one of the following required changes:

S - set synchronization mode.
D - set decode mode.
P - set compressed mode.
N - set normal mode.
C - set nibble counting mode.

There may be the word BY following the track number. This is the value of the increment. If no value appears here, use 1.

Next will appear the parameter changes for copying the given tracks. The format is:

parm : value

where **parm** is the parameter number to change and **value** is the number to set the parameter to.

The dash (-) is used to indicate a range. The notation 04-0A:00 means to set all of the parms from 4 through and including 0A to 00. The same holds true for instructions such as copy tracks A-E. This means copy tracks A, B, C, D and E.

Remember: All disks must be copied in the given order.

A Sample Listing

1) INVADERS
2) D 0
3) S 1-10 BY 3 00:FC 01:EE 09-0A:BB
4) P 2.5

1) The program name is Invaders.

2) Set mode to decode before copying track 0.

3) Set the mode to synchronized and change parameter 00 to FC, 01 to EE and 09 through 0A to BB. Set the increment to 3 and copy tracks 1 thru 10.

4) Set the mode to compressed and copy track 2.5

One final note: Before writing to ask about copy instructions that apparently do not work, ensure you have done the following:

I. Be certain you followed the copy instructions in order.

2. Be certain you followed all of the instructions, such as write-protecting the copy before using it.

3. Be certain you are using high quality diskettes to make the copy. You can be certain the original was. 16-sector programs require double density diskettes.

4. If the status line shows write errors, compress the track. This will compensate for drive speed to some extent.

5. Try swapping drives; that is, put the original in drive 2 and the copy in drive 1. Make certain you write-protect the original; otherwise, you might accidentally write to it and destroy it.

6. Try putting the copy on another diskette. You will encounter defective disks in even the best brands.

7. Try the copy on at least one other set of drives. Some drives are better or worse than others.

8. Try changing parm 25 to 01. This reduces error checking a little bit but usually does no harm to the copy.

A I HI-RES GRAPHIC ADVENTURES
N 0-21
C 22.....16:FF

AKEM-STONE
D 0-22

ALIEN RAIN
N 0-5
N 6-F.....00:FE 0D:21 0E:00

APPLE ADVENTURE
N 0-22.....00:FE

APPLE PANIC
N 0-5
N 6-D.....00:FE

APPLEWRITER ///
S 0-22
Note: On ver. 2.1 change 11:02

ASTEROID FIELD
N 0-12
Note: You may need to compress tracks 0-2 and 11

A2-PB1 PINBALL
N 0-15

AUTOBAHN
S 0.....00:FE
S 4-6
S 9.5-C.5

BAG OF TRICKS
C 0-14.....00:FE 11:00 15:40 17:02 18:FE 19:FD

BEER RUN
S 0
S 1.5-D.5

BILL BUDGES SPACE ALBUM
N 0-11

BILL BUDGES 3-D GRAPHICS PACKAGE
N 0-2
N 4-8
N 11-18

BORG
N 0
DS 1.5-B.5.....00:FE 01:DD 02:AD 03:DA 04-0A:00 0B:0C

DS D-14

BPI GEN LGR, ACC RCV, & PAYROLL
N 0-10
N 12-22
N 11.....11:00 15:18 17:04 18:AD 19:FB 1A:E6 1B:FF

BUG ATTACK Note: requires version 2.2 or later
N 0-12
SD 1D-22 BY 5.....00:FE 01:AA 02:D5 03-0A:00 0B:0C 0F:01 10:01 16:FF 22:02 24:02 25:01 26:02 28:02
Note: D-speed must be adjusted for nibble count.

CANNONBALL BLITZ
N 1-22
D 0.....0B:06 05-0A:00 16:FF

CASINO
D 0-22.....00:DB 11:00 15:C0 17:08
19:AA 1E:DB 1F:DB

CASTLE WOLFENSTEIN
D 0-22.....00:FE

CASTLES OF DARKNESS
N 0-22.....00:DB

COPTS & ROBBERS
D 0
SD 1.5-F.5.....00:FE 02:AD 03:DA 04-0A:00
0B:0C

COUNTY FAIR
N 0-11.....11:00 15:40 17:02 18:AF
1E:DE

CRANSTON MANOR
N 0-17
N 19-22
N 18.....13:1F

CROSS CLUES
D 0-22.....14:FE 15:60 17:04 19:BF
1B:AA

CROSSFIRE
S 0-22

CROSSWORD MAGIC
SD 0-22.....00:FE

THE CROSSWORD MACHINE
N 0-22

CRUSH CRUMBLE AND CHOMP
D 0-22

CYBER STRIKE
N 0
N 3-B
N 11-1C

CYBORG
D 0
D 1-22.....09:EB

CYBORG (new version)
DS 0-22.....00:FE 03:00 05-0A:00 0B:0C

DARK FOREST
D 0
SD 1-5.....00:FE 03:A5 04-0A:00 0B:0C
SD 9-1F

DB MASTER (NEW)
S 0-5.....00:FE
S 6.5-22.5
Note: Do NOT use the decode option on track 1.

DB UTILITY PACK
Same as DB MASTER

DESK TOP PLANNER
D 0-22.....11:00 15:40 17:03 18:AA
19:EB 1A:FD

DIC'-TIO-NARY
N 0-1F
D 20.....0B:05 16:FF

DOG FIGHT (OLD)
N 0-1
N 4-10

DRAGON FIRE
N 0-22
Note: This is a 16 sector diskette.

ELECTRIC DUET
N 0-22.....11:00 15:60 17:02 18:DE
19:AA

ELIMINATOR
N 0-20
N 22
C 21.....16:FF

EPOCH
N 0
N 1.5-F.5

EXPEDITER
N 0-22
N 3.....13:1D
N 1D

E-Z DRAW
DS 0-22

FALCONS
N 0
D 1.5-4.5 BY 1.5.....00:FE 01:DF 02:AD 03:DE
04-0A:00 0B:0C
D 5.5
D 7-A
D 10-12
D 13.5-14.5
D 16-17.5 BY 1.5
D 19-1A
D B.5

FIREBIRD
SD 0
S 1.5-B.5.....00:FE 01:DD 02:AD 03:DA
04-0A:00 0B:0C

FLIGHT SIMULATOR
N 0-22
N 0-21 BY 1.5
N 7-8
N 9.5

FOOSBALL
D 0
SD 1.5-9.5.....00:FE 01:DD 02:AD 03:DA
04-0A:00 0B:0C
D B-22.....25:02 26:03 27:03
SD A.....0F:01 10:01 12:7E 21:17
25-27:04

GAMMA GOBLINS
DS 0
DS D
DS 1.5-B.5.....00:FE 01:DD 02:AD 03:DA
04-0A:00 0B:0C

GENETIC DRIFT
D 0
D 1-3.....00:FC 01:BB 02:D5 03:DD
04-08:00 09:BB 0A:BB 0B:0C
D 4.5-6 BY 1.5
D 7.5-B.5
D D.....01:D4
D E.5-22.5.....01:AD 02:B5 03:DE

GENERAL MANAGER
N 0-22
N 4.....13:1D

Alternate method
N 1-22
DC 0.....05-0A:00 0B:06 16:FF

GOLD RUSH
DS 0-22.....00:FE 03:00 05-0A:00 0B:0C

GORGON
N 0
S 1.5-F.5
Note: New versions may use SNEAKERS parms

HELL FIRE WARRIOR
D 0-22

HI RES CRIBBAGE
N 0-2
N 4-22
N 3.....13:1F

HI RES ADVENTURE -2
N 0-22

HUNGRY BOY
D 0
N 1.5.....00:FE 01:AD 02:DE 03:BE
04-0A:00 0B:0C
N 3.5-13.5 BY 4
N 5-15 BY 4
N 6-16 BY 4

INTERNATIONAL GRAN PRIX
N 0
N 2
N 4-C
N 11

INVENTORY MANAGEMENT SYSTEM
D 0-22

JAWBREAKER
N 0-22
N 3.....13:1D

KAVES OF KARKHAN
D 0-22

LAFF PAK
D 1-22
D 0.....05-0A:00 0B:06 16:FF

LOGO
N 0
N 2-22
C 1.....16:FF

LOWER REACHES OF ASPHAI
D 0-22

MARAUDER
D 1-22
DC 0.....05-0A:00 0B:06 16:FF

MASTER TYPE
D 0-22.....00:FE 03:00 05-0A:00 0B:0C

MOUSKATTACK
N 1-22
D 0.....05-0A:00 0B:06 16:FF

MISSION ESCAPE
D 0
N 1-1D

MISSILE DEFENSE
N 0-22

NEUTRONS
D 0-22

OO-TOPOS
D 0-22

OO-TOPOS (NEW)
DS 0-22.....00:FE 03:00 05-0A:00 0B:0C

ORBITRON
DC 0
DS 1.5-E.5.....00:FE 01:DD 02:AD 03:DA
04-0A:00
DS B.5.....01:B5 02:D5 03:AA 14:FF

OUTPOST
DC 0
DS 1.5-9.5.....00:FE 01:DD 02:AD 03:DA
04-0A:00 0B:0C
N B.5.....01:D5 02:AA 03:AD

PEGASUS II
N 0-22
N 3.....13:1D

PFS & PFS REPORTS
D 1-13
D 0.....01:93 02:F3 03:FC 04-0A:00
11:00 15:E0 17:04 18:93
19:F3 1A:FC 1B:FF
Note: Write protect copy diskette before using.

PFS & PFS REPORTS (new release)
DSC 0-1
DSC 3-22
DSC 2.....11:02 14:F8 15:60 17:04
19:F3 1A:FC 1B:FF
Note: Set max retries to 2. Write protect the copy disk before using.

PHANTOMS 5
N 0
N 2-1C

PRISM PRINT
S 0-21.....0C:08 0F:01 10:01
SD 22.....00:FE 01:AA 02:D5 03:9F
04-0A:00 0B:0C 11:02 15:10
17:03 18:AA 19:D5 1A:9F
25:02 26:03 28:03

THE PRISONER
N 0-22

PULSAR II
N 0.....00:FE
N 2-C
N 11-19
N 1A.5-1D.5

RASTER BLASTER
DS 0
DS 1.5.....00:FE 01:AD 02:DE 03-0A:00
0B:0C
DS 3.5-F.5 BY 4
DS 5-11 BY 4
DS 6-12 BY 4

REAR GUARD
N 0-20
N 22
C 21.....16:FF

RETRO-BALL
D 0.....00:FE
D 4-6
D 9-C
D E-10
D 12-14
D 17-1D
D 20-22

REVERSAL
N 0-2
N 3.5
N 5-22

RINGS OF SATURN
D 0-22

SABOTAGE
N 0-22

SAT ENGLISH (SIDE 1)
Note: This two sided disk requires version 2.2.
SD 0
D 1-3.....01:EE 02:FC 03:97 04-0A:00
D 6-22
SD 4-5.....00:FE 01:FF 02:FF 03:DB
Note: The copy process may 'hang up' during the analyze mode of tracks 4 & 5. This is normal. If it happens, press RESET and try the track again.

SAT ENGLISH (SIDE 2)
Note: this requires version 2.3, See all notes for side 1. Side 2 is very similar.
SD 0
D 1-3.....01:97 02:EE 03:D5 04-0A:00
D 6-22
SD 4-5.....00:FE 01:FF 02:FF 03:DB
26:06 28:06

SCREENWRITER
N 1-22
CD 0.....05-0A:00 0B:06 16:FF

SNACK ATTACK
N 0-11.....11:00 15:40 17:02 18:AF
19:DE

SNAKE BYTE
P 0
S 1.5-F.5

SNEAKERS
N 0
S 1.5-F.5

SNEAKERS (NEW)
DS 0
DS D.5
DS 1.5-C.5.....00:FE 01:DD 02:AD 03:DA
04-0A:00 0B:0C
DS 2.5.....00:FF

SNOGGLE (JOYSTICK)
N 0-2
N 3-D.....00:FE

SOFT PORN
N 0-22
N 3.....13:1D

SOUTHERN COMMAND
D 0.....07:00
D 1-22.....01:D4 03:B7

SPACE EGGS
N 0
N 2-6
N 11-13
N 14-1A

SPACE RAIDERS
N 0-4

SPACE WARRIOR
D 0
D 1-5 BY 4.....00:FE 01:DF 02:AD 03:DE
04-0A:00 0B:0C
D 2.5-6.5 BY 4
D 3.5
D 8-11

SPECIAL DELIVERY SOFTWARE
D 0-22

STAR BLASTER
D 0
D 7-20 BY 1.5.....00:FE 01:DF 02:AD 03:DE
04-0A:00 0B:0C

STAR THIEF
Note: Requires Vers. 2.2
NC 0-13
NC 22.....16:FF 21:05 22:02
Note: You will have to open the copy drive to adjust the D-speed. This is required to preserve the nibble count.

STEP BY STEP (new)
D 0-22

SUICIDE
D 0
D 11.5-20.5 BY 1.5..00:FE 01:DF 02:AD 03:DE
04-0A:00 0B:0C

SUPER SCRIBE
N 0-22
N 3.....13:1D

SUPER STELLER TREK
D 0
D 1-22.....00:FE 01:EE 02:EF 03:FE
04-0A:00 0B:0C

SWASHBUCKLER
D 0-22.....00:DB 11:00 15:C0 17:08
19:AA 1E:DB 1F:DB

TEMPLE OF ASPHAI
D 0-22

TETRAD
N 0-22

THIEF
D 0
D 1-3.....00:FE 01:AE 02:DE 03:FE
04-0A:00 0B:0C
D 6-22
DS 4.....01:DB 02:AD 03:FE 0F:00
DSP 5.....0F:01
Note: Drive speed must be exactly set according to APPLE's D-speed test. Vers. 2.2 and above may copy tracks 4 & 5 by preserving the nibble count. To do this, change parms: 16:FF, 0F:01, 10:01, then copy tracks 4 and 5 synchronized.

THRESHOLD
N 0-22
N 1.....13:1D

TIME ZONE (disk 1)
DS 0-22

TIME ZONE (remaining diskettes)
N 0-22

TWERPS
P 0
N 1.5-E.5
N 1A

ULTIMA (player master side)
N 0-22

ULTIMA (program side)
N 0-22.....00:FE 25:0B
Note: Set retries to 3 and recopy tracks with read or write errors as many times as required to copy properly.

ULYSSES AND THE GOLDEN FLEECE
N 0-22
N 3.....13:1D

VISICALC ///
S 0-22
Note: On Ver. 2.1 change parm 11:02.

VISICALC 16 (Early versions)
N 0-22

VISICALC 16
D 0
D 2-16

VISIDEX
D 0-22.....11:00 15:60 17:03 18:AA
19:EB 1A:FD

VISIFILE
D 0-22.....11:00 15:40 17:03 18:AA
19:EB 1A:EC

VISISCHEDULE
D 0-22.....11:00 15:60 17:03 18:AA
19:EB 1A:EC

VISITREND/VISIPLOT
D 0-6
D 8-22
D 7.....11:00 15:40 17:03 18:AA
19:EB 1A:FC
Note: separate versions copy alike.

VISITERM
D 0-5
D 7-22
D 6.....11:00 15:20 17:03 18:AA
19:EB 1A:FC

WIZARD AND THE PRINCESS
N 0-22

WIZARDRY
S 0
N 1-9
N F-22
C A.....16:FF
S A-E.....0B:03 0C:08 0F:01 10:01
16:00
Note: Place a write protect tab over the notch in the copy of the boot side.

ZORK I & II (new versions)
D 0-22.....05:00

ZOOM GRAPHICS
DS 0-22

How to make back ups using:

Copy][plus Parameters

Copy][plus is a product of Central Point Software

The following is a list of parameters to change in order to back up certain pieces of software with Copy II Plus version 4.1. To the right of the program name is the abbreviated name of the publisher. For a complete list of the publishers, refer to page 23.

When making a backup, be sure to follow the steps in order. Often a parameter will not be re-listed if it is set for a prior range of tracks.

To back up a program, first find its name in the list of parameters. Directly below the name is a list of the tracks to copy and parameters to change. If the word BY is used, set the increment to the value that follows it. Use the default increment of one if no other figure is given.

When the word SECTMOD appears, it means that a sector should be changed using the Track Sector-Editor. Be sure to patch the read/write routines if the listing shows PATCHED and to use the correct DOS (3.2 or 3.3). Place the destination disk in drive one, then perform the changes listed. The command format is:

SECTMOD [F=n, C=n, T=n, S=n]
DOS 3.n PATCHED
CHANGE ADDRESS A1 FROM A2 TO A3

The meaning of F, C, S, T and A1, A2, A3 are explained below:

F- Disk format to be used. The value (n) will be either 13 or 16.
C- Toggle. The value (n) will be either on or off.
T- Track to be modified.
S- Sector to be modified.
A1- Location to be changed in the sector buffer.
A2- Old value.
A3- New value.

The middle line from the example gives the DOS (3.2 or 3.3) patched. Some diskettes can be duplicated using the default parameters (select the Bit Copy option from the main menu). If the diskette you wish to back up is not listed, try the default settings anyway.

An asterisk (*) next to the product name indicates that those parameters were submitted by users and have not been verified.

3-D GRAPHICS SYSTEM (CP)
0-8
11-12
15-17

Alternative Method
0-2
4-8
11-18

A2-PB1 (PINBALL) (SL)
0.....10:96
1-15.....A:3 E:DB F:AB 10:BF 44:1 45:D 46:F

ABM (MU)
0-22

ACCOUNTANT (DSS)
0-22.....3C:1 4B:1

ACE CALC (ART)
0-22

Alternate Method
0-22.....10:96

ACE WRITER (ART)
0-22.....10:96

Alternate Method
0-22

ADAPTABLE SKELETON (UNK)
0-22.....20:96 9:0 19:AA 1F:AA

ADDRESS BOOK (MU)
0-22.....D:1 10:96 24:96

ADVANCED VISICALC FOR THE APPLE IIe (VCP)
0-22.....10:96 24:96 D:1

ADVANCED VISICALC FOR THE APPLE /// (VCP)
Same as Visicalc ///

AE (BS)
0.....A:3 E:DD F:AA 10:AD
1.5-C.5
E-1E.5 Step 1.5.....A:3 E:D5 F:AA 10:96 51:1 52:03 53:18 54:0

AGENDA FILES (AC)
0-22.....10:96

AIR SIMULATOR (MS)
0-F

AIR TRAFFIC CONTROLLER (AG)
0-22.....10:96
23.....31:0 50:1 10:96

AKALABETH (CP)
0.....9:0 31:0
2-3.....E:DE F:AA 10:AD
6-18

ALGEBRA ONE & TWO (EW)
0-22

ALGEBRA SERIES (EW)
0-22.....10:96 9:0 24:96 D:1 31:0

ALIEN RAIN & TYPHOON (BS)
0-5.....9:0 31:0 D:D5 F:0
6-E.....E:DE

ALKEM STONES (L10)
0-22.....A:3 10:96

ALPHA BYTES (LTS)
0-22.....9:0

AMPERMAGIC (AD)
0-22

APPLE ACCESS /// (UNK)
0-22

APPLE ADVENTURE (AC)
0-22.....D:1 10:96 24:96

APPLE BARREL (CDS)
0-22

APPLE BUSINESS BASIC (AC)
0-22

APPLE BUSINESS GRAPHICS (AC)
0-22.....D:1 10:96 24:96

APPLE CILLIN II (XPS)
0-C

APPLE II BUSINESS GRAPHICS (AC)
0-22.....D:1 10:96 24:96

APPLE /// BUSINESS GRAPHICS (AC)
0-22
(ERROR 2 OKAY)

APPLE FORTRAN (AC)
0-22

APPLE LOGO (AC)
0-22
1.....A:1 4B:1 50:1 E:AA 1C:AA 3B:1 4D:8

Alternative Method
0-22
1.....A:1 4B:1 50:1 E:FC 19:FD 1C:AA 1F:EE

Alternative Method
0-22
1.....A:1 4B:1 50:1 E:AA 1C:AA

Alternative Method
0-22
1.....A:1 4B:1 50:1 3B:1 4D:8

Alternate method
2-22
0.....D:1 24:96 10:96
1.....A:1 50:1 4B:1 E:AA F:D6 10:EE
(ERROR 6 OK)
NOTE: We have been told that Apple Logo requires a lot of persistence! Keep retrying track 1 until the disk works. The disk drive speeds need to be within .1 of 200 milliseconds per revolution

APPLE PANIC (BS)
0-D

Alternate Method
0-5.....9:0 F:0
6-D.....E:DE

APPLE PASCAL 1.1 (AC)
Use COPY DISK from MAIN MENU

APPLE PILOT (AC)
0-22

Alternate Method
0-22.....10:96 24:96 D:1

APPLESOFT COMPILER (MIS)
0-22

APPLE WORLD (USA)
0-23

APPLE VISISCHEDULE /// (VCP)
Copy disk from main menu.

APPLEWRITER II AND //e (AC)
0-22.....10:96

Alternate Method
0-22.....D:1 10:96 24:96 3F:1

APPLEWRITER II PRE-BOOT (VX)
0-22.....10:96 9:0

Alternate method
0-23.....10:96 9:0 3F:1

APPLEWRITER /// (AC)
0-22.....D:1 10:96 24:96

APPOINTMENTR HANDLER (UNK)
0-22

APVENTURE TO ATLANTIS (SY)
0-22.....10:96 24:96 9:0 31:0 D:1

A2-PB1 (PINBALL) (SL)
0.....10:96
1-15.....A:3 E:DB F:AB 10:BF 44:1 45:D 46:F

ARSENE LARCIN (LOD)
0-23.....10:96

AUTOBAHN (SRS)
0
4-6.....D:1
9.5-C.5

AUTOMATED ACCOUNTING FOR MICROCOMPUTERS (UNK)
0-22.....10:96

AZTEC (DM)
0-22.....D:1 10:96 24:96

BACK-IT-UP II (SEN)
0.....10:96 9:0
1.5-B.5.....10:B5 A:3

BACK-IT-UP II+ 2.3 (SEN)
0-D.....10:96 9:0
(ERROR on track 1 okay)
Note: Sensitive to drive speed.

BAG OF TRICKS (QS)
0-15.....0E:D6 3E:2 34:1 35:DF

Alternate Method
0
1-15.....E:D6
SECTMOD [T:0 S:8] DOS 3.2 PATCHED
Change address A0 from 20 to 60.

BANDITS (SRS)
0
1.5-1A.5
1C.5-1F.5.....D:1

BASIC FRANCAIS (LOD)
0-23.....10:96

BASIC TUTOR SERIES (EC)
0-22.....9:0 10:96

BATTLE CRUISER (MGI)
0-22
4.....44:0

BATTLE FOR NORMANDY (SSI)
0-22.....E:D4 10:B7 34:1 37:6E 38:FE

Alternate Method
0-22.....E:D4 10:B7 34:1 38:FE

BATTLE OF SHILOH (SSI)
0-22.....E:D4 10:B7

BATTLESIGHT (VER)
Use COPY DISK from MAIN MENU.

BEER RUN (SRS)
0.....9:0
1.5-D.5.....D:1 3B:40

BEST OF MUSE (MU)
0-22

BILL BUDGE'S 3-D GRAPHICS (CP)
0-8
11-12
15-17

Alternate Method
0-22

BILL BUDGE'S SPACE ALBUM (CP)
0-B

BILL BUDGE'S TRILOGY OF GAMES (CP)
0-A

BIRTH OF THE PHOENIX (PHO)
0-9

BOLO (SY)
0-22.....D:1 9:0 24:96 10:96

BOMB ALLEY (SSI)
0-22.....E:D4 10:B7 34:1 37:6E 38:FE

BORG (SRS)
0.....10:96 9:0
1.5-B.5.....D:1 24:96 A:3 E:DD F:AD 10:DA 3B:40
D-20

Alternate method
0.....E:DD F:AD 10:DA D:1 22:00 23:00 24:00

BRAIN SURGEON (UNK)
0-22
Error 1 on Trk 11 OK.

BRIDGEMASTER (DY)
0-22

C-DEX TRANING PROGRAM (CX)
0-22.....E:D6 F:AB 10:96 1A:AB 1D:AB 20:AB

CANNONBALL BLITZ (SOL)
0-22.....10:96
SECTMOD [T:17 S:0E] DOS 3.3
Change address CD from 49 to 60

Alternate method
0-22
3-F.....3B:1 A:1 4B:1 4D:8 50:1 (Error 6 OK)

Alternate method
0-22.....10:96
SECTMOD [T:17 S:0E] DOS 3.3
Change address C8 from 49 to 60

CARAIBES (LOD)
0-23.....10:96

CARTELS AND CUTTHROATS (SSI)
0-22.....E:D4 10:B7

CASINO (DM)
0-22.....10:96

CASTLE OF DARKNESS (LOG)
0.....D:1 24:96 10:96 9:0
1-22.....E:AB F:AB

CASTLE WOLFENSTEIN (MU)
0-22.....D:1 31:0

CAVERNS OF FREITAG (MU)
0-2.....9:0 10:96
3-22.....F:DA

CAVES OF OLYMPUS (SAM)
0-22.....10:96 9:0

CEILING ZERO (TKS)
0-2
3-11.....9:0 E:D6 1C:D6 34:1 38:F9 4F:1

CELLS (UNK)
0-22

CHESS 7.0 (OD)
0-22.....10:96 9:0

Alternate Method
0-22.....10:96 9:0 8:1 3E:2

LE CHOMEUR (LOD)
0-23.....10:96

CHOPLIFTER (BS)
0.....A:3 44:1 45:D 9:0 0:F 50:3
1-8.....A:FD 31:0 43:0 45:10 4F:1 46:12
9.....45:8 46:D
A-B.....45:2
C-1E.5 STEP .5.....A5:8 10:D4 51:1 D:1
20.....45:6 D:0 4F:0

NOTE: Choplifter Serpentine David's Midnight Magic and Starblazer use track arcing are very sensitive to drive speed. If you have problems try reversing drives.
Another Choplifter hint: Just use a single drive for copying. Error 5 on Track 1C.5 is OK.

COLOSSAL CAVE ADVENTURE (FC)
0-22

COMPUTER AIR COMBAT (UNK)
0-22.....E:DB F:D5 10:DE 8:1

COMPUTER AMBUSH (SSI)
0-22.....E:D4 10:B7 34:1 37:6E 38:FE

COMPUTER BASEBALL (SSI)
0-22.....E:D4 10:B7 34:1

COMPUTER MATH GAMES (AD)
0.....10:AD A:3
1-2.....10:DB
3-22.....10:96

COMP. MODELS FOR MANAGEMENT (AW)
0-22

COMPUTER NAPOLEONICS (SSI)
0-22.....E:DB F:D5 10:DE 8:1

COMPUTER QUARTERBACK (SSI)
0-22.....34:1M 37:6E 3E:2 9:0 0E:D4 10:B7

COMPUTER STOCKS & BONDS (UNK)
0-22

CONGO (SS)
0-22.....D:1 9:0 24:96 10:96

COPTS AND ROBBERS (SRS)
0.....10:96 9:0
1.5-F.5.....D:1 24:96 A:3 E:DD F:AD 10:DA 3B:40

COPY][PLUS (CE)
See manual pages 2-4 2-6

COUNTING BEE (EW)
0-22

COVETED MIRROR-SIDE 1-(PEN)
0-22 STEP 2.....9:0 10:96 E:D5
1-17 STEP 2.....E:D4
19.....9:1
1B-21 STEP 2.....9:0

COVETED MIRROR-SIDE 2-(PEN)
0-22 STEP 2.....9:0 10:96 E:D5
1-21 STEP 2.....E:D4

Note:Boot The COVETED MIRROR on side 2.

CRANSTON MANOR (SOL)
0-22
18.....3B:1 A:1 4B:1 4D:8 50:1 (ERROR 6 OK)

CRIME WAVE (PEN)
0-10 step 2.....E:D5 F:AA 10:96 9:00 6:04 31:00
1-11 step 2.....E:D4 F:AA 10:96 9:00 6:04 31:00

CRISIS MOUNTAIN (SY)
0-22.....10:96 24:96 9:0 31:0 D:1

Alternative Method
0.....10:96
3-22.....9:0 3A:0 50:20

CROSSFIRE (SOL)
0-B.....9:0
1.....3B:1 A:1 4B:1 4D:8 50:1 (ERROR 6 OK)

CRUSH CRUMBLE AND CHOMP (AUT)
0-22.....10:96 9:0

Alternative Method
0-22.....10:96

APPLE SPOTLIGHT INSTANT ZOO (UNK)
0-22

CUBIT (MM)
0-22.....10:96 9:0 31:0 24:96

DARK CRYSTAL (SOL)
Use Copy Disk from main menu for all four disks.
SECTMOD DISK 1A: [T:5 S:F]
Change addresses A8-AA ALL TO EA
SECTMOD DISK 1A: [T:7 S:C]
Change addresses 22-24 ALL TO EA

DATESTONES OF RYN (EP)
0-22.....A:3 10:96

DAVID'S MIDNIGHT MAGIC (BS)
0.....A:3 44:1 454:D 9:0 0:F 50:3
1-A.....44:0
B.....44:1 31:0 43:0 45:8
C-19 STEP.5.....10:F5 F:FD 51:1 4F:1 D:1
See note at Seawolf.

DAWN PATROL (TSR)
0-22.....9:0 10:96

DB MASTER (OLD) (SW)
0-5.....10:96 24:96 D:1
6.5-22.5.....D:0

DB MSTR UTILITY PAKS 1 & 2 (SW)
0-5.....10:96 24:96 D:1
6.5-22.5.....D:0

DEADLINE (IC)
0-22.....10:96 1E:BC

DEMON'S FORGE (ART)
0-22

DESKTOP PLAN II (VCP)
0-22.....10:96 34:1 36:2A

Alternative method
Same as Visifile

DICTIONARY 2.1 (SOL)
COPY DISK from MAIN MENU
SECTMOD [T:8 S:F]
Change addresses:
13 to 4C
14 to 24
15 to 6E

DISK ORGANIZER (SEN)
0
1.....3B:1 A:1 4B:1 4D:8 50:1 (Error 6 OK)
2-4.....D:1
A-B

DISK RECOVERY (SEN)
0-22.....10:96 9:0 A:3
Error 2 on Track 1 OK. May take several tries

DLM SOFTWARE (DLM)
0-22

DRAGON FIRE (L10)
0-22

DUNG BEETLES (UNK)
0
1.....A:3 E:F5 F:F6 10:F7
4-22
SECTMOD [T:0 S:1] DOS 3.2
Change addresses:
6D from 01 to 7B
6E from 61 to 69

DUNGEON (TSR)
0-22.....10:96 9:0

EARLY GAMES (CPS)
Use Copy Disk from main menu.

EDUC. ACTIVITIES SOFTWARE (UNK)
0-22

EINSTEIN COMPILER (EIN)
Use COPY DISK from MAIN MENU.
SECTMOD [T:8 S:4]
Change addresses:
2A from BD to 4C
2B from 8C to E2
2C from C0 to 91

ELECTRIC DUET (IN)
Use Copy Disk from main menu.

ELIMINATOR (ADA)
0-21
SECTMOD [T:3 S:0D] DOS 3.3 PATCHED
Change addresses:
2E from 20 to EA
2F from 30 to EA
30 from 72 to EA

ESCAPE (UNK)
0-22

ESCAPE FROM ALCATRAZ (SY)
0-22.....10-96 9:0 31:0 8:1

ESCAPE FROM RUNGISTAN (SRS)
0-2.....10:96
3-22.....10:F7

Alternate method
0-21

EXECUTIVE BRIEFING SYSTEM (LTS)
0-22.....9:0
SECTMOD [T:21 S:0] DOS 3.3
Change address 27 from FB to 22

EXECUTIVE SECRETARY (SOF)
0-22.....9:0 8:1 10:96

Alternate method
0-22.....8:0 10:96 31:0 9:0

EXPEDITOR (SOL)
0-22.....10:96
3 & 1F.....3B:1 A:14B:1 4D:8 50:1
(Error 6 OK)

E-Z DRAW 3.3 (SRS)
0-22.....9:0 E:D7 10:96 8:1 A:2 4:F3 3A:3 D:1 24:96 31:0

FACEMAKER (SPN)
FIRST:Use COPYA from DOS 3.3 system master
THEN:Use COPY II PLUS
0.....34:01 36:2A 37:1B 38:FC 3E:02

FASTGAMMON (QS)
0-22

FIREBIRD (GB)
0-0D.....10:96 9:0
1.5-B.5.....D:1 24:96 A:3 E:DD F:AD 10:DA 3B:40

FIRST CLASS MAIL (CTS)
0-22

FLIGHT SIMULATOR (SL)
0.....10:96
1.5-21 STEP 1.5.....E:DB F:AB 10:BF A:3 4E:1
7-8
9.5

FORMAT II (KN)
COPY DISK from MAIN MENU

FORMAT II (Version 7) (KN)
0-22.....10:96
SECTMOD [T:B S:5] DOS 3.3
Change addresses:
04 from A9 to 4C
05 from 03 to 31
06 from BD to 68

FRAZZLE (MU)
0-22

GALACTIC ATTACK (SIR)
0-22.....10:96 24:96 D:1

GALACTIC GLADIATORS (SSI)
0-20.....10:B7 E:D7 9:0 31:0
21-22.....34:1

GAME SHOW (CAI)
0-22.....9:0

GENERAL MANAGER (SOL)
Use COPY DISK from MAIN MENU for working program and sample files.

Master program:
0-22.....9:0

Alternate Method
0-22.....10:96
SECTMOD [T:1F S:0E] DOS 3.3
Change addresses:
C1 to 4B
C2 to E0
C3 to 49
SECTMOD [T:21 S:01] DOS 3.3
Change address 2E to 60

Alternate method for master program
COPY DISK from MAIN MENU
SECTMOD [T:0D S:0E] DOS 3.3
Change addresses:
2C from 60 to EA
SECTMOD [T:21 S:0B] DOS 3.3
Change addresses:
D7 from E3 to CB
SECTMOD [T:21 S:0E] DOS 3.3
Change addresses:
01 from 08 to 60

Method for version 2.0N
COPY DISK from MAIN MENU
SECTMOD [T:20 S:0B] DOS 3.3
Change addresses:
09 from 20 to EA
0F from 20 to EA
10 from 00 to EA
11 from 70 to EA

Method for version 2.0Y
COPY DISK from MAIN MENU
SECTMOD [T:20 S:0B] DOS 3.3 PATCHED
Change addresses:
27 from 00 to EA
28 from 70 to EA
29 from 20 to EA
29 from 20 to EA
2A from 0F to EA (Optional)
26 from 20 to EA (Optional)

GEOMETRY & MEASURE Vol 1 & 2 (UNK)
0-22.....D:1 10:96 24:96

GERTRUDE'S PUZZLES (LC)
0-22.....10:96 9:0

GLOBAL WAR (MU)
0-22

GOBBLER (SOL)
0-22.....9:0
3.....3B:1 A:1 4B:1 4D:8 50:1 (Error 6 OK)

GOLD RUSH (SNT)
0-22.....D:1 9:0 24:96 10:96

GORGON (SRS)
0.....10:96 9:0
1.5-E.5.....D:1 24:96 A:3 E:DD F:AD 10:DA 3B:40

GRAPHICS PROCESSING (SW)
Main Disk:
0-22.....19:DD 1A:AA
Utilities disk is not protected.

GRAPHTRIX (DAT)
0-22

GUADALCANAL CAMPAIGN (SSI)
0-22.....E:D4 10:B7 34:1 37:6E 38:FE

HADRON (SRS)
0.....10:96 9:0
1.5-E.5.....D:1 24:96 A:3 E:DD F:AD 10:DA 3B:40

HELLFIRE WARRIOR (AUT)
0-22

HEART LAB (UNK)
0-22

HI-RES COMPUTER GOLF (AG)
0-22
(both sides)

Alternate Method
0-22.....19:DF D:1 34:1

HI-RES COMPUTER GOLF VERSION 2 (AG)
Copy both sides.
0-22.....10:96

HI-RES FOOTBALL (SOL)
0-22

HI-RES SECRETS (AG)
0-22.....10:96 4:FB 19:DF 1F:DF A:1

Alternate method
0-22

HOME ACCOUNTANT (CTS)
0-22.....9:0 10:96

Alternate method
0-22.....9:0

HOME ACCOUNTANT 2.0 (CTS)
0-22

HOME ACCOUNTANT 2.01 (CTS)
Use COPY DISK from MAIN MENU

HOME MONEY MINDER (CTS)
0-22.....10:96 9:0

HYPERSPACE WARS (CTS)
0-22.....9:0

INCREDIBLE JACK (BUS)
0-22
Write protect copy before using.

Alternate method
0-22.....10:96 24:96
23.....D:1 9:0

INSTANT ZOO (UNK)
0-22.....D:1 10:96 24:96

INTERACTIVE FICTION (ADA)
0-22

INVASION ORION (AUT)
0-22

INVENTORY OF EQUIP. (UNK)
0-22

INVOICE FACTORY (ML)
0-22

JAW BREAKER (SOL)
0-22.....9:0
3.....3B:1 A:1 4B:1 4D:8 50:1
(Error 6 OK)

JIGSAW (ML)
0
1-17.....D:1 24:96 E:D3 F:96 10:F2 9:0 31:0

Alternate method
0.....10:96 9:0 31:0
A-17.....10:96 9:0 31:0
1-9.....E:D3 F:96 10:F2 9:0 31:0

KABUL SPY (SRS)
Side One:
0
1-21.....10:F7
22.....A:5 E:AA F:D5 10:D5 11:BD 12:BD
SECTMOD [T:0 S:0] DOS 3.3 PATCHED
Change addresses:
49 from 20 to EA
4A from 03 to EA
4B from 20 to EA

Side Two:
0-21.....10:F7

KEY PERFECT (MSP)
0-22

KNIGHTS OF DIAMONDS (SIR)
(both sides)
0-22.....10:96 24:96 D:1
Write protect disk before using.

Alternate method
Boot side:
0-22.....D:1 10:96 24:96 34:01 37:00
Be sure to write-protect side one.
Scenario side:
A-22
0-9.....D:1 10:96 24:96 4B:1
(Error 6 OK)

KNIGHTS OF THE DESERT (SSI)
0-22.....E:D4 10:B7

KNOW YOUR APPLE (MU)
0-22

KRELL LOGO (new) (KL)
0-22.....10:96
SECTMOD [T:2 S:3]
Change addresses:
5B from D0 to EA
5C from 03 to EA

Alternate method
0-22.....10:96 9:0
SECTMOD [T:2 S:3]
Change addresses:
5B from D0 to EA
5C from 03 to EA

LETTER PERFECT (LJK)
0-22.....10:96 9:0

LIST HANDLER AND UTILITY (SVS)
(older version)
1-11
0.....9:0 A:3 44:1 45:D 50:3
12-22.5 step .5.....D:1 E:F5 F:D7 10:F7 45:8 46:D 51:1
See note for Seafox.

LIST HANDLER Version 1.1 (SVS)
PROGRAM DISK
1-11
0.....9:0 A:3 44:1 45:D 50:3
12-19.5 step .5.....E:D7 F:D7 10:DD 45:8 46:D 51:1
20-22.5 step .5.....E:0 F:FD 10:D4
UTILITY DISK
1-11
0.....9:0 A:3 44:1 45:D 50:3
12-22.5.....D:1 E:D2 F:D7 10:DF 45:8 46:D 51:1

LJK EDIT 6502 (LJK)
0-22.....10:96 9:0

MAGIC WINDOW I AND II (ART)
0-22

Alternate method
0-22.....10:96 24:96 D:1

MAGIC WINDOW II (ART)
0-23.....(Error 2 on track 23 OK. Try 3C:4 if problems.)

MAGICALC (ART)
0-22.....9:0

MAGIC MAILER (UNK)
0-22

MAGIC WINDOW I & II (ART)
0-22

MAILING LIST (UNK)
0-22

MARAUDER (SOL)
0-22.....10:96 9:0
Sectmod [T:3 S:7] DOS 3.3
Change address 90 from A8 to 60

MARS CARS (DM)
0-22.....10:96

MASK OF THE SUN (ULS)
Sides A and B
0-22.....10:96
Side A
SECTMOD [T:2 S:0D] DOS 3.3 PATCHED
Change address 42 from 8F to EA
Change address 43 from C0 to EA
Do not write protect backup

MASTER TYPE (new) (LNS)
0-22.....9:0 37:FF 34:1 38:BF 35:EB 39:EB 36:AB 10:96

MASTER TYPE (old) (LNS)
0-2
3-22.....E:D4 (ERROR on track 1B okay)
SECTMOD [T:0 S:3] DOS 3.2 PATCHED
Change address 63 from 38 to 18
SECTMOD [T:2 S:A] DOS 3.2 PATCHED
Change address 2E from 23 to 2E

MATH STRATEGY (AC)
0-22.....10:96 24:96 D:1

MECC (Vol. 1 & 2) (CW)
0-22
2.....10:96 9:0

MECC (Vol. 3 4 and & 7) (CW)
0-22.....9:0 8:1

MEGAWRITER (MH)
Use COPY from MAIN MENU.

METEOR MULTIPLICATION (DLM)
0-22

MICROBE (SY)
0-22.....10:96 9:0 31:0

Alternate method
0-22.....10:96 24:96 9:0 31:0 D:1

MICRO COOKBOOK (VC)
0-22

MICRO DEUTSCH (KL)
0-22.....E:D4
Error 2 on Track 1B is OK

MICRO SKILLS (EU)
0
1-22.....10:96 19:AA 1C:A 31:00
SECTMOD [T:0 S:3] DOS 3.3 PATCHED
Change address 42 from 38 to 18

MICROSOFT ADVENTURE (MIS)
0-22

MICRO WAVE (CC)
0-22
11.....3B:1 A:1 4B:1 4D:8 50:1

Alternate Method
0-22.....10:96
SECTMOD [T:2 S:1] DOS 3.3
Change addresses:
DA from A9 to AD
DB from 60 to 03
DC from 8D to 81
DD from 7E to 60

Alternate Method
0-22

MILLIKEN SERIES (ML)
0-22

MINER 2049'ER (MF)
0.....4B:1 10:96
1-22.....4B:0 E:D3 F:96 10:F2 A:3 9:0 31:0 8:1 D:1 24:96 6:6

Alternate method
0.....4B:1 10:96
1-3.....E:D3 F:96 10:F2 A:3 9:0 31:0 8:1 D:1 24:96 6:6 1C:96 1D:D3 1E:E5 19:D3
4-22.....4B:0

MINIT MAN (PEN)
0-22 STEP 2.....10:96 9:0
1-21 STEP 2.....E:D4

MISSILE DEFENSE (SOL)
0-22.....D:1

MISSING RING (DM)
0-22.....D:1 24:96 10:96 34:1
Do not write protect!

MISSION: ASTEROID (SOL)
0-22

Alternate method
0-22.....10:96 24:96 D:1

MISSION: ESCAPE (MSP)
0-1D

MIX AND MATCH (AC)
Use COPY DISK from MAIN MENU.

Alternate Method
0-22.....9:0 10:96

MONEY STREET AND UTILITIES (BES)
0-22.....Errors OK

MOUSKATTACK (SOL)
0-22.....10:96
SECTMOD [T:18 S:03]
Change address B1 from 49 to 60

MULTI-DISK CATALOG (SEN)
0-8
3.....A:1 E:AF 3B:1 4B:1 4D:8 50:1

MULTIPLAN (MIS)
0-22.....10:96

MURDERS BY THE DOZEN (CBS)
Use COPY DISK from MAIN MENU.

MUSIC MAKER (SS)
0-22

MYSTERY HOUSE (SOL)
0-22

NUETRONS (L10)
0-22.....A:3 10:96

NIBBLES AWAY I (MWD)
0-22

NIBBLES AWAY II VERSION C2 (MWD)
0.....10:96
1-E.....E:D7 10:97
10-15

NIBBLES AWAY II VERSION C3 (MWD)
0-15.....E:D7 F:AA 10:97
(Error 2 on Track 0F OK)

NIGHTMARE ALLEY (SY)
0-22.....10:96 9:0 34:1 31:0

NORTH ATLANTIC '86 (SSI)
0-22.....E:D4 10:B7 34:1 37:60

OLYMPIC DECATHALON (MIS)
0-22.....9:0

Alternate method
0-22

OO-TOPOS (SNT)
0-22

OPERATION APOCALYPSE (SSI)
0-22.....E:DB F:D5 10:DE 8:1

ORBITRON (SRS)
0-1.....9:0 31:0
1.5-F.5
(Write protect copy!)

OUTPOST (SRS)
0.....10:96
1.5-B.5.....D:1 24:96 A:3 E:DD F:AD 10:DA 3B:40

PARTS OF A MICROSCOPE (UNK)
0-22

PEEPING TOM (ML)
0
1.....E:F5 F:AB 10:BE 9:0
4-22
SECTMOD [T:0 S:1] DOS 3.2
Change address 6E from 60 to 68

PEGASUS II (SOL)
0-22
3.....3B:1 A:1 4B:1 4D:8 50:1 (Error 6 OKAY)

PERSONAL FINANCE MANAGER (AC)
0-22.....10:96

PERSONAL SECRETARY (SFS)
0-22.....10:96 9:0

PFS FILE & PFS REPORT (SPC)
COPY DISK from main menu.
Write protect copy!

PFS GRAPH //e (SPC)
Same as PFS File & PFS Report

PHANTOMS FIVE (SRS)
0.....9:0
2-1C.....3A:0 50:20

PIK (APPLE /// BOOT PROGRAM) (AC)
Use COPY DISK from main menu

PINBALL (A2-PB1) (SL)
0.....10:96
1-15.....A:3 E:DB F:AB 10:BF 44:1 45:D 46:F 30:3 D:1

PINBALL CONSTRUCTION (BC)
Use COPY DISK from main menu

POOL 1.5 (IDSI)
0-15
1E-21
SECTMOD [T:0B S:7] DOS 3.2 PATCHED
Change address 6A from 8D to 60
SECTMOD [T:0 S:3] DOS 3.2 PATCHED
Change address 63 from 38 to 18

POWER TEXT (BP)
Use COPY DISK from MAIN MENU.

PRESDENT ELECT (SSI)
0-22.....E:D4 10:B7 34:1

PRISM (MAG)
0-22

PRISONER I & II (EW)
0-22.....10:96
SECTMOD [T:1F S:0E] DOS 3.3
Change addresses:
D5 from AD to 2F
D6 from 99 to AF
D7 from F0 to 32

PRO FOOTBALL (SDL)
0-22

LE PROPIO (LOD)
0-23.....10:96

PULSAR II (SRS)
0
1C.5-1D.5.....D:1
2-C.....E:DD
13-19
1A.5-1B.5

PSAT WORD ATTACK SKILLS (EW)
0-22

QUEST (PEN)
0-22.....E:0 F:AA 6:4 9:0 10:96 31:0

Alternate method
Side 1
0-22 Step 2.....10:96 19:DA 1F:DA 3C:1
1-21 Step 2.....E:D4
Side 2
0-22

QUICK FILE (AC)
0-22

RASTER BLASTER (BC)
0.....10:96
5-11 STEP 4.....D:1 9:0 31:0 A:2 E:AD F:DE 3B:40
6-12 STEP 4
7.5-F.5 STEP4
1.5-3.5 STEP2

READABILITY INDEX (UNK)
0-22

REPTON (SRS)
0-D.....E:FD F:DA 10:DD
SECTMOD [T:00 S:00] DOS 3.3
Change addresses:
8C from 4C to A9
8D from 80 to 4C
8E from BA to 8D
8F from 00 to 18
90 from 00 to BB
91 from 00 to A9
92 from 00 to 1B
93 from 00 to 8D
94 from 00 to 19
95 from 00 to BB
96 from 00 to A9
97 from 00 to BB
98 from 00 to 8D
99 from 00 to 1A
9A from 00 to BB
9B from 00 to 4C
9C from 00 to 80
9D from 00 to BA

RENDEZVOUS (EW)
0-23.....10:96 9:0

Alternative Method
0-22.....10:96 24:96 D:1 9:0 31:0

Alternative method
0-23.....10:96 9:0 24:96

RESCUE AT RIGEL (EP)
0-22.....A:3 10:96

RICOCHET (EP)
0-22.....10:96 9:0 8:1

ROACH HOTEL (ML)
0
1.....A:3 E:EE F:EA 10:FE
4-22
SECTMOD [T:0 S:1] DOS 3.2 PATCHED
Change addresses:
75 from 01 to 7B
76 from 61 to 69

ROBOT WARS (MU)
0-22.....D:1 31:0

ROCKY'S BOOTS (LC)
0-22.....10:96 9:0

SABOTAGE (SOL)
0-22
3.....3B:1 A:1 4B:1 4D:8 50:1
(Error 6 OK)

SARGON (HN)
0-1A.....10:F7

SCHEDULE OF EQUIP. (UNK)
0-22

SCREENWRITER II (SOL)
Use COPY DISK from MAIN MENU.
SECTMOD [T:3 S:B] DOS 3.3
Change adressess:
94 from 20 to EA
95 from 00 to EA
96 from 7F to EA
SECTMOD [T:13 S:4] DOS 3.3
Change addresses:
4D from 20 to EA
4E from 00 to EA
4F from 60 to EA

SEA DRAGON (UNK)
0-22

Alternative method
0-22.....10:96 24:96 D:1

SEAFOX (BS)
0.....A:3 44:1 45:D 9:0 0:F 50:3
1-8.....4:FD 31:0 43:0 45:10 4F:1 46:12
9.....45:8 46:D
A-B.....45:2
C-1E.5 step .5.....45:8 10:D4 51:1 D:1
20.....45:6 D:0 4F:0
NOTE: Seafox, Spider Raid, Choplifter, Serpentine, David's Midnight Magic and Star Blazer use track arcing and are very sensitive to drive speed. If you have problems try reversing drives.

SENSIBLE SPELLER (old) (SEN)
0-10.....10:96 9:0

Alternative method
0-22.....10:96 9:0 3B:40
1.....4B:1 A:2 50:2 E:D4 F:D4 3B:1
Note:Carefully adjust the duplicate drive speed when copying tracks 1 to match nibble count on the original disk and ignore errors.

Alternative method
0-22.....10:96 9:0
Note:Errors 2 on track 1 may be OK. Very sensitive to drive speed. Retry track 1 several times if necessary.

SERPENTINE (BS)
Same as Seafox

Alternative method
Same as Seafox but copy tracks 20-22 on last set of parameters.

SHERWOOD FOREST (PH)
0-22

Alternative method
Use COPY DISK from MAIN MENU until copy process hangs. Then bit copy tracks 1F-22.

SNACK ATTACK (old version) (DM)
0-12
SECTMOD [T:0 S:3] DOS 3.2 PATCHED
Change address 63 from 38 to 18

SNACK ATTACK (DM)
0-12
SECTMOD [T:1 S:3] DOS 3.2 PATCHED
Change address 39 from 38 to 18

SNEAKERS (SRS)
0.....9:0 10:96 44:1 45:10 D:1
1.5-C.5.....44:0
D.5

Alternative method
0.....9:0 10:96 44:1 45:10
1.5-C.5.....44:0 (Error 1 on Track A OK)
D.5.....44:1

SNOGGLE (BS)
0-9.....9:0 8:1

SOFTPORN ADVENTURE (SOL)
0-22.....9:0
3.....3B:1 A:1 4B:1 4D:8 50:1 (Error 6 OK)

SPACE EGGS (SRS)
0.....9:0
2-6
11-1A

SPACE INVADERS (UNK)
0-22.....10:96

SPACE VIKINGS (SL)
0-22

Alternate Method
0-22.....10:96 21:DA 8:1 A:3

SPECTRE (DM)
0-2.....10:96 9:0 8:1
3-22.....31:0 E:C5 10:B5

SPEED READER (AC)
0-22.....9:0 10:96

SPELLING STRATEGY (AC)
0-22.....10:96 24:96 D:1

SPIDER RAID (IN)
0
1-17.....A:3 E:92 F:93 4F:1 10:95 44:1 46:A 9:0 8:1 D:1 24:96 3F:1 34:1 36:2A 37:97 31:0 43:0
1.5-17.5.....E:95 10:92
Works only for new versions.
See note for Seafox.

SPITFIRE SIMULATOR (MS)
0-F
15

SPY'S DEMISE (PEN)
1-11 step 2.....9:0 10:96 E:D4
0-12 step 2.....6:4 31:0 (ERROR 2 on track 12 okay)

Alternate method
0-10 STEP 2.....9:0 10:96
1-11 STEP 2.....E:D4

Alternate method
0-12 STEP 2.....6:4 31:0 E:D5

STARBLASTER (PIC)
0.....10:96 9:0
7-20 STEP 1.5.....E:DF F:AD 10:DE

STARBLAZER (BS)
Same as Sea fox

STARCROSS (IC)
0-22.....10:96

STARSHIP COMMANDER (VOY)
0-22.....D:1 10:96 24:96

STATE OF THE ART ACCOUNTING (SAA)
0-22.....3C:4
Write protect copy!

STELLAR INVADERS (AC)
0-22

STEP BY STEP I & II (PDI)
0-22

STERLING SWIFT PRODUCTS (SSP)
0-22

STOCK PORTFOLIO SYSTEM (SMI)
3-22
0-2.....4:FD 8:1 10:AD

Alternate method
3-22
0-2.....4:FD 8:1 10:96

STOCK AND BONDS (AVH)
0-22

STORE MANAGER (HT)
0-22

STRIP POKER (ARW)
0-22

SUPER GRADEBOOK (HOB)
0-22.....10:96

SUPER GRAPHISME /// (LOD)
0-23.....10:96

SUPER PILOT (AC)
0.....10:96
2-22
SECTMOD [T:0 S:A] DOS 3.3 PATCHED
Change addresses:
79 from 43 to EA
7A from 41 to EA
7B from C6 to EA

SUPER PILOT (AC)
0-22.....10:96 24:96 D:1
(MAIN DISK ONLY)
Use COPY DISK for lesson and Super Co-Pilot

SUPER TAXMAN II (HAL)
0-22
Write protect copy!

Alternate method
0-22.....10:96 24:96 D:1

SUPER TEXT (MU)
0-22.....D:1 31:0

SUPER TEXT 40/80 (MU)
0-22.....9:0

SUPER SCRIBE II (UNK)
0-22.....10:96
3.....3B:1 A:1 4D:8 5D:1 (Error 6 OK)
1F.....3B:1 A:1 4D:8 5D:1 (Error 6 OK)

SUSPENDED (IC)
0-22.....10:96 1E:BC

Alternate method
Use COPY DISK from MAIN MENU.
Write protect before using.

SWASHBUCKLER (DM)
0-22
SECTMOD [T:0 S:3] DOS 3.3 PATCHED
Change address 42 from 38 to 18

Alternate method
0-22.....D:1 10:96 24:96

Alternate method
0-22

TAWALA'S LAST REDOUBT (BS)
0-22.....D:1

TAX MAN (HAL)
0-22

Alternate method
0-B

Alternate method
0.....10:96
1-B.....10:DA
11-12.....10:B5
13.....10:96

TAX MANAGER (ML)
Use COPY DISK from MAIN MENU

TAX PREPARER (HOW)
Use COPY DISK from MAIN MENU

Alternate method
0-22

TEMPLE OF APSHAI (EP)
0-22.....A:3 10:96

Alternate method
0-22.....A:3 10:DB

TERRAPIN LOGO (TER)
Format Target Disk
0-22.....10:96
Write protect backup before using.

TEST CONSTRUCTION (HOB)
0-22.....10:96

THREE MILE ISLAND (MU)
0-22

THRESHOLD (SOL)
0-22
1-23 STEP 22.....3B:1 A:1 4B:1 4D:8 50:1 (ERROR 6 OK)

THUNDERBOMBS (PEN)
0-10 step 2.....E:D5 F:AA 10:96 9:00 6:04 31:00
1-11 step 2.....E:D4 F:AA 10:96 9:00 6:04 31:00

TIME MANAGER (ICP)
0-22

TIME ZONE (SOL)
(Disks B thru L)
Use COPY DISK from MAIN MENU.
(Disk A)
0-22.....9:0
1.....3B:1 A:1 4B:1 4D:8 50:1

TORPEDO FIRE (SSI)
See three Alternates for Warp Factor

Alternate Method
0-22.....E:D4 10:B7 34:1

TRANSEND I (SSM)
0-23.....Error on Track 23 OK.

Alternate Method
0-22.....10:96

TRANSLYVANIA (PEN)
0-22.....E:0 10:96

Alternate Method
0-22 step 2.....10:96 9:0
1-21 step 2.....E:D4

TUBE WAY (DM)
0-22

TWERPS (SRS)
0.....9:0 10:96
1.5-E.5.....D:1 24:96 A:3 E:DD 10:DA 3B:40

TYPE ATTACK (SRS)
0.....10:96
1-22.....E:AD F:DA 10:DD 24:96 A:3 D:1

TYPING TUTOR (MIS)
Copy disk from main menu

U-BOAT COMMAND (SY)
0-22.....10:96 9:0 31:0 D:0 24:96 (Ignore Errors)

U-DRAW II (MU)
0-22.....D:1 31:0

ULTIMA II (SOL)
Use COPY DISK then
SECTMOD [T:3 S:0C]
CHANGE ADDRESSES 84 85 86 ALL TO EA.

Alternate Method
0-22.....10:96 9:0 34:1 31:0

Alternate method
0-23

ULTIMA /// (OS)
0-23.....9:0A:3 44:1 50:3 3B:1 A:1 D:1 10:96 50:1

ULYSSES & GOLDEN FLEECE (SOL)
0-22.....9:0
3.....3B:1 0A:1 4B:1 4D:8 50:1 (ERROR 6 OKAY)

Alternate Method
Use COPY DISK from main menu
3.....3B:1 0A:1 4B:1 4D:8 50:1 (ERROR 6 OKAY)

V.C. (AVH)
0-22

VERSA FORM (AST)
0-22

VISICALC (VCP)
0-16

Alternate Method
0-15

Alternate Method
0-16.....A:3

VISICALC II ENHANCED VERSION (VCP)
0-16.....Error 2 on Track 1 OK.

VISICALC //e 128K VERSION (VCP)
0-22.....10:96 24:96 D:1 9:0 31:0

VISICALC PRE-BOOT (VX)
0-22.....9:0 10:96

Alternate method
0-23.....10:96 9:0 3F:1

VISICALC FOR THE APPLE /// (VCP)
0-22.....10:96 24:96 D:1

VISISCHEDULE /// (VCP)
Copy disk from main menu.

VISIDEX, VISISCHEDULE, VISITERM, VISITREND/VISIPLOT (VCP)
Don't use Bit Copy. Use COPY DISK from MAIN MENU.

VISIFILE (VCP)
0-22.....10:96 34:1 36:2A 37:EB 3E:2

WARP FACTOR (SSI)
0-22

Alternate Method
0-22.....E:DB F:D5 10:DE

Alternate Method
0
1-22.....E:DB F:D5 10:DE 8:1

Alternate method
0-22
20.....9:0

WILDERNESS CAMPAIGN (SY)
0-22

WINDFALL (EW)
0-22.....10:96

WITNESS (IC)
0-22.....10:96 1E:BC

WIZARD & PRINCESS (SOL)
0-22

WIZARDRY (SIR)
Boot Side:
0-23.....10:96 24:96 D:1
Write protect back-up before using.

Scenario:
0-22.....10:96 24:96 D:1

Alternate for Boot Side
Use Copy Disk then
A-E.....10:96 24:96 D:1 4B:1
Write protect copy before using.

Alternate for Scenario Side
A-E.....10:96 24:96 D:1 4B:1
DO NOT write protect.

Alternate for both sides
Use COPY DISK from MAIN MENU then Bit Copy.
0.....D:1 10:96 24:96
A-E.....4B:1 (Error 6 OK)
Be sure to write-protect boot side.

Alternate for both sides
0-9.....10:96
0F-22
A-E.....D:1 4B:1 10:96 24:96
If Error 6 then recopy that track.

WIZ PLUS (DM)
0-22.....10:96 24:96 D:1

WIZ MAKER (ARS)
0-22.....D:1 24:96 10:96 34:1 8:1

WORLD HANDLER (SVS)
0-22

WORLD'S GREATEST BLACKJACK PROGRAM (AC)
0-22

WRITE AWAY (MWS)
Use Copy Disk

ZARDAX (CW)
0-22.....D:12 10:96 24:96

ZARGS (IN)
Same as Spider Raid

ZAXXON (DS)
1-12
0.....4B:1 9:0 (ERRORS OK)

Alternate method
0-13.....4B:1 D:1 10:96 24:96

Alternate method
0-13.....4B:1 9:0 10:96 24:96 19:CC 3C:1 (Error 1 OK)

Alternate method
1-12.....10:96
0.....4B:1 (Errors OK)
13

Alternate method
0-13
SECTMOD [T:0 S:07]DOS 3.3 PATCHED
CHANGE ADDRESSES 00-02 TO 4C C0 08.

Alternate method
3-12
0-2.....4B:1 9:0 10:96 24:96 19:CC 3C:1

ZOOM GRAPHICS (PHO)
0-22.....10:96 9:0

Alternate method
0-22.....10:24 9:0

Alternate method
0.....10:96
2-22 step 2.....9:0 8:1 3E:2
1-21 step 2.....E:D4

ZORK I II III (IC)
0-22.....10:96 1E:BC

Alternate Method
0-22

Alternate Method for Zork II
0-23.....10:96 9:0 3F:1

Hardware Solutions

Many readers complained that they couldn't use the softkeys because they didn't have the Integer Firmware card. Well, here's one solution...

The auto-start ROM is a mixed blessing. The auto-start feature allows programmers to create a Turnkey System.

The user need only insert a program disk and switch on the computer. The monitor ROM will automatically cause the disk to BOOT and the program will be up and running. The reset switch can be locked out. An unknowledgeable user cannot accidentally crash the program.

This is great for computer users who have no need or desire to learn about computers. But, for the hobby or business person who is trying to modify lines or fix a bug in a program, the auto-start ROM will make life miserable. It is all but impossible to stop a running program.

One solution is to not buy software that cannot be modified. Another solution is to purchase an Integer Firmware card. (The old F-8 monitor ROM does not have the auto-start feature) The price for this can range from $100 to $200, depending on whether it is purchased new or used.

A less expensive solution is to purchase just the F-8 monitor ROM for an Apple II from an Apple dealer and replace the autostart ROM in the Apple II + whenever program modifications are needed.

The procedure is simple. Care should be exercised, however, because the pins on the Integrated Circuit (IC) are easily bent. Read all of the following instructions thoroughly before beginning.

CAUTION: This procedure may void any dealer warranties!

1. Turn off the power to the computer. Remove the cord.

2. Remove the top cover and set it aside.

3. Touch the metal power supply case to discharge any static from your body. (The power supply is the large box on the left side.) Do this again before you handle the ROM.

4. Locate the F-8 ROM (see fig. 2). Using a small, flat screwdriver, gently pry up one side of the IC about 1/16th of an inch.

5. Gently pry up on the other end of the IC about 1/16th of an inch.

CAUTION: Be sure to pry up on the IC chip and not on the socket. (see fig. 1)

6. Repeat steps 4 and 5 until the IC is free.

7. Set the chip aside in a safe place. (If the F-8 ROM you bought comes with a case, use that.)

REMEMBER: Static is your worst enemy! Handle the chip as little as possible.

8. Pick up the chip you purchased and examine it. One end will have a notch and a small dot near one corner. The chip must be inserted with the notch and small dot pointing toward the keyboard.

CAUTION: Applying power with the chip in the socket backwards may destroy the chip and damage other components on the motherboard.

9. Insert the chip in the socket. You can prealign the pins on the chip by pressing them upon a flat surface. Be gentle and use even pressure. Insure that the chip is fully seated in its socket.

10. Replace the top cover and reconnect the power cord.

11. Turn on the computer. Your Apple II + will now emulate an Apple II cold start. When you see the asterisk prompt, type:

6 ctrl P return

To enter this line, press the "6" key. Then while holding the "CTRL" key down, press the "P" key. Then press the "RETURN" key. This will BOOT a disk in slot 6.

Follow these same steps when you wish to re-install the auto-start ROM.

figure 1

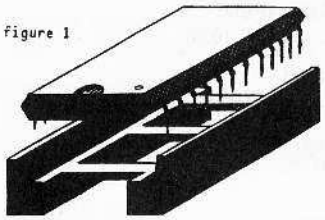
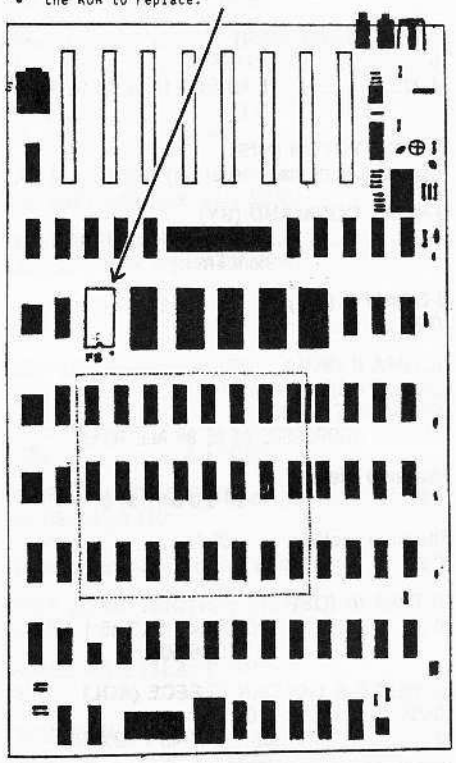


figure 2

Your Apple Motherboard showing the the ROM to replace.



Locksmith Parameters continued from page 30

Alternate method
s 00-09
0F-22.....36=01
0A-0E

WIZARDRY #2 (Knight of Diamonds) (SIR)
s 00-09 BY 1
s 0F-22 BY 1
s 0A-0E BY 1.....36=01
Uses nibble count.
Write protect before running.

WORD HANDLER (SVS)
00.....46=96 54=12
11-22
01-0C.....44=FF 45=DF 46=DE
NB type 8 errors O.K.

WORD HANDLER II (SVS)
00.....46=96 54=12 53=00
11-22
01-0C.....44=FF 45=DF 46=DE
For type 8 error, recopy track until good.

ZORK (IC)
00-22.....1E=0B
03.....4C=1B 57=00 E9=02
NB uses nibble count.

ZORK I (IC)
00-22.....46=96 40=14

ZORK II (IC)
00-22.....46=96 40=14

A MENU Hello Program

By Robb Canfield

Requirements:

APPLE II with 48K
DOS 3.3
Applesoft in ROM

MENU HELLO is a user-oriented program easily modified for individual needs. A menu HELLO program makes "turnkey" operation possible by providing a quick, simple and user-friendly way to LOAD, RUN, BLOAD or BRUN programs on a disk.

WARNING: MENU has a problem reading inverse or flashing files. MENU will print garbage for the file name and will generate errors if anything is done to that file by the Mini-Menu. MENU will only work properly on a normal catalog (one that prints only the unmodified catalog header and the file names in a normal fashion). A catalog will not be read properly by MENU if the catalog routine prints either how much space is left or that the disk is okay.

Most users have probably written a menu program of some sort, such as:

```
10 PRINT "1) SUPERCOPY 1.0"
20 PRINT "2) FILE FIXER"
30 PRINT: INPUT "SELECT ONE "; A$
40 PRINT CHR$(4) "RUN" A$
```

This is great for a few programs on selected diskettes, but writing a new menu program for each disk can be tiring at best and probably not worth the effort.

MENU is a more advanced type of menu program that allows the user to do a multitude of things with the directory (where DOS stores all the information printed when a CATALOG is done), from loading a program to locking or unlocking some or all of the programs on a disk.

Enter the listing for MENU in the order explained in "How To Enter MENU." Save the program. If MENU is used for the HELLO program, be sure that it is the one that runs first when the disk is booted. Do this by saving MENU under whatever file name is run when the disk boots or by initializing new disks so MENU is the HELLO program. This can be done by simply typing INIT MENU, followed by a V and the volume number desired (1 - 254).

Now run it. READING CATALOG will be visible in the center of the screen. At this time MENU is reading the catalog. After a few seconds the first part (page) of the catalog will be seen.

To page through it (when there are a lot of files in the catalog), use the left and right arrow keys.

To select a program, just type its letter code. This will cause the Mini-Menu to be entered. The Mini-Menu is self-prompting (it asks all the questions) and simple to use.

The only confusing part that may arise is when a binary file is run or loaded. The Mini-Menu will ask for the running or loading location. This is an optional choice. To load or run the binary file at other than the original address, enter the new address. Press return to use the original address.

Remember: Always precede a hex location with a "$".

It is possible to delete, rename, unlock, lock, load (bload), run (brun) a program and exec a file (depending on its type) using the Mini-Menu. The Mini-Menu also automatically updates the options for the user. This is done so that a locked file is not locked again and a text file isn't run or loaded. There is even an lock/unlock all mode that will lock or unlock every file on a disk. Whenever a file is selected, all valid commands are displayed along with their explanations. All commands are normal keys. No control characters are used in the Mini-Menu.

When a catalog is displayed, the file names and their status codes are shown. This is a typical example:

***A 089 SUPER INVADERS**

The following is an explanation of the various parts of the file status code preceding the actual file name (SUPER INVADER):

***** - The asterisk means the file is locked. If the file were unlocked this would be a space.

A - The 'A' tells DOS that the program is in Applesoft. 'I' means Integer, 'B' means binary, and 'T' means text file.

089 - These three numbers represent the length of the program or text file in sectors. This number will always be three digits.

The status section of the file takes up seven characters. The file name takes up 30 characters (DOS always reserves 30 characters for the file name. All characters after the actual file name are spaces and printed as such.) DOS also prints a carriage return after it finishes printing the file name, so the following file name is put on the next line down.

The machine language portion of MENU fools DOS into putting the file name and its status into the string array (NA$) instead of on the screen. Everything that is normally printed on the screen is put into the array instead, including the catalog header. The information can then be used as desired.

How to enter MENU

MENU consists of two parts:
1. A machine language listing
2. A BASIC listing

The machine language routine must be entered from the monitor (*) before the BASIC listing is typed.

First of all, make sure that the Applesoft pointers are set correctly by typing:

FP

and pressing return

Then enter the monitor by typing:

CALL -151

and pressing return.

Now type the following lines: (Don't forget to press return after each line.)

```
0800:00 37 08 00 00 B2 A9 28
0808:8D 53 AA A9 08 8D 54 AA
0810:A9 36 8D 55 AA A9 08 8D
0818:56 AA A0 08 B1 6B 8D 2C
0820:08 C8 B1 6B 8D 2D 08 60
0828:29 7F 8D FF FF EE 2C 08
0830:D0 03 EE 2D 08 60 00 00
0838:00 00
```

Figure 1

While still in the monitor, set the end-of-program counter to point to location $0839 by typing:

AF:39 08

Now type:

800.839

The figures on the screen are what is called a hex dump. Compare it with the one shown in Figure 1. If any line is not the same, reenter that entire line. After the corrections are completed, return to Applesoft by typing:

3D0G

(Or, for those with the Autostart ROM, just press reset.)

Line 0 should be the only line of the program at this time. The machine language subroutine is hidden behind the REM statement and will not be affected by RENUMBER or line changes if it is the first line in the program.

Now, enter the BASIC listing as shown, and save it to the disk. There is no need to type the REMs in the BASIC listing, but it may help when modifying the program later.

Caution: Do NOT under any conditions delete or modify line 0. Any change to line 0 will cause MENU to run incorrectly.

A Line-by-line Explanation

Here is a line-by-line description of MENU. The subtitles are the same as the ones used in the listing itself. Line numbers precede their explanations.

0: The machine language part of MENU is hidden within the REM statement on line 0. (see "How to enter MENU").

Initialize string storage (10-50)

10: First, reserve memory for 104 file names and the catalog header (NA$).

20: Each element of NA$ must be set to 38 characters. This is done by creating a string (HE$) consisting of 40 " = 's and running through a loop that sets each element of NA$ to the first 38 characters of HE$ (using LEFT$), except for NA$(0) which is the catalog header and, as such, only needs 19 characters.

30: The loop runs backward so that the NA$(0) is the first element filled when a catalog is done.

32-33: Get the amount of memory available in the machine (48K or 32K) by checking location 984 (dec). This value is then POKEd into the machine subroutine hidden in line 0.

40: Set enough room aside for the catalog header.

Reread catalog (55-220)

55-200: CALL the machine subroutine and read the catalog into memory. This is the hard reentry point. Going to this line will cause all information about the current catalog to be deleted and a new one read.

206-220: Find the last page of the catalog. This page minus one is stored in the variable MA.

Print file names (230-310)

230, 240: Clear window and print the catalog header. Line 230 is is also known as the SOFT entry point.

250: Print the boundary (HE$).

270: Control the wrap-around feature.

320-370: Look for a blank file name (all " = 's). If it is blank, then get a file selection. Otherwise, continue printing file names until both columns are filled.

Get file selection (440-550)

440-460: Set text window and print page number and other information.

470: Get a file selection and check if user:

475, 480: page through the catalog.

500: help.

510: exit the menu.

520: read another catalog. If a file selection was entered, make sure that the file exists.

530: If it doesn't, ignore this selection and get another.

550: Otherwise, go to the Mini-Menu.

The mini-Menu

The Mini-Menu is divided into 12 main routines.

1. Initialize Defaults
2. Center Printing
3. UNLOCK File
4. LOCK File
5. RUN/BRUN File
6. DOS Control
7. LOAD/BLOAD File
8. DELETE File
9. RENAME File
10. LOCK/UNLOCK all
11. Exit Mini-Menu
12. EXEC Text File

NOTE: The word "flag" (which appears below) is used to denote a variable whose value will cause certain actions to be taken. The flags are:

B is the Binary/ Text flag.
- B = 0 Applesoft or Integer file.
- B = 1 Binary file.
- B = 2 Text file.

L is the Lock/Unlock flag.
- L = 0 Unlocked.
- L = 1 Locked.

Initialize defaults (560-650)

560,565: Print MINI-MENU, then set A$ equal to the proper file name and print it.

570: Set default options (LOCK, LOAD, RUN).

575: If the first character of A$ is an asterisk (*), change the lock option to unlock and set the lock flag (L = 1).

580: Use the second character of A$ to set the file type flag. If it is binary, change the options from LOAD to BLOAD, RUN to BRUN, and set the binary flag (B = 1).

585: If the file type is text, change the run option to EXEC and set the text flag (B = 2).

590-610: Print all options available. Do not print the load option if the file is text.

615-630: Prepare for future errors and redefine A$ as just the file name (the status is removed).

650: GET choice.

Lock file (660-700)

This routine will lock a file only if the user confirms the action and the lock flag is set.

660: Check for the K command (LOCK). If not K, then go to routine 4 (UNLOCK).

670: Otherwise, confirm action (GOSUB 3000).

680: If the lock request is not confirmed, go back to the Mini-Menu.

690,700: If confirmation is given, lock the file and change file name NA$(X) to include an asterisk (*). Return to the main menu.

Center printing

Lines 690-699 are similar in appearance to routines in each of the other sections, so it is numbered 3 and will be referred to each time the same function is performed.

690: Center the text vertically.

692, 694: Center the text horizontally.

699: Print the operation being performed (in this case the operation is LOCK) and perform that action. This is where major revisions will be found (compare lines 699, 745, 860, 979, and 1100).

Unlock file (710-760)

This routine will unlock a file only if the request is confirmed and the lock flag is not set.

710: Check for the U command (UNLOCK). If the lock flag is set or the U command was not entered, go to routine 5 (RUN/BRUN).

720: If the U command is entered and the lock flag is not set, confirm the action (GOSUB 3000).

740: Lets us know what's going on (see routine 3); then return to main menu.

Run/brun file (765-800)

765: If the R command (RUN/BRUN) was not entered, continue to routine 7 (LOAD/BLOAD).

770: Set the DOS command to BRUN (C$ = BRUN).

800: If the binary flag is zero, set the DOS command to RUN (C$ = RUN) and go to routine 6 (DOS CONTROL).

DOS control (810-890)

This routine has two entry points:

A. OPTIONAL ADDRESS (line 810)

B. DEFAULT ADDRESS (line 830)

Line 810 will ask where to put the file (for BRUN or BLOAD). If the return key is pressed, the binary program will default to its normal location. If a different location is desired, then enter that location (in hex or decimal). A hex location MUST be preceded by a dollar sign ($).

Enter at line 820 to bypass the optional BLOAD/BRUN address.

To enter either routine, C$ must equal the DOS command that is associated with the desired action and A$ must be equal to the file name . To enter the second routine, L$ and B$ must both be cleared.

820: Control the default address for binary files.

830: Get confirmation of the action to be performed (GOSUB 3000).

840: If confirmation is not given, go back to the Mini-Menu.

850-880: If confirmation is given, print the action (see routine 3) and call DOS to do it.

890: Return to the main menu.

Load/bload file (900-930)

900: Check for the L command (LOAD or BLOAD). If not L, go to routine 8 (DELETE).

910: Otherwise, set the default to BLOAD.

920: If the binary flag is clear (B is not equal to 1), change to LOAD (C$ = LOAD) and enter routine 6 (DOS Control) at the second entry point (line 820).

930: If the binary flag is set, enter routine 6 at the first entry point (line 810).

Delete file (940-980)

940: Check for the D command (DELETE). If not D, go to routine 9 (RENAME).

950: Otherwise, check the lock flag to see if the file is locked. If so, issue a warning (THIS FILE IS LOCKED).

960: Confirm action (GOSUB 3000).

965: If confirmation is not received, then return to the Mini-Menu.

970: If confirmation is received, center the text (see routine 3) and unlock the file (in case it was locked); then delete it.

980: Restart the program.

Rename file (990-1120)

990: Check for the C command (RENAME). If not C, go to routine 10 (UNLOCK/LOCK ALL).

1010: Otherwise, check the lock flag to see if the file is locked. If it is, print a warning message.

1030: Confirm action (GOSUB 3000).

1050: If confirmation is not received, return to the Mini-Menu.

1070: Otherwise, INPUT a new file name.

1075: If the file name is greater than 30 characters, print an error message and go back to the Mini-Menu.

1080: If the new name is nothing (return was pressed), then return to the Mini-Menu.

1090-1100: Center the text (see routine 3) and do the required action.

1110: LOCK the file if the old file name was locked (check the lock flag).

1115: Redefine the old file name, NA$(X), to be the first seven characters (status) of the old file name plus the new name.

1120: Return to the main menu.

Lock/unlock all (1160-1270)

This routine will allow us to UNLOCK or LOCK ALL of the files on the disk.

1160: If A was not selected, go to routine 11 (EXIT).

1170: Otherwise, get the choice (LOCK or UNLOCK).

1180: If the L or U key is not pressed, go back to the Mini-Menu.

1190, 1200: Depending upon whether U or L was pressed, set the DOS command to LOCK or UNLOCK.

1210: Center the text (see routine 3).

1220-1260: Run through a loop performing the DOS action (LOCK or UNLOCK) and modifying each file name, NA$(X) to contain an asterisk (locked) or a blank (unlocked) depending on the action taken.

1230: When a file consisting of " = 's is found, exit to the main menu.

Exit mini-Menu (1280)

1280: If the X, ctrl X, return or excape key is pressed, go back go the main menu. Otherwise, go to routine 12 (EXEC).

Exec text file (1282-1300)

1282: If the E command is not selected, or if the text flag isn't set (B is not equal to 2), then go to the Mini-Menu.

1284: Otherwise, set DOS command to EXEC (C$ = EXEC) and clear L$ and B$. Go to the second entry point in routine 6 (line 830).

Error control

Control of error messages is done with ONERR GOTOs. Depending upon where the ERRor occurred, these are the responses:

ERR IN NEW NAME. PLEASE TRY AGAIN

1310-1340: The new name of a file (change command) is illegal. Lock the file and print message. Return to the Mini-Menu.

!!! -UNABLE TO READ DIRECTORY !!!

1350-1360: DOS was unable to read the disk - probably an I/O ERROR. Print message and exit the program.

ERR IN LOAD ADDRESS

1370: A bad loading address (used in BLOAD/BRUN command). Print the message to the screen and go to line 1340 to get a keypress. Reenter the Mini-Menu.

STRANGE ERR. I WILL REREAD THE CATALOG

1380: A strange error was encountered, so reread the catalog. Go back to the hard entry point (line 55).

Print help (3010-3080)

3010-3080: Print the instructions for the help mode (invoked by pressing the escape key when in the main menu).

Confirmation of action (3000)

3000: Asks for Y or N in order to confirm an important DOS action.

Commands for MENU

RETURN :	Read a new catalog into memory.
ctrl C :	Display a disk catalog. Does not change the catalog in memory.
← :	Page backward thru the menu.
→ :	Page forward thru the menu.
A - Z :	Used to make single key selections. Will enter the mini-menu.
ctrl X :	Exit the program
ESC :	"HELP Mode". Displays the commands in abbreviated form.

The BASIC listing starts here.

```
0 --- SEE TEXT ---
5 TEXT : HOME
6 REM INITALIZE STRING STORAGE
10 DIM NA$(104)
20 FOR X = 1 TO 40:HE$ = HE$ + "=":
   NEXT X
30 FOR X = 104 TO 1 STEP - 1: NA$(X)
   = LEFT$ (HE$,38): NEXT
32 ST% = PEEK (984)
33 FOR X = 2058 TO 2073 STEP 5: POKE
   X,ST%: NEXT
40 NA$(0) = LEFT$ (HE$,19)
50 D$ = CHR$ (4):G$ = CHR$ (7) +
   CHR$ (7)
52 REM HARD ENTRY (RE-READ CATALOG)
55 TEXT : HOME
60 VTAB 12: HTAB 12: PRINT "READING
   CATALOG ": VTAB 12: HTAB 29
180 REM CALL MACHINE SUBROUTINE AND
   GET A CATALOG
190 ONERR GOTO 1350
200 CALL 2054: PRINT D$"CATALOG"
202 REM GET MAXIMUM PAGE
206 FOR X = 1 TO 3: IF MID$ (NA$(X *
   26 + 1),2,1) = "=" THEN 210
208 NEXT
210 MA = X - 1: POKE 216,0: PRINT
   D$"PR#0": PRINT D$"IN#0": PRINT
220 X = 0
225 REM PRINT OUT FILENAMES ALSO
   SOFT ENTRY POINT
230 HOME
240 VTAB 1: HTAB 10: PRINT MID$
   (NA$(0),3,15)
250 VTAB 3: PRINT HE$: VTAB 19:
   PRINT HE$
270 IF X < 0 THEN X = MA
280 IF X > (MA) THEN X = 0
290 POKE 34,4: POKE 35,18
300 VTAB 5: HTAB 1: PRINT
310 HOME
315 REM PRINT FILENAMES ON LEFT SIDE
   OF SCREEN
320 FOR Y = 1 TO 13
322 A$ = MID$ (NA$(26 * X + Y),8,17)
325 IF MID$ (A$,2,1) = "=" THEN Y =
   Y - 1: GOTO 440
330 A$ = CHR$ (64 + Y) + " " + A$
350 PRINT A$
360 NEXT
370 VTAB 5
375 REM PRINT FILENAMES ON RIGHT
   SIDE OF SCREEN
380 FOR Y = 1 TO 13
382 A$ = MID$ (NA$(26 * X + Y +
   13),8,17)
385 IF MID$ (A$,2,1) = "=" THEN MA =
   X:Y = Y + 12: GOTO 440
390 A$ = CHR$(65+Y+12) + " " + A$
410 HTAB 21: PRINT A$
420 NEXT
427 REM CHECK FOR LEGAL SELECTION
430 Y = 26
435 REM PRINT PAGE # AND GET FILE
   SELECTION
440 POKE 34,20: POKE 35,24
450 VTAB 22: HTAB 30: PRINT "PAGE "X
   + 1" OF "MA + 1;
460 VTAB 24: HTAB 10: INVERSE :
   PRINT "PRESS 'ESC' FOR HELP";:
   NORMAL
470 VTAB 21: HTAB 1: PRINT "SELECT
   ONE (PRESS A KEY ) ";: GET A$:
   PRINT
475 IF A$ = CHR$ (8) AND MA < > 0
   THEN X = X - 1: GOTO 270
480 IF A$ = CHR$ (21) AND MA < > 0
   THEN X = X + 1: GOTO 270
490 IF A$ = CHR$ (24) THEN 1300
500 IF A$ = CHR$ (27) THEN 3010: REM
   IS IT ESC
510 IF A$ = CHR$ (13) THEN 55
520 IF A$ = CHR$ (3) THEN TEXT :
   HOME : PRINT D$"CATALOG": GET
   B$: PRINT : PRINT : HTAB 8:
   PRINT "HIT ANY KEY TO CONTINUE
   ";: GET B$: GOTO 230
```

```
530 IF A$ > CHR$ (Y + 64) THEN 470
540 IF A$ < "A" OR A$ > "Z" THEN 470
550 A = ASC (A$) - 64
555 REM INITALIZE DEFAULTS FOR THE
    MINI-MENU AND PRINT OUT CHOICES
560 TEXT: HOME: VTAB 2: HTAB 16:
    PRINT "MINI-MENU"
565 VTAB 7: HTAB 3:A$ = NA$(X * 26 +
    A): PRINT A$
570 L = 0:B = 0:L$ = "K) LOCK":B$ =
    "L) LOAD":B1$ = "R) RUN"
575 IF LEFT$ (A$,1) = "*" THEN L =
    1:L$ = "U) UNLOCK"
580 IF MID$ (A$,2,1) = "B" THEN B =
    1:B$ = "L) BLOAD":B1$ = "R)
    BRUN"
585 IF MID$ (A$,2,1) = "T" THEN B =
    2:B$ = "E) EXEC"
590 PRINT L$: PRINT B$: IF B < > 2
    THEN PRINT B1$
595 PRINT "D) DELETE": PRINT "C)
    CHANGE PROGRAM NAME"
600 PRINT "A) LOCK/UNLOCK ALL"
610 PRINT "X) EXIT TO CATALOG"
615 ONERR GOTO 1370
620 B$ = "": FOR Y = 37 TO 8 STEP -
    1: IF MID$ (A$,Y,1) = " " THEN
    NEXT
630 A$ = MID$ (A$,8,Y - 7)
640 PRINT
650 PRINT "ENTER YOUR CHOICE > ";:
    GET B$: PRINT: PRINT
655 REM LOCK FILE
660 IF B$ < > "K" OR L < > 0 THEN
    710: REM LOCK FILE ONLY IF IT IS
    UNLOCKED NOW
670 PRINT "LOCK "A$;: GOSUB 3000
680 IF B$ < > "Y" THEN 560
690 HOME: VTAB 12
692 Y = 40 - LEN (A$) - LEN (B$) - 8:
    IF Y < 2 THEN Y = 2
694 HTAB (Y / 2)
699 PRINT "LOCKING "A$: PRINT
    D$"LOCK"A$
700 NA$(X * 26 + A) = "*" + RIGHT$
    (NA$(X * 26 + A),37): GOTO 230
705 REM UNLOCK FILE
710 IF B$ < > "U" OR L < > 1 THEN
    760: REM UNLOCK FILE ONLY IF
    FILE IS NOW LOCKED
720 PRINT "UNLOCK "A$;: GOSUB 3000
730 IF B$ < > "Y" THEN 560
740 HOME: VTAB 12:Y = 40 - LEN (A$)
    - 11: IF Y < 2 THEN Y = 2
745 HTAB Y / 2: PRINT "UNLOCKING
    "A$: PRINT D$"UNLOCK"A$
750 NA$(X * 26 + A) = " " + RIGHT$
    (NA$(X * 26 + A),37): GOTO 230
760 IF B = 2 THEN 940
762 REM RUN/BRUN FILE
765 IF B$ < > "R" THEN 900
770 L$ = "":B$ = "":C$ = "BRUN"
800 IF B = 0 THEN C$ = "RUN": GOTO
    830
805 REM IF FILE IS BINARY GET OP
    TIONAL LOAD ADDRESS
810 PRINT: PRINT "WHERE TO "C$;: IN
    PUT " (DEC $HEX ) > ";L$: PRINT
820 IF L$ < > "" THEN B$ = " AT "
830 PRINT C$" "A$B$L$;:E$ = B$:
    GOSUB 3000:T$ = E$:E$ = B$:B$ =
    T$
840 IF E$ < > "Y" THEN 560
845 ONERR GOTO 1370
850 PRINT: HOME: VTAB 12
852 Y = 40 - LEN (A$) - LEN (C$) -
    LEN (L$) - LEN (B$) - 5
854 IF Y < 2 THEN Y = 2
856 HTAB (Y / 2)
860 PRINT C$"ING "A$B$L$
870 IF L$ < > "" THEN L$ = ",A" + L$
880 PRINT D$C$A$L$
890 GOTO 230
895 REM LOAD/BLOAD SELETED FILE
900 IF B$ < > "L" THEN 940
910 B$ = "":L$ = "":C$ = "BLOAD"
920 IF B < > 1 THEN C$ = "LOAD": GOTO
    820
930 GOTO 810
935 REM DELETE FILE
940 IF B$ < > "D" THEN 990
950 IF L = 1 THEN PRINT G$"THIS FILE
    IS LOCKED"
960 PRINT: PRINT "DELETE "A$;: GOSUB
    3000
965 IF B$ < > "Y" THEN 560
970 PRINT: HOME: VTAB 12
972 Y = 40 - LEN (A$) - 9
974 IF Y < 2 THEN Y = 2
976 HTAB (Y / 2)
979 PRINT "DELETING "A$: PRINT
    D$"UNLOCK"A$: PRINT D$"DELETE"A$
980 RUN
985 REM RENAME FILE
990 IF B$ < > "C" THEN 1160
1000 ONERR GOTO 1310
1010 IF L = 1 THEN PRINT G$"THIS
    FILE IS LOCKED"
1020 PRINT
1030 PRINT "RENAME "A$;: GOSUB 3000
1040 PRINT
1050 IF B$ < > "Y" THEN 560
1060 PRINT
1070 PRINT "CHANGE "A$" ";: INVERSE:
    PRINT "TO";: NORMAL: PRINT " ";:
    INPUT B$
1075 IF LEN (B$) > 30 THEN HOME:
    VTAB 12: GOTO 1320
1080 IF B$ = "" THEN 560
1090 HOME: VTAB 12
1092 Y = 40 - LEN (A$) - LEN (B$) -
    12
1094 IF Y > 38 OR Y < 3 THEN Y = 2
1096 HTAB (Y / 2)
1100 PRINT "CHANGING "A$" ";: IN
    VERSE: PRINT "TO";: NORMAL:
    PRINT " "B$: PRINT D$"UNLOCK"A$:
    PRINT D$"RENAME"A$","B$
1110 IF L = 1 THEN PRINT D$"LOCK"B$
1115 NA$(X * 26 + A) = LEFT$ (NA$(X
    * 26 + A),7) + B$
1120 GOTO 230
1130 REM UNLOCK/LOCK ALL FILES
1160 IF B$ < > "A" THEN 1280
1170 PRINT: PRINT "LOCK/UNLOCK ALL
    FILES ";: INVERSE: PRINT "L/U";:
    NORMAL: GET B$
1180 IF B$ < > "L" AND B$ < > "U"
    THEN 560
1190 IF B$ = "U" THEN B$ =
    "UNLOCK":T$ = " "
1200 IF B$ = "L" THEN B$ = "LOCK":T$
    = "*"
1210 PRINT: HOME: VTAB 11: HTAB (40
    - LEN (B$) - 6) / 2: PRINT
    B$"ING ALL"
1220 FOR Y = 1 TO 105
1230 IF MID$ (NA$(Y),8,2) = "=="
    THEN 230
1240 VTAB 13: HTAB 10: PRINT MID$
    (NA$(Y),8,30)
1250 PRINT D$B$ MID$ (NA$(Y),8,30)
1252 NA$(Y) = T$ + RIGHT$
    (NA$(Y),37)
1260 NEXT
1270 GOTO 230
1275 REM EXIT MINI-MENU?
1280 IF B$ = "X" OR B$ = CHR$ (27) OR
    B$ = CHR$ (13) OR B$ = CHR$ (24)
    THEN 230
1281 REM EXEC TEXT FILE
1282 IF B < > 2 OR B$ < > "E" THEN
    560
1284 C$ = "EXEC":L$ = "":B$ = "":
    GOTO 830
1300 TEXT: HOME: END
1305 REM ERR ROUTINES
1310 IF L = 1 THEN PRINT D$"LOCK"A$
1320 PRINT: PRINT G$" ERR IN NEW
    NAME. PLEASE TRY AGAIN"
1330 ONERR GOTO 1350
1340 PRINT: HTAB 7: PRINT "PRESS ANY
    KEY TO CONTINUE";: GET B$: GOTO
    560
1350 POKE 216,0: PRINT D$"PR#0":
    PRINT D$"IN#0": PRINT
1360 HOME: VTAB 12: HTAB 3: PRINT
    G$"!!! - UNABLE TO READ DIRECTO
    RY !!!": POKE 216,0: END
1370 HOME: VTAB 12: IF LEFT$ (L$,3)
    = ",A$" THEN HTAB 10: PRINT
    G$"ERR IN LOAD ADDRESS": GOTO
    1340
1380 PRINT G$"STRANGE ERR. I WILL
    RE-READ THE CATALOG": PRINT:
    HTAB 6: PRINT "PRESS ANY KEY TO
    CONTINUE";: GET B$: GOTO 55
2999 REM SUBROUTINE TO GET KEYPRESS
3000 PRINT " ";: INVERSE : PRINT
    "Y/N";: NORMAL : GET B$: PRINT :
    RETURN :
3005 REM PRINT INSTRUCTIONS
3010 TEXT : HOME : VTAB 2: HTAB 16:
    PRINT "COMMANDS": PRINT : PRINT
    HE$: PRINT
3020 PRINT "> RIGHT ARROW MOVE FOR
    WORD THRU MENU": PRINT
3030 PRINT "< LEFT ARROW MOVE BACK
    WORD THRU MENU": PRINT
3040 INVERSE : PRINT "M";: NORMAL :
    PRINT " 'RETURN' GET'S A NEW
    CATALOG": PRINT
3050 INVERSE : PRINT "C";: NORMAL :
    PRINT " CTRL 'C' GET A NORMAL
    CATALOG": PRINT
3060 INVERSE : PRINT "X";: NORMAL :
    PRINT " CTRL 'X' EXIT MENU":
    PRINT
3070 PRINT "    ANY LETTER A-Z GO'S
    TO MINI-MENU": PRINT
3080 VTAB 20: HTAB 7: PRINT "PRESS
    ANY KEY TO CONTINUE";: GET B$:
    PRINT : GOTO 230
4000 REM
4005 REM PROGRAM WRITTEN
4010 REM BY
4020 REM ROBB CANFIELD
```

5 - $0050
6 - $BA76
10 - $6753
20 - $F009
30 - $6DF9
32 - $4E1A
33 - $973D
40 - $B813
50 - $5224
52 - $46F9
55 - $591E
60 - $C079
180 - $FFD9
190 - $9236
200 - $F979
202 - $CFC7
206 - $2F7F
208 - $39E4
210 - $0EB0
220 - $A834
225 - $930B
230 - $B03D
240 - $7114
250 - $53C0
270 - $3FD6
280 - $8EE0
290 - $B61D
300 - $8E6D
310 - $E451
315 - $6865
320 - $96F1
322 - $530F
325 - $5A22
330 - $968F
350 - $774A
360 - $FA9B
370 - $0255
375 - $7CAD
380 - $8ADB
382 - $CED7
385 - $957B
390 - $CC80
410 - $994B
420 - $BF31
427 - $10FA
430 - $09D0
435 - $A56A
440 - $4E50
450 - $9F95
460 - $80DA
470 - $9BE8
475 - $EDB6
480 - $73A3
490 - $C7A4
500 - $0C53
510 - $0E7F
520 - $C11B
530 - $8219
540 - $0AB7
550 - $90BB
555 - $CD3E
560 - $F026
565 - $B429
570 - $C7A2
575 - $BAD1
580 - $2A17
585 - $35F4
590 - $7D58
595 - $5D54
600 - $0694
610 - $5646
615 - $C21B
620 - $DF34
630 - $B843
640 - $3041
650 - $BB2E
655 - $C489
660 - $788E
670 - $9C37
680 - $12B4
690 - $BFFB
692 - $9599
694 - $10DA

Menu checksums

699 - $CEE4
700 - $E406
705 - $8C41
710 - $8A37
720 - $F944
730 - $A8B3
740 - $2481
745 - $94F8
750 - $CD4F
760 - $1823
762 - $306C
765 - $282D
770 - $47A2
800 - $6F5C
805 - $74E9
810 - $52A7
820 - $7A73
830 - $57BD
840 - $B575
845 - $05FA
850 - $44E0
852 - $A075
854 - $4432
856 - $8723
860 - $E2A9
870 - $1F26
880 - $DCC7
890 - $36AF
895 - $EBAB
900 - $73CF
910 - $E6AE
920 - $BB3A
930 - $3A13
935 - $A733
940 - $4302
950 - $C867
960 - $ED0A
965 - $28FE
970 - $E2D1
972 - $7310
974 - $3D99
976 - $AA78
979 - $75B0
980 - $2356
985 - $C607
990 - $2A88
1000 - $5740
1010 - $8AD5
1020 - $9C50
1030 - $0713
1040 - $97CC
1050 - $B008
1060 - $88E3
1070 - $C09D
1075 - $5314
1080 - $C47B
1090 - $B32D
1092 - $E878
1094 - $03D3
1096 - $A9C3
1100 - $95AB
1110 - $90E5
1115 - $88D9
1120 - $308C
1130 - $2CD4
1160 - $6EB4
1170 - $A57B
1180 - $6FE8
1190 - $385F
1200 - $12F2
1210 - $06FE
1220 - $F09E
1230 - $C2FA
1240 - $0FDC
1250 - $5DD3
1252 - $2A22
1260 - $C22D
1270 - $09CA
1275 - $FFEF
1280 - $CDBB
1281 - $8051
1282 - $EAA3
1284 - $9331
1300 - $71B1
1305 - $2235
1310 - $755E
1320 - $C5B7
1330 - $7FB8
1340 - $EACB
1350 - $8E58
1360 - $7B0C
1370 - $29D9
1380 - $6010
2999 - $2C27
3000 - $D274
3005 - $6084
3010 - $F25D
3020 - $1559
3030 - $576C
3040 - $A57B
3050 - $5605
3060 - $26A9
3070 - $E475
3080 - $4460
4000 - $EB52
4005 - $7C2D
4010 - $7A01
4020 - $AE74

Using Both Sides of Your Diskettes

You can increase your disk storage from 143,360 to 286,720 bytes by simply using both sides of a 'single-sided" disk. All you need to accomplish this feat is a standard hole punch.

Flip it

The only thing that prevents the use of both sides of single-sided diskettes is that they are effectively write-protected. In other words, there is no notch on the flip side.

Take two floppies and flip one over so that they are facing each other. Mark where and how far in the notches are and then use the hole punch to cut the second notch. Now, initialize the disk normally.

Don't flip it!

Makers of minifloppies and disk drives do not recommend that you use both sides of your diskettes if you have a one-head drive because:

1. When the drive head is applied to one side, a felt loading pad is pressed against the other side. That pad accumulates oxide particles that may scour the reverse side. When flipped, the contaminated pad may then scour the prime side as well. This may lead to premature loss of data and the accumulation of read errors that may go unnoticed or be intermittent, making the drive unreliable.

2. The direction of rotation is reversed when the diskette is flipped and this may dislodge oxide particles that have accumulated on the liner material. The results would be similar to those in (1).

Neither problem occurs on a two-head drive because the pressure pad is replaced by another write-head and the direction of rotation does not change.

What to do about flaws

Sometimes the flip side of your one-sided diskettes will contain flaws. These are areas where the oxide coating on the disk are too thin to reliably store data.

To avoid losing your valuable files, it would be wise to check the flip side before use. It's also a good idea to check the front side. Sometimes DOS will write a marginal address or data mark. The disk will then appear to have a flaw but reinitializing the disk will clear this.

There are at least two software packages (programs) that will check your disks for flaws and then alter the Volume Table Of Contents (VTOC) so that these bad sectors are defined as already used:

1. Disk Prep ($25), by Sympathetic Software, 9531 Telhan Dr., Huntington Beach, CA 92646

2. Disk Recovery ($30), by Sensible Software, Inc., 24011 Seneca, Oak Park, MI 48237

Diskedit Checksums, continued from page 11

10A0:20 CC 0A A9 3C AA AC 00 $5466
10A8:C0 30 07 CA D0 F8 E9 01 $C0E9
10B0:D0 F3 60 A9 15 20 8F 0A $2D11
10B8:A0 1E AD 34 08 20 D1 10 $03FC

10C0:AE 6D 08 F0 08 A9 A0 91 $8196
10C8:28 C8 CA D0 F8 60 AD 2B $CB58
10D0:08 48 AE 2F 08 D0 2A 8E $18C4
10D8:6D 08 A9 A4 91 28 C8 68 $C06B
10E0:48 4A 4A 4A 4A 09 B0 C9 $4ECF
10E8:BA 90 02 69 06 91 28 C8 $3907
10F0:68 29 0F 09 B0 C9 BA 90 $6022
10F8:02 69 06 91 28 C8 84 24 $5C10
1100:60 A2 02 8E 6D 08 A2 B0 $1043
1108:68 C9 64 90 12 E8 E9 64 $684E

1110:C9 64 B0 F9 CE 6D 08 48 $9BE1
1118:8A 91 28 C8 A2 B0 68 C9 $9BFB
1120:0A 90 0A E8 E9 0A C9 0A $3B75

1128:B0 F9 CE 6D 08 48 AD 6D $39DC
1130:08 C9 02 F0 04 8A 91 28 $1D64
1138:C8 68 09 B0 91 28 C8 60 $AB0C
1140:A2 15 86 23 A2 00 8E 6D $1C5F
1148:08 E8 8E 72 08 20 58 FC $092E
1150:E8 20 4A F9 AD 01 09 20 $E0DE
1158:06 12 AD 02 09 20 06 12 $E4F1

1160:20 62 FC 20 62 FC A2 0B $8105
1168:A0 02 20 62 FC 20 09 12 $31A1
1170:BD 00 09 20 06 12 E8 88 $4274
1178:D0 F6 BD 00 09 E8 2A 48 $63D9
1180:90 08 A9 AA 20 ED FD 4C $CA56
1188:8D 11 20 09 12 A0 00 68 $455D
1190:4A F0 04 C8 4A 90 FC B9 $9425
1198:D2 11 20 ED FD 20 09 12 $7F5C
11A0:A0 1E 8C 6C 08 BD 00 09 $3DE1
11A8:85 E0 4A 4A 4A 4A 4A A8 $409C

11B0:B1 E9 29 F0 18 65 E0 C9 $DFA2
11B8:80 30 06 C9 A0 10 02 A9 $FD3E
11C0:AE 20 ED FD E8 CE 6C 08 $2D2A
11C8:D0 DB E8 E8 D0 9A CA 86 $8F69
11D0:E4 60 D4 C9 C1 C2 D3 D2 $077B
11D8:C1 C2 AD 2B 08 C9 FF D0 $7CCA
11E0:05 A9 A0 8D 2B 08 4A 4A $04CD
11E8:4A 4A 4A A8 B9 F6 11 18 $6229
11F0:6D 2B 08 4C ED FD C0 80 $5E68
11F8:80 40 40 00 00 C0 A9 A4 $FC93

1200:20 ED FD AD 2B 08 20 DA $0B6C
1208:FD A9 A0 4C ED FD AE 2F $7FBD
1210:08 F0 EB AE 2B 08 A9 00 $BD61
1218:20 24 ED 4C 09 12 00 00 $91EC
1220:00 $CE11

Inside Castle Wolfenstein

By Robb Canfield

REQUIREMENTS:
Super IOB program
Castle Wolfenstein by Muse Software

Castle Wolfenstein is an arcade-adventure game using hi-res graphics. You are an escaped prisoner of war, trapped in a castle full of Nazi guards and SS troops. You must find the path to freedom and maybe a set of War Plans that are also in the castle.

The game is enjoyable and very addictive. Unfortunately it has one rather annoying routine — every time you run into a wall the screen flickers and a horrible noise issues from the speaker.

After playing quite a few games, I became frustrated by this sound and resolved to eliminate it. The first problem that I encountered was our old enemy, software protection.

The Lock

Castle Wolfenstein is on a modified 13-sector disk that will boot on 13 or 16-sector Apples. The only protection used is to write even sector numbers to the disk. This means that the sectors step by two (ie. 0,2, 4,6,8,10). In order to break this protection scheme, read "Super IOB".

Making Changes

I located three different sound routines. The locations to change are listed in Figure 1 in the form of pokes. To turn these routines ON or OFF follow these steps:

1) Load the file to change

BLOAD @WOLF

2) POKE in the desired change(s).
3) Save the modified file

BSAVE @WOLF,A$810, L$16EB

Figure 1

Sound Routine	POKE	ON	OFF
Grenades	4405	48	16
Gun Fire	4045	48	16
Wall Collision	4087	1	0
Screen Flicker	5327	141	96

Turning off these routines has no effect on the game other than eliminating the specified sound. For example, turning off the wall noise does not turn off the screen flickering, nor does it stop the Nazi's from moving.

Strategies

Any game has its DOs and DON'Ts, and Castle Wolfenstein is no different. When you are on the first level (a single room with no doors, only a stairway), wait for the guards to be in a prime position before shooting. The guards will not attack you unless you move, attack them or they bump into you. When you leave a room, try to point the gun in a direction that gives the best chance of scoring a hit. This usually means pointing the gun in the direction you are moving. When entering a new room-leave it immediately. This allows you to think about the situation and ready your gun appropriately.

When to Kill

Try to kill the guards when they are next to a doorway. This stops the other guards and SS from getting to you. The Nazi's won't cross over fallen bodies. They can still fire at you, they just can't catch you. This can be used to create a safe place from which to throw grenades and such.

Shooting Through Walls

Another handy technique is to shoot through walls. For some reason Castle Wolfenstein will let you fire through corners. This allows you to shoot a guard and not risk being caught. One can also open up chests that are located in a corner. This saves time and avoids unnecessary risks.

Advanced Playing Techniques

The following techniques may be considered cheating by the less enlightened, but more open-minded individuals will readily see we are only taking advantage of the program and its limitations.

Life Beyond Death

Normally when the reset key is pressed Castle Wolfenstein saves the current game. You can change this so that instead of saving the game, you exit back to Applesoft. Once there, you can re-boot and resume the game one room back. There is a reset routine in both @INIT and @WOLF, but you are only concerned with the routine in @WOLF for the moment. I have listed the routine here so that it can be easily modified. A complete explanation of how the reset vector works can be found on pages 36 and 37 of the Apple][Reference Manual.

Figure 2

```
1187: A9 C7     LDA #$C7     Set the
1189: 8D F2 03  STA $03F2    RESET vector
118C: A9 1E     LDA #$1E     so that it
118E: 8D F3 03  STA $03F3    jmps to 1EC7
1191: 49 A5     EOR #$A5     Set the power
1193: 8D F4 03  STA $03F4    up byte
```

I wanted the Apple to re-boot the disk when I pressed the RESET key. To do this, type:

BLOAD @WOLF

Enter the monitor and type the following line:

1191:EA EA

Return to Applesoft (3D0G) and save @WOLF:

BSAVE @WOLF,A$810, L$16EB

NOTE: The ESC key (which saves the game) will still operate normally.

For The Aggressive Player

Castle Wolfenstein was written so that every room is stored on a unique sector. When the game first starts the track/sector list of CASTLE is read and stored in memory. The first sector contains the variables. Every time you enter a new room, the old room is saved and the new one is read in. This means that any room modifications you have made (grenades are handy for this) will be saved. It also allows you to go back one room if you just happen to make a fatal error. Also, if you have a disk editing program, such as DiskEdit, you can give yourself 255 bullets and grenades.

The Right Sector

First you have to find where this sector is. Read track $11, sector $C. This is the first sector of the catalog for Castle Wolfenstein. Look for the program ^BACKUP (some copies may have the name BACK-UP). If it is not on this sector, try sector $0B of the same track. If you still haven't found it, you are doing something wrong.

After locating the name, back up three bytes to find the Track/Sector list (how this information is stored is explained on pages 129-131 of the DOS Manual). Read this sector. Look at the thirteenth byte ($0C). The first number is the track where the first sector of the program is located, the next

byte is the sector. Read this track/sector. You now have the first sector of the program in memory ready to be modified.

I found the file name ^BACKUP on track $11, sector $0B. The third byte back from the name was $14 and the second byte was $0C, so I read track $14, sector $0C. After reading this sector, I looked at the 13th ($0C) byte. It was $20 and the next byte was $0B. This meant that the first sector of the program was located on track $20, sector $0B.

Some Custom Changes

Once the sector is in memory, move to the proper location and change the byte to the desired value (use the O command to move the cursor if you are using DISKEDIT). The table in Figure 3 shows the item, the location in the sector and the value to place there. All values are in hexadecimal.

Example: To get 255 bullets, move the prompt to location $47 and change the value there to $FF.

Figure 3

Desired Item	Byte	Value
Bullets	$47	0-$FF
Grenades	$48	0-$FF
Uniform	$49	$01
Bullet-Proof Vest	$4A	$01
War plans	$6C	$01
Rank	$6D	see text
Room #	$40	see text
Resurrection	$6F	$00
% Chance of Hit	$4B	0-$FF
% Chance of Recog.	$4D	0-$FF

NOTE: Change both ^BACKUP and ^CASTLE (also known as CASTLE) to be sure the modifications stick. These two files are used alternately at different levels of play.

Where Are You?

The map in Figure 4 shows the layout of the Castle. Each room has a number. This is the number to use if you need to change rooms.

NOTE: You may end up in a wall if you play with the room number. If this happens, you will have to try another room or change your position in the room. Bytes $43-$45 have something to do with your position within the room.

Your rank can be changed to a higher level which will cause the game to be much harder and more interesting. The values that correspond to the various ranks are as follows

$10 Private	$90 Captain
$30 Corporal	$A0 Colonel
$50 Sergeant	$C0 General
$70 Lieutenant	$E0 Field Marshall

Resurrection

If you happen to press RESET too late, your game can still be retrieved if you stop it before playing again. Put a $00 in byte $6F in the sector.

Giving It your Best Shot

Byte $4B determines the percent chance of your achieving a kill. The higher the number ($FF is greatest), the better your chances.

I'm Not Really Here

Byte $4D determines the percent chance of your being caught with $FF being the greatest percent chance of being recognized.

Some Minor Glitches

1) When you have more than 10 bullets, the display will still show you as having only ten bullets. This value will decrement once for each shot fired. Do not get bullets from a box. If you do, the program will replace the actual number of bullets you have with 10.

2) The grenade value appears as a letter or symbol that changes for each grenade thrown. The grenades do decrement by one for each thrown.

Neither of the above problems affects the play of the game, except to give you a lot of bullets and grenades.

Escaping Castle Wolfenstein

The path out of Castle Wolfenstein is always the same. The contents of each room are randomized for each new game. Once this map is memorized it becomes easier to escape the castle. Unfortunately, the plans are not guaranteed to be on the way out. My favorite tactic is to run for the exit, zapping or dodging as required and opening all chests I find along the way. If I haven't found the plans by the time I reach the last room, I backtrack and search until I find them.

Another APT

This APT program is an alternate method to changing the variables in Castle Wolfenstein.

Enter the program and save it to disk.

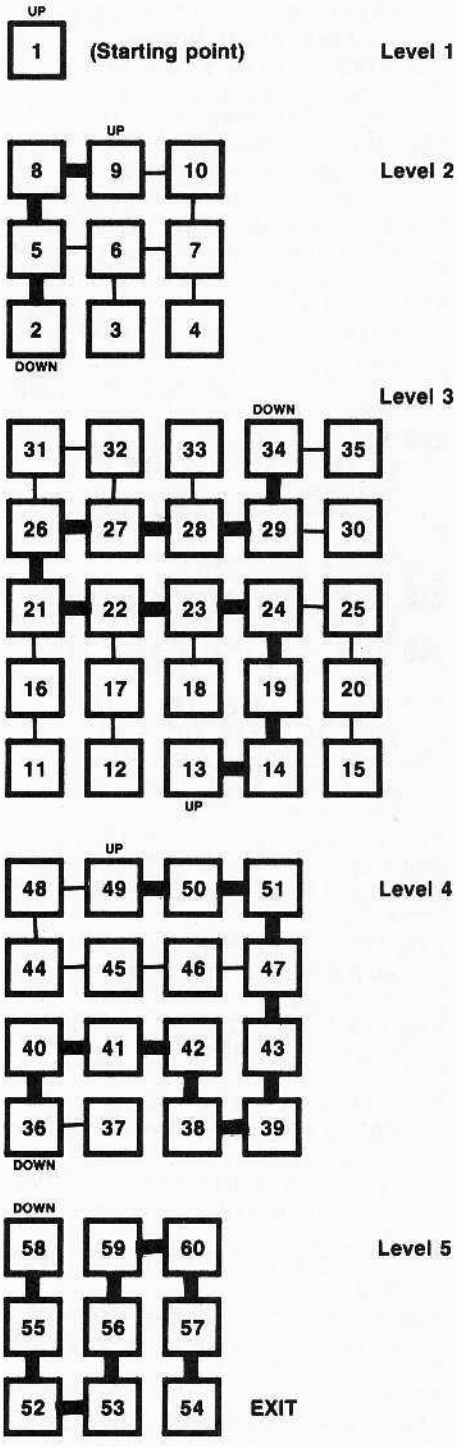
SAVE WOLFENSTEIN APT

Insert your unprotected Castle Wolfenstein disk and run the program. You will be asked for the file you wish to alter.

Remember: change both BACKUP and CASTLE in the same way to insure that the variables are indeed changed (or ^BACKUP and ^CASTLE if these are on your disk instead).

The default answer is Yes. ESC will return you to the main menu, or if pressed from the main menu will terminate the program. There is one additional choice provided, FIX BAD FILE. If when you are playing Castle Wolfenstein, the program freezes, try this option. It should restore the game and eliminate all SS who are actively pursuing you.

Figure 4



```
100 TEXT : HOME : NORMAL :
    HIMEM:16380
110 DATA 32,227,3,76,217,3
120 FOR X = 0 TO 5: READY: POKE
    768 + X,Y: NEXT
130 D$ = CHR$ (13) + CHR$ (4)
140 INVERSE : VTAB 2: HTAB 5:
    PRINT "A.P.T. FOR CASTLE
    WOLFENSTEIN": NORMAL
150 VTAB 7: HTAB 5: PRINT "1)
    CHANGE CASTLE": HTAB 5:
    PRINT "2) CHANGE BACKUP"
160 PRINT : HTAB 5: PRINT "3)
    CHANGE ^CASTLE": HTAB 5:
    PRINT "4) CHANGE ^BACKUP"
170 POKE 216,0: VTAB 13: HTAB
    10: PRINT "WHICH ONE (1-4) "
    CHR$(7);: GET A$
180 IF A$ = CHR$ (27) THEN HOME:
    PRINT "PROGRAM TERMINATED":
    END
190 IF A$ < "1" OR A$ > "4" THEN
    140
200 FI$ = "CASTLE": IF A$ = "2"
    OR A$ = "4" THEN FI$ =
    "BACKUP"
210 IF A$ > "2" THEN FI$ = "^" +
    FI$
220 ONERR GOTO 170
230 PRINT D$"VERIFY"FI$: POKE
    216,0: GOSUB 810
240 B1 = PEEK (TB) + PEEK (TB +
    1) * 256
250 POKE TR, PEEK (B1 + 12):
    POKE SE, PEEK (B1 + 13)
260 POKE CMND,1: POKE BU,0:
    POKE BU + 1,64: POKE VOL,0:
    CALL 768
270 IF PEEK (ERR) > 15 THEN 740
280 FI$ = "MAXIMUM GRENADES":
    GOSUB 920: IF B THEN POKE DB
    + 72, 255
290 FI$ = "MAXIMUM BULLETS":
    GOSUB 920: IF B THEN POKE DB
    + 71, 255
300 FI$ = "A UNIFORM": GOSUB
    920: IF B THEN POKE DB +
    73,1
310 FI$ = "A BULLET PROOF VEST":
    GOSUB 920: IF B THEN POKE DB
    + 74, 1
320 FI$ = "THE WAR PLANS": GOSUB
    920: IF B THEN POKE DB +
    108, 1
330 IF PEEK (DB + 111) = 0 THEN
    350
340 FI$ = "TO BE RESURECTED":
    GOSUB 920: IF B THEN POKE DB
    + 111,0
350 HOME
360 FI$ = "TO CHANGE YOUR RANK":
    GOSUB 920: IF NOT B THEN 450
370 PRINT : PRINT "CURRENT RANK
    IS "; INT (( PEEK (DB + 109)
    / 16 + 1) / 2)
380 PRINT : POKE WL,5: PRINT :
    PRINT "1) PRIVATE": PRINT
    "2) CORPORAL"
390 PRINT "3) SERGEANT": PRINT
    "4) LIEUTENANT": PRINT "5)
    CAPTAIN"
400 PRINT "6) COLONEL": PRINT
    "7) GENERAL": PRINT "8)
    FIELD MARSHAL"
410 PRINT : PRINT "WHICH ONE
    (1-8) ";: GET A$
420 POKE WL,0:A = VAL (A$)
430 IF A < 1 OR A > 8 THEN PRINT
    : PRINT : PRINT "MAINTAINING
    OLD RANK": FOR X = 1 TO 500 :
    NEXT : GOTO 450
440 POKE DB + 109,(2 * A - 1) *
    16
450 HOME
460 FI$ = "TO CHANGE ROOMS":
    GOSUB 920: IF NOT B THEN 510
470 PRINT : PRINT "CURRENTLY IN
    ROOM "; PEEK (DB + 64)
480 PRINT : INPUT "ENTER ROOM
    NUMBER (1-60) ";A$
490 B = VAL (A$): IF B < 1 OR B
    > 60 THEN PRINT : PRINT
    "MAINTAINING OLD ROOM": FOR
    X = 1 TO 500: NEXT : GOTO 510
500 POKE DB + 64,B
510 HOME
520 FI$ = "TO CHANGE PERCENT
    CHANCE OF ACHIEVING A HIT":
    GOSUB 920
530 IF NOT B THEN 600
540 PRINT : PRINT "CURRENT
    CHANCE IS ";
550 PRINT INT ( PEEK (DB + 75) /
    255 * 100);"%"
560 PRINT : INPUT "ENTER PER-
    CENT WANTED ";A$
570 IF A$ = "" THEN 600
580 B = VAL (A$): IF B < 0 OR B
    > 100 THEN 600
590 POKE DB + 75,255 * B / 100
600 HOME
610 FI$ = "TO CHANGE PERCENT
    CHANCE OF BEING RECOGNIZED":
    GOSUB 920:
620 IF NOT B THEN 680
630 PRINT : PRINT "CURRENT
    CHANCE IS "; INT ( PEEK (DB
    + 77) / 2.55)"%"
640 PRINT : INPUT "CHANGE PER-
    CENT CHANCE TO ";A$
650 IF A$ = "" THEN 680
660 B = VAL (A$): IF B < 0 OR B
    > 100 THEN 680
670 POKE DB + 77,255 * B / 100
680 HOME
690 FI$ = "TO FIX A BAD FILE":
    GOSUB 920: IF NOT B THEN 710
700 FOR X = 110 TO 256: POKE DB
    + X,0: NEXT : FOR X = 76 TO
    106: POKE DB + X,0: NEXT
710 HOME : VTAB 12: HTAB 16:
    PRINT "WRITTING"
720 POKE CMND,2: POKE VOL,0:
    CALL 768: HOME
730 IF PEEK (ERR) < 16 THEN 140
740 PRINT CHR$ (7)"WARNING, DOS
    ERROR"
750 E = PEEK (ERR)
760 IF E = 16 THEN PRINT "WRITE
    PROTECTED (REMOVE TAB)
770 IF E = 64 THEN PRINT "DRIVE
    ERROR (I/O)"
780 IF E < > 10 AND E < > 40
    THEN PRINT "UNUSUAL ERROR,
    CODE = ";E
790 TEXT
800 END
810 FT = 46582:SL = 46583:DR =
    46 584
820 TR = 47084:SE = 47085
830 NS = 46574:TB = 46537
840 WL = 32:WW = 33:WT = 34
850 CMND = 47092:ERR = 47093
860 VOL = 47083
870 BU = 47088
880 DB = 16384
890 HOME : INVERSE : PRINT "FILE
    NAME:";
900 NORMAL : PRINT " ";FI$
910 POKE WT,5: VTAB 6: POKE ERR,
    0: RETURN
920 B = 0: PRINT "DO YOU WANT
    ";: POKE WL,12: INVERSE :
    PRINT FI$;: NORMAL : PRINT "
    (Y/N) ";: GET A$
930 POKE WL,0
940 IF A$ = CHR$ (27) THEN POP :
    GOTO 140
950 IF A$ = CHR$ (13) THEN PRINT
    : RETURN
960 PRINT A$
970 IF A$ = "Y" THEN B = 1
980 RETURN
```

Checksums

```
100 - $86D4     400 - $96DE     700 - $F005
110 - $B0A7     410 - $7721     710 - $3198
120 - $4FB5     420 - $4853     720 - $99A5
130 - $65DF     430 - $A1F2     730 - $C804
140 - $EFC0     440 - $0089     740 - $C9F9
150 - $E6D6     450 - $30BD     750 - $33E5
160 - $D6B4     460 - $1402     760 - $9073
170 - $B194     470 - $3B76     770 - $2034
180 - $F6AD     480 - $C499     780 - $AEB0
190 - $A6A3     490 - $F4D1     790 - $D616
200 - $66AA                     800 - $F9B1
210 - $3871     500 - $3BFF     810 - $EF5B
220 - $09BE     510 - $B6E0     820 - $7E7C
230 - $D16D     520 - $480B     830 - $FBB9
240 - $F4DD     530 - $72D9     840 - $0407
250 - $8355     540 - $5646     850 - $A57E
260 - $EED0     550 - $83BB     860 - $6492
270 - $7B5F     560 - $739C     870 - $0B86
280 - $3D07     570 - $48B4     880 - $6D65
290 - $A3BC     580 - $4083     890 - $C419
                590 - $94A8
300 - $1117     600 - $5A41     900 - $81D7
310 - $BB94     610 - $6B6F     910 - $C5D0
320 - $6CD5     620 - $FA0D     920 - $2804
330 - $283D     630 - $6715     930 - $EBA5
340 - $EAE5     640 - $659F     940 - $0A05
350 - $CC51     650 - $32F3     950 - $A10C
360 - $66B5     660 - $667C     960 - $2D42
370 - $6F01     670 - $14DE     970 - $8B3E
380 - $D7D6     680 - $34B5     980 - $7842
390 - $53DB     690 - $D346
```

Learn to use and understand $trings with

Text Invaders

By Bev Haight

Applesoft string manipulation at first seems a bit complex; however, with experience, it becomes perfectly logical and simple. If you're in between these two extremes (total ignorance and total know-how), then you understand that concatenation sometimes is a chore, albeit a necessary chore, especially when writing business and educational software.

This game, Text Invaders, depends on string manipulation. Certain characteristics of the arcade version have been duplicated in this text-page pseudo-clone, including the "Thump! Thump!" of marching invaders as they descend upon you. The invaders also "march" back and forth, their "legs" alternately slashes and inequality signs "/ \ < >". All this animation is accomplished by tabbing and printing strings.

The variable names were chosen to be as explanatory as possible. Since Applesoft only uses the first two letters of a variable name, the names may be shortened to just two characters. However, be certain to include the variable suffix (%, $ or array notation).

Text Invaders manipulates several strings. The most complex of these is called INVADER$(1). It is with this string that we will see the use of all the string-manipulating commands: STR$[n], VAL [a], LEN[A$], MID$[H$,b,1], RIGHT$[A$,b], LEFT$[A$,b] and, of course, concatenation.

The program will be explained in five steps.

Step 1. Introduction, Graphics and Variables

Step 2. The Rhythm, March and Descent of the Invaders

Step 3. Tank Commands and Motion

Step 4. The Invaders Attack

Step 5. The Tank Strikes Back

Step 1: Introduction & Initial Graphics

This will set up the initial graphics and let the player select a skill level.

The introduction, or "skill choice selector," is an example of an active user-proof (except for ctrl C and reset) keyboard entry program. It will accept only what it is looking for, flash an error message when applicable and, while waiting for an entry, display the choices available in an interesting fashion. To do this, it does not use INPUT or GET statements, which halt program execution, but rather PEEKs the keyboard (-16384) to see if a selection has been made. If no selection has been made, it goes about merrily flashing the next choice and PEEKing the keyboard until an entry has been made. If the entry is not valid, an error message is flashed one letter and one buzz at a time, yet quickly enough for the entire message to be printed and removed in one second. It does this by using the MID$ function and a FOR-NEXT loop.

The program sequence in Step 1 is:

A. Define variables (10000-10910).

B. Text Page Graphics are printed: The Screen is whited out and inverse text is printed [11000-11030], followed by normal text on the top and bottom [11050-11070]. A "window" is made (POKEs 32 through 35, see the Applesoft Reference Manual) and cleared (CALL-958) in lines 11100-11120. Flashing "bunkers" or barriers are printed, consisting of regular slashes and back slashes (11200).

C. The player's skill level is requested (12000-12090). While waiting for the answer, lines 12050-12070 flash and buzz across the printed choices, prompting the player to choose. If the player chooses anything other than numbers 1 through 5, the error routine is activated, printing and buzzing quickly across the top part of the graphics field and vanishing (12080-12090).

D. The window is cleared (line by line) of the skill prompts, leaving the flashing bunkers intact (12100), and the proper number of invaders are created (13000-13040) and printed in the window (12210).

String concatenation appears first in line 10150 (BUNKER$), and then in line 10900 where SPACE$ is created out of spaces.

In lines 13020 and 13030 the INVADER strings are created. To change the appearance of the invaders, you need only change these lines. For example, if you want the invaders to have different legs, change line 13030. If you want their hit points to be less than 9, change I$ to 8 or 5 or some other number. WARNING: Do not make I$ larger than a single character or the program will not work! I$ must not be greater than 9.

Step 2: Rhythm, March and Descent

Now that the preliminaries are out of the way, it's time to get down to a serious invasion.

Step 2 consists of all the invasion controls: the "thump! thump!" of their march back and forth, down the screen and toward and through the "bunkers."

The invasion sequence, repeated over and over, goes like this:

1. Select the gait of the invaders' march (10100).

2. Select the invaders' direction of travel (ITRAVEL%, 20100).

3. Determine the HTAB and VTAB value (HINVADER%,VINVADER%) and clear old invader strings from the window (going right 24000-24200, or going left 25000-25200).

4. VTAB, HTAB and PRINT the invaders (24100-24300), FRHYTHM is used as a flip-flop switch to select which legs to print (29500). FRHYTHM is either 1 or -1. Adding 1 to each value results in either 0 or 2; hence the last part of line 24200: PRINT INVADER$ (FRHYTHM% + 1). Remember: INVADER$(0) is slashes and INVADER$(2) is inequality signs.

5. The thumps are then added, depending again upon which type of leg is printed (FRHYTHM%, 15000-15200). The two types of thumps are created by PEEKing -16336 twice times the quantity 16 less the VTAB of the INVADER$(1). The only difference between the two thumps is the interval created by the addition or subtraction process.

6. The whole process repeats itself. If the invaders are too far right, they must march left (24000/2), and if too far left, they must go right (25100). If too low, then the game must end (10000) and the player is given the choice of beginning again or quitting (21000-21090).

Step 3: March of the Text Invaders

This is a very short step that gives command of the tank.

1. The keyboard is checked (10200)(10000) to see if:

a. The escape key is pressed, which means to start all over again (12000).

b. The return key is pressed, which stops all tank motion.

c. The right arrow key is pressed, to go right.

d. The left arrow key is pressed, to go left

2. Move the tank either right (20000) or left (30000) or make it stand still (35000) by erasing the old tank and printing a new one (21000-23000).

3. Display the tank's hit power. This is

a novel routine that uses the data statements and then VTAB/HTABs each character into a rectangular area that measures five columns by five rows. The result is a number from 9 to 1 printed in "*" or "#" alternately (400-450).

Step 4: The Invaders Attack

The sequence here is more difficult because it consists of numerous subroutines. The most important of these is the Attack Sequence itself which will be explained first.

A. The attack itself begins (900) with a random number between 1 and the number of original invaders, N% (between 4 and 8). This number times 4 plus 3 gives the "centers" of each invader in INVADER$(1). The old invaders are erased and the random number, RAN%, is checked for validity.

B. That particular "center" is pulled out of INVADER$(1).

915 TEMP$ = MID$ (INVADER$ (1), RAN% + 1,1)

C. And turned into a number.

920 TEMP% = VAL (TEMP$)

D. The number is decremented by one.

940 TEMP% = TEMP% - 1

E. Then turned back into a string.

TEMP$ = STR$ (TEMP%)

F. And put right back into INVADER$ (1).

970 INVADER$(1) = LEFT$ (INVADER$ (1), RAN%) + TEMP$ + RIGHT$ (INVADER$ (1), LEN (INVADER$(1)) - RAN% - 1

G. If an invader is already dead, then this step will get rid of its legs in almost the same way (980). Lack of hit power (and therefore no more bombs to drop) is another reason for an invader's demise and disappearance. Later, in Step 5, this same routine is used to decrement the invaders when they're hit by missiles.

H. Condense the invaders by removing all spaces on the right side (660) and the left side (670), keeping the string compact and changing the HTAB value appropriately.

The next phase in Step 4 is the sequence followed when the screen is examined in order to determine if the invaders' bombs have hit anything.

The most important part here is, of course, the actual routine that examines the text screen memory. This is accomplished by using the SCRN (x,y) command (see the Applesoft Manual), which normally returns the color code (0-15) of the x,y coordinates (between 0 and 39 for x, 0 and 47 for y . However, since each text character is composed of two such color codes, a formula must be used that will return the text character instead of the colors. The formula given in the Applesoft Manual is:

CHR$ (SCRN (x - 1, 2 * (y - 1)) + 16 * SCRN (x - 1, 2 * (y - 1) + 1))

which will return the character at position (x,y).

In this program a window has been used, which causes this formula to give an incorrect value, it is looking at the wrong spot on the screen. Therefore, the x value must be incremented by 1 (line 510) before passing the value to this subroutine (lines 10-20). Since the keyboard values are PEEKed, the CHR$ function has been removed. Here is what the program does:

A. Check the screen, has the bomb hit anything?

1. Empty space... continue on... line 520.
2. A "!"... hit a missile... line 530.
3. Flashing slashes... hit the bunkers... line 540.
4. Oh no, too low! Hit the dirt!... line 550.
5. Must have hit a tank!... lines 570-580.

B. Destroy whatever the bomb has hit.

1. The tank goes boom (13200-13290)! This is the most complicated destruction scene except for the way the invaders will die (more on that in Step 5).
2. The bunker goes zap (850-890)! This scene simply buzzes while flashing from inverse to normal a few times.
3. The bomb hits the ground! This does a short buzz and leaves a crater after the screen flashes back and forth between text page 1 and text page 2.
4. Your missile gets hit! The two vanish in a noisy incandescence of slashes, letter "I's and dashes.

Step 5: The Tank Strikes Back

Like the search routine used to find out if the bombs hit anything, the missile search routine also uses the SCRN function in lines 10 and 20.

Did you shoot (line 150)? And if so, did you hit anything? (Line 1050 takes you to lines 50 through 90.)

1. Hit nothing, so go on (line 55).
2. Hit your own bunker (line 60).
3. Hit a bomb (line 65).
4. Hit an invader
   a. on the left side (75), so destroy it from the left side (610).
   b. on the right side (80), so destroy it from the right side (620).
   c. in the middle (85), so destroy it from the middle (630).
5. Hit the top of the playing area (line 90).

That's it. Debug and enjoy!

IMPROVEMENTS

An Applesoft compiler was used to make the program really zip along. And it did...too quickly. The invaders' descent should be slowed down. I made them randomly choose to move or stand still but still wiggle those skinny legs of theirs; that way you don't quite know which way they're going to go.

Invaders BASIC listing

```
9 GOTO 10000
10 XX% = X% - 1:YY% = 2 * (Y% - 1 )
20 XK% = SCRN( XX%,YY%) + 16 * SCRN(
    XX%,YY% + 1): RETURN
30 VTAB TV%: HTAB TH%: PRINT " " ;:
    RETURN
35 VTAB TV%: HTAB TH%: PRINT "!" ;:
    RETURN
40 IF TV% > 0 THEN GOSUB 90: HTAB
    TH%: PRINT " ";
45 TH% = HG%:TV% = VG% - 2: GOSUB
    35: RETURN
50 IF TV% < 1 THEN RETURN
52 GOSUB 30:TV% = TV% - 1: GOSUB 35:
    IF TV% < 5 THEN GOTO 90
55 X% = TH% + 1:Y% = TV% - 1: GOSUB
    10: IF XK% = 160 THEN RETURN
60 IF XK% < 127 THEN FOR A = 1 TO 5:
    GOSUB 30: GOSUB 1530: GOSUB 35:
    GOSUB 1530: GOSUB 30: GOSUB
    1530: NEXT A:TV% = TV% - 1:
    GOSUB 30:TV% = 0: RETURN
65 IF XK% = 171 THEN 800
70 IF TV% < > VI% + 1 THEN RETURN
72 SS% = 1
75 IF XK% = 221 THEN GOSUB 600: RA%
    = X% - HI%: GOTO 910
80 IF XK% = 219 THEN GOSUB 620: RA%
    = X% - HI% - 2: GOTO 910
85 GOSUB 610:RA% = X% - HI% - 1:
    GOTO 910
90 VTAB TV%: HTAB TH%: PRINT " " ;:
    VTAB TV% - 1: HTAB TH%: PRINT
    "$";: FOR A = 1 TO 7:BU = PEEK
    (NO): NEXT A: HTAB TH%: PRINT
    ".";:TV% = 0: RETURN
100 KE% = PEEK (KE): IF KE% > 12 7
    THEN POKE ST,0
110 IF DI% AND FS% < 0 THEN KE% = 160
120 IF KE% = 155 THEN GO% = 1: GOTO
    10000
150 IF KE% = 160 THEN AU% = 1: GOSUB
    40
155 IF KE% = 141 THEN TT% = 3
160 IF KE% = 149 THEN TT% = RI%: DI%
    = 0: GOTO 200
170 IF KE% = 136 THEN TT% = LE%: DI%
    = 0: GOTO 300
190 ON TT% GOTO 200,300,350
200 HG% = HG% + 1: IF HG% > 36 THEN
    HG% = 36:TT% = LE%: RETURN
210 VTAB VG%: HTAB 1: CALL - 86 8:
    VTAB VG% - 1: CALL - 868
220 VTAB VG%: HTAB HG% - 1: PRINT
    "=";: INVERSE : PRINT PO%;: NOR-
    MAL : PRINT "=";
230 VTAB VG% - 1: HTAB HG%: PRINT
    "I";: RETURN
300 HG% = HG% - 1: IF HG% < 3 THEN
    HG% = 3:TT% = RI%: RETURN
350 GOTO 210
400 B% = 1: FOR A = 19 TO 23: FOR AA
    = 1 TO 5
410 VTAB A: HTAB AA + 5
420 PI$ = MID$ (PO$(PO%),B%,1)
430 INVERSE
450 PRINT PI$;: NORMAL :B% = B% + 1:
    NEXT AA,A: RETURN
500 IF IH% = 0 THEN 900
510 IV% = IV% + 1:X% = IH% + 1:Y%
    =IV%: GOSUB 10
520 IF XK% = 160 THEN 700
530 IF XK% = 161 THEN 800
540 IF XK% = 92 OR XK% = 111 THEN 850
550 IF IV% > 17 THEN 750
```

```
570 INVERSE : GOSUB 13200:PO% = PO%
    - 1: IF PO% = 0 THEN 21000
580 VTAB VG% - 2: HTAB IH%:IH% = 0:
    PRINT " ";
590 GOSUB 400: RETURN
600 FOR A = 0 TO 2: VTAB VI% + 1 :
    HTAB X% - 1: GOSUB 630: VTAB
    VI%: HTAB X% - 1: GOSUB 630:
    HTAB X%: GOSUB 630: HTAB X% + 1:
    GOSUB 630: VTAB VI% + 1: HTAB X%
    + 1: GOSUB 630: HTAB X%: GOSUB
    630: NEXT
605 BP% = 0: GOSUB 1600: GOTO 640
610 FOR A = 0 TO 2: VTAB VI%: HTAB X%
    - 1: GOSUB 630: HTAB X% - 2:
    GOSUB 630: HTAB X%: GOSUB 630:
    VTAB VI% + 1: HTAB X% - 2: GOSUB
    630: HTAB X%: GOSUB 630: HTAB X%
    - 1: GOSUB 630: NEXT
615 BP% = 10: GOSUB 1600: GOTO 64 0
620 FOR A = 0 TO 2: HTAB VI% + 1 :
    HTAB X% - 1: GOSUB 630: VTAB
    VI%: HTAB X% - 1: GOSUB 630:
    HTAB X% - 2: GOSUB 630: HTAB X%
    - 3: GOSUB 630: VTAB VI% + 1:
    HTAB X% - 3: GOSUB 630: HTAB X%
    - 2: GOSUB 630: NEXT
625 BP% = 0: GOSUB 1600: GOTO 640
630 PRINT ZA$(A);:X = PEEK (NO) +
    PEEK (NO): RETURN
640 TH% = 0:TV% = 0: GOTO 970
650 REM
660 IF MID$ (IN$(1), LEN (IN$(1 )) -
    2,1) = " " THEN FOR AA = 0 TO
    2:IN$(AA) = LEFT$ ( IN$(AA), LEN
    (IN$(AA)) - 4): NEXT AA: IF LEN
    (IN$(1)) > 4 THEN GOTO 660
670 IF MID$ (IN$(1),4,1) = " " THEN
    FOR AA = 0 TO 2:IN$(AA) = " " +
    RIGHT$ (IN$(AA), LEN ( IN$(AA))
    - 5): NEXT AA:HI% = HI% + 4: IF
    LEN (IN$(1)) > 4 THEN 670
690 RETURN
700 VTAB IV%: HTAB IH%: PRINT "+";:
    VTAB IV% - 1: HTAB IH%: PRINT "
    ";: RETURN
750 VTAB IV% - 1: HTAB IH%: PRINT "
    ";: FOR T = 1 TO 2
770 POKE - 16299,0: INVERSE : VTAB
    IV%: HTAB IH%: PRINT "^";: GOSUB
    1530: POKE - 16300,0
780 NORMAL : HTAB IH%: PRINT CHR$
    (223);: GOSUB 1530
790 NEXT T:IH% = 0: RETURN
800 FOR BO = 1 TO 10:ZZ = PEEK (NO)
810 VTAB IV% - 1: HTAB IH% - 1:
    PRINT CHR$ (220);"I";"/";: VTAB
    IV%: HTAB IH% - 2: PRINT
    "-=#=-";: VTAB IV% + 1: HTAB IH%
    - 1: PRINT "/I"; CHR$ (220) ;:ZZ
    = PEEK (NO)
820 VTAB IV% - 1: HTAB IH% - 1:
    PRINT "";: VTAB IV%: HTAB IH% -
    2: PRINT "";: VTAB IV% + 1: HTAB
    IH% - 1: PRINT "" ;: NEXT BO
830 FOR A = 1 TO 10:ZZ = PEEK ( NO):
    VTAB TV%: HTAB TH%: PRINT "#";:
    FOR B = 1 TO 5: NEXT B : HTAB
    TH%: PRINT " ";: NEXT A
840 TV% = 0:TH% = 0:IV% = 0:IH% = 0:
    RETURN
850 FOR T = 1 TO 6: VTAB IV% - 1 :
    HTAB IH%: INVERSE : PRINT " ";:
    VTAB IV%: HTAB IH%: PRINT "%";:
    GOSUB 1530
860 VTAB IV% - 1: HTAB IH%: NORMAL :
    PRINT " ";: VTAB IV%: HTAB IH%:
    PRINT " ";: GOSUB 1530: NEXT
880 IH% = 0
890 RETURN
900 R% = RND (1) * N%:RA% = R% * 4 +
    3: IF LEN (IN$(1)) < 5 THEN VTAB
    VI%: HTAB 1: CALL - 868: VTAB
    VI% + 1: CALL - 868: RETURN
905 IF RA% > LEN (IN$(1)) THEN
    RETURN
910 IF RA% < 0 THEN RA% = - RA%
915 TE$ = MID$ (IN$(1),RA% + 1,1 )
920 TE% = VAL (TE$): IF TE$ = " "
    THEN 900
930 IF TE% > 9 THEN TE% = TE% / 10:
    GOTO 930
935 IF SS% THEN IF TE% < = PO% THEN
    TE% = 1
940 TE% = TE% - 1:QI% = IV% + 1:T E$
    = STR$ (TE%):QH% = RA% + HI%: IF
    TE% < 1 THEN TE$ = " "
945 IF SS% THEN 960
950 IV% = VI% + 2:IH% = RA% + HI% :
    GOSUB 700: VTAB VI%: HTAB IH%:
    INVERSE : PRINT TE$;: NORMAL
960 SS% = 0
970 IN$(1) = LEFT$ (IN$(1),RA%) +
    TE$ + MID$ (IN$(1),RA% + 2)
980 IF TE$ = " " THEN FOR AA = 0 TO
    2:IN$(AA) = LEFT$ (IN$ (AA),RA%
    - 1) + "" + MID$ (IN$(AA),RA% +
    3): NEXT AA
990 GOTO 650
1000 IF VI% = 14 THEN 21000
1010 FOR RH = 0 TO 100 STEP VI% * 10
1020 GOSUB 100: REM <CHECK KEYBOARD>
1040 GOSUB 500: REM <MOVE INVADER
    BOMBS>
1050 IF AU% THEN GOSUB 50
1080 NEXT RH
1090 GOTO 2000
1500 FOR A = 0 TO 16 - VI%: IF FR% >
    0 THEN 1520
1510 ZZ = PEEK (NO) - PEEK (NO) :
    NEXT : RETURN
1520 ZZ = PEEK (NO) + PEEK (NO) :
    NEXT : RETURN
1530 ZZ = PEEK (NO): RETURN
1600 BP% = BP% * SK% + PO% + PT%
1610 VTAB 19: HTAB 30: INVERSE :
    PRINT "";: HTAB 30: PRINT PTS%;:
    NORMAL : RETURN
1999 REM <INVADER MOVEMENT>
2000 IF LEN (IN$(1)) < 5 THEN 20000
2010 IF IT% = LE% THEN 2500
2400 HI% = HI% + 1: IF HI% > 37 - LEN
    (IN$(1)) THEN IT% = LEF T%: VTAB
    VI%: HTAB 1: CALL - 868: IF SK%
    < 3 THEN VI% = V I% + 1
2410 VTAB VI%: HTAB 1: CALL - 868:
    VTAB VI% + 1: HTAB 1: CALL - 868
2420 VTAB VI%: HTAB HI%: PRINT I
    N$(1): VTAB VI% + 1: HTAB HI %:
    PRINT IN$(FR% + 1);
2430 FR% = - FR%: GOSUB 1500: GOTO
    1000
2500 HI% = HI% - 1
2510 IF HI% < 2 THEN VTAB VI%: HTAB
    1: CALL - 868:IT% = RI%:VI% =
    VI% + 1
2520 GOTO 2410
10000 SPEED= 255: NOTRACE : NORMAL :
    TEXT : HOME
10100 NO = - 16336:ST = - 16368 :KEY
    = - 16384
10110 HI% = 3:VI% = 5:IT% = 1:I$ =
    "9"
10120 RI% = 1:LE% = 2:TR% = 2:FR% =
    - 1
10130 PO% = 9:TT% = 2
10140 HG% = 19:VG% = 17:DI% = 1:A G%
    = 1
10150 BU$ = CHR$ (220) + CHR$ ( 239)
    + CHR$ (220) + CHR$ ( 239)
10160 IVRAY% = 0:IHRAY% = 0:UFO% = 1
10170 TH% = 2
10180 ZA$(0) = "*":ZA$(1) = ":":Z
    A$(2) = " "
10500 DATA" ### ## #### #### ## ### "
10510 DATA"  ##   ##   ##   ## ## "
10520 DATA"####    ## ### ##   #####"
10530 DATA"####    ## ###    ###### "
10540DATA  "** **** *******   **   **"
10550DATA  "*******   ****    ****** "
10560 DATA"##   ##   #### ## ## ### "
10570 DATA"#####   ##  ##  ##   ##  "
10580 DATA" ### ## ## ### ## ## ### "
10590 DATA" ### ## ## ####   ##   ##"
10600 IF GO% THEN 11000
10800 ERR$ = "CHOOSE A NUMBER FROM 1
    TO 5"
10900 FOR A = 1 TO 40:SPACE$ =
    SPACE$ + " ": NEXT A
10910 FOR A = 0 TO 9: READ PO$(A ):
    NEXT A
11000 INVERSE : FOR A = 3 TO 24:
    VTAB A: HTAB 1: PRINT SP$;: NEXT
    A
11010 VTAB 19: HTAB 24: PRINT
    "SCORE:"
11020 VTAB 19: HTAB 2: PRINT "TANK":
    PRINT "VALUE": HTAB 4: PRINT
    "IS": HTAB 3: PRINT "-=>";
11030 VTAB 22: HTAB 14: PRINT
    "SOFTKEY TEXT INVADERS 2.0";
11050 NORMAL
11060 VTAB 24: HTAB 2: PRINT
"<- LEFT";: HTAB 15: PRINT "RETURN =
    STOP";: HTAB 33: PRINT
    "RIGHT->";
11070 VTAB 1: HTAB 2: PRINT "SPACE =
    SHOOT!";: HTAB 26: PRINT "ESC =
    NEW GAME";
11100 POKE 32,1: POKE 33,38: POKE
    34,3: POKE 35,16
11110 VTAB 3: HTAB 1: CALL - 95 8
11120 VTAB 3: HTAB 1: INVERSE :
    PRINT RIGHT$ (SP$,38);
11200 FLASH : FOR A = 12 TO 14: VTAB
    A: HTAB 4: PRINT BU$;: HTAB 13:
    PRINT BU$;: HTAB 22: PRINT BU$;:
    HTAB 31: PRINT BU$;: NEXT A:
    NORMAL
11210 GOSUB 400
11220 GOSUB 210
11230 GO% = 0
12000 VTAB 5: HTAB 7: PRINT "WHAT IS
    YOUR SKILL LEVEL?";: VTAB 7:
    HTAB 9: PRINT "1....2....
    3....4....5";: VTAB 9: HTAB 14:
    PRINT "< PICK ONE >";
```

—continued on page 61—

Zyphyr Wars

By Bev Haight

Zyphyr Wars 2.0 is an Applesoft hi-res "shoot the invaders" type of arcade game. This particular one is not only a complete game in itself, but it is also a "core" program that can be easily altered and adapted to demonstrate the ease with which such games are created. (It is an egotistical myth that the writing of computer games is a difficult art!) In addition, this game has several tricks that allow it, and perhaps others, to play more quickly.

Playing the game

A city is built at the bottom of the screen, and the player's ship, a satellite, is drawn at the top. Ten UFOs, or Zyphyrs (Zs) appear between the ship and the city skyline below. The Zs must be destroyed before they destroy the ship and the city. If the Zs succeed in creating deep craters, the game ends. The Zs zap the city and the ship with death rays, and the ship shoots back with rays of its own. But because the player's ship is shooting down at the Zs, a miss will destroy part of the city, and if there are too many misses, the game ends quite dramatically.

When the first ten Zs are destroyed, the player advances to the next skill level and another ten Zs appear. Each higher level will subject the player's ship to more frequent attacks. If the player's ship is hit, it plummets to the ground, destroying whatever is beneath it. There are only a few ships provided to the player.

Special features

This program is a fast single-player game with some interesting features. First, instead of using a hi-res text/character generator (although it could be adapted to use one), this program flashes the hit points by switching screens (text and hi-res) in rapid, repetitive bursts that give the illusion of text on the hi-res page. Second, the Zs, appear to zip across the screen much faster than is really possible. The illusion of speed is created by connecting their old and new positions with a line and then adding a "zip" sound ("Zyphyr" is a close approximation of this zipping sound). Third, there is an entire cityscape to protect rather than simple bunkers to hide behind. And, finally, this game uses shape tables. That is, the buildings consist of "shapes" stacked up in columns. The Zs are also conglomerate shapes (allowing the programmer to change their shapes). But the players's ship is a single shape. The shapes are POKEd into memory page 3 ($300 hex or 768 in decimal) along with a short routine. DRAW and XDRAW are used extensively, along with HPLOTs, in order to demonstrate the various hi-res graphic possibilities. (For example, all HPLOTs could be turned into XDRAWs.)

Paddles Only

When the ship is moved back and forth, it does not have to step across the screen smoothly. Using absolute paddle positions, the ship "materializes" at whatever value the paddle happens to be turned to when its value is checked by the program. That means that the player can also make the ship zip from one side to the other, a necessary feature if the player wishes to zap the Zs.

Rays, Not Bombs

Don't move the missiles or bombs because that slows everything down. Instead, HPLOT or XDRAW lines (rays), giving the appearance of an immediate hit or miss.

Hit or Miss?

The program does not need to go through a long search to determine if a ray has hit or missed. The program only permits the Zs to appear directly over the center of each column of buildings, and then only one Z per building. The player's ship moves across in a similar fashion, appearing only over the center of each building. Therefore, the only necessary search is for a Z in the same column the player's ship is in.

Sound Effects

There are a variety of noises used in this game, including a soft "zip: sound. Another favorite sound routine may be used instead.

The Program

The 40 columns (representing the 40-character-column text format) must be reserved by dimensioning them at the very start.

COL%(40) This integer variable array will store which Z is above which building (column). If empty, it will be equal to zero.

HEIGHT(40) This array will store the height of the building. When struck by a ray or a falling ship, the building will diminish in height until a crater is dug into the ground.

XUFO%(10), YUFO%(10) These arrays store the X and Y positions of all ten Zs.

To keep track of the player's ship, there are two variables:

PN%, the New Paddle position,
PO%, the Old Paddle position.

Hi-Res Text Trickery

After the variables are defined, the game itself is set up in lines 12000 through 12090. The Earth is HPLOTted (EARTH = 152) on hi-res page 1 (13000-13030). Then, on text page 1 (13000-13130), it sets the bottom of the text window to 19 (13180). That means that the four rows seen below the hi-res screen in the mixed-mode are actually outside of the window. This will pose a small problem later when you try to print below the window. The ground is continued into the text page. This gives the illusion of having text on the hi-res page, an effect that will later be enhanced.

The city shapes up

Next, the buildings are calculated and XDRAWn (14000-14090). Only the inner 38 columns are used for buildings (14000). A random height is chosen and subtracted from EARTH. (Remember that as the height of the building increases, the actual Y value gets LOWER, not higher because the Y value at the very top of the screen is zero!)

Now, one of the five building shapes from the shape table is chosen at random (14040). The five shapes in the table show a window (or windows) at various positions. When stacked together (14060), something resembling a modern building is created. Shape #6 (see diagram) is a plain and simple vector dash, but at SCALE=7 it is a solid line (14050). This line separates the building levels.

Stars in four colors

After the buildings are up, stars are drawn from top to bottom (14100). Because stars are just plotted points (14130), color is inevitable even when specifying HCOLOR=3 (white) or HCOLOR = 7 (white). By alternating between the two types of white, all four colors are plotted (green, blue, violet and red).

The Zyphyrs

The Zs are calculated next. Each will appear on a specific row corresponding to a text page row, for reasons to be discussed later. Only one Z per column and per row is permitted (15010). COL%(n) is then filled with the row number of the Z that occupies it.

Ready to play

The game is now ready to play. The text displays are printed. in a unique way so that they appear below the text window. Normally, with a string of more than one character, only the first character would be printed outside the window; the rest would appear inside. Therefore, a long string must be VTABbed and HTABbed into place one character at a time.

The main game sequence is controlled in lines 100 to 190, a loop with GOSUBs. This allows for the addition of other routines. For example, this could be made into a two-player game with a tank or gun moving along the bottom that shoots upward at the other player's ship or at the Zs.

GOSUB 1000

The first GOSUB moves the ship in response to the paddle. It draws and redraws the ship. But drawing and redrawing anything on a single hi-res page will make it flicker. So instead of using two entire hi-res pages to remove the flicker effect (which uses up too much memory), remove the flicker by allowing the program to erase the ship and redraw it only when the position of the ship must be changed (1100).

GOSUB 2000

The second GOSUB is more complex and is composed of its own series of GOSUBs. It moves the Zs, but only one at a time and at random. To move them, the program must first check if the random Z it selected still exists (it could have been shot). If it doesn't exist, then that's the end of that. If the Z is still around, the program then generates another random number and checks if that particular column is empty. If it isn't, then the program runs off to a routine that allows the Z to shoot down at the city. But if the column is empty, the program checks to see if the Z should shoot at your ship. The chances of it shooting at you will increase as you ascend the various levels (by wiping out all ten Zs).

Ten zipping Zyphyrs

Finally, the Z gets to move. It deletes itself from the old COL% and places its number in the new COL%. It changes its XUF% value accordingly. Now comes the illusion of movement, the "zip." A FOR-NEXT loop is initiated that goes from 3 to 4 because it determines the HCOLOR (white is 3 and black is 4). Drawing and undrawing the Z is a GOSUB inside a GOSUB. And, instead of storing a Z shape in the vector table a more flexible but slower format (for variety) was chosen that permits easier alteration of the Z shape (200-260) without changing the shape table itself.

GOSUB 3000

The third main GOSUB is conditional and depends on whether the paddle button is depressed at the time that the routine checks it. If it isn't depressed, the program loops back on itself. If it is pressed, that means that the ship is shooting. The program checks to see if a Z occupies the column. If so, the Z vanishes and U% (UFO counter) is raised by one. If U% is over 9, a new level has been reached and ten new Zs are created. If U% is not over 9, any building below will be completely wiped out and your score will be decremented by a thousand points. When the player accumulates a negative 100,000 points, the game ends.

Switching screens

When a Z is hit, the hit points are flashed in the same place where the Z expired. That effect is accomplished by switching screens rapidly (9030-9080). In this way, one can project text on the hi-res page for brief moments without using a character (or block-graphics) generator.

Conclusion

This version is small enough to fit comfortably beneath hi-res page 1. However, if you add any more routines or remarks, consider loading it above hi-res page 1 to avoid wiping out the tail end of your program when you run it. Use a loader program similar to the following:

```
10 TEXT: HOME: POKE 103,1: POKE
     104,64: POKE16384,0: PRINT
     CHR$(4)"RUN ZYPHYR WARS"
```

Also, if an Applesoft compiler is available, try compiling it to make it more challenging.

BASIC listing starts here

```
1 REM ** START OF PROGRAM **
2 TEXT : HOME : HGR
3 REM ZYPHYR WARS COPYRIGHT 1982
    SOFTKEY PUBLISHING P.O.BOX 44549
    TACOMA, WA 98444
10 DIM HEIGHT%(40), XUFO%(10),
    YUFO%(10), COL%(40): GOTO 10000
99 REM MAIN GAME SEQUENCER
100 GOSUB 1000
110 GOSUB 2000
120 IF PEEK (B1) > 127 THEN GOSUB
    3000
190 GOTO 100
199 REM DRAW ZYPHYRS
200 ROT= 0: X% = XUFO%(Z) * 7: Y% =
    YUFO%(Z): IF Y% = 0 THEN RETURN
205 SCALE=1: DRAW 6 AT X%+3, Y% - 2
210 SCALE=5: DRAW 6 AT X%+1, Y% - 1
220 SCALE= 1: DRAW 5 AT X%, Y%
225 SCALE=11: DRAW 6 AT X%-2, Y% + 1
230 SCALE=13: DRAW 6 AT X%-3, Y% + 2
235 SCALE= 7: DRAW 6 AT X%, Y% + 3
240 SCALE= 1: DRAW 5 AT X%, Y% + 4
250 RETURN
260 X% = XUFO%(COL%(PN%)) * 7: Y% =
    YUFO%(COL%(PN%)): ROT= 0: GOTO
    205
299 REM BUZZ!
300 FOR S = 1 TO 3: N = PEEK (BUZZ):
    NEXT : RETURN
310 FOR S = 1 TO 10: N = PEEK (BUZZ):
    NEXT : RETURN
320 N = PEEK (BUZZ) - PEEK (BUZZ):
    RETURN
349 REM DISPLAY GUNS
350 HTAB 3: GUN$ = "<" + STR$ (G
    UN%) + ">": INVERSE : VTAB 21:
    FOR A = 1 TO LEN (GUN$) : PRINT
    MID$ (GUN$, A, 1);: NEXT :
    NORMAL : RETURN
359 REM DISPLAY LEVEL
360 VTAB 21: HTAB 35: LEVEL$ = "(" +
    STR$ (LEVEL%) + ")": INVERSE :
    FOR A = 1 TO LEN (LEVEL$) :
    PRINT MID$ (LEVEL$, A, 1);: NEXT
    : NORMAL : GOSUB 5300: RETURN
399 REM DRAW RAY
400 ROT= 16: SCALE= 3
410 HCOLOR= 3: FOR A = 8 TO TY% STEP
    TY% / 10: DRAW 6 AT TX%, A: POKE
    6, A: POKE 7, 3: CALL 768: NEXT
420 HCOLOR= 0: FOR A = 8 TO TY% STEP
    TY% / 10: DRAW 6 AT TX%, A: NEXT
440 RETURN
549 REM LEVEL COUNTER
550 U% = U% + 1: IF U% > 9 THEN U% =
    0: LEVEL% = LEVEL% + 1: GOSUB
    15000
560 RETURN
599 REM ZYPHYR SHOOTS
600 UX% = XUFO%(Z) * 7 + 3: UY% =
    YUFO%(Z): ZX% = XUFO%(Z): ZH% =
    HEIGHT%(ZX%): IF SF% > 0 THEN
    SF% = 0: GOTO 660
610 FOR AA = 3 TO 4: HCOLOR= AA:
    HPLOT UX%, UY% TO UX%, ZH%: NEXT
    : GOSUB 1000: GOSUB 5100:
620 GOSUB 900
640 HEIGHT%(ZX%) = ZH%: RETURN
650 UX% = PN% * 7 + 3: UY% = 7: ZX% =
    PN%: ZH% = HEIGHT%(ZX%): GOTO
    620
660 SCALE= 3: ROT= 16: ZH% = 0: IF
    PN% = ZX% THEN ZH% = 5
670 FOR AA = 3 TO 4: HCOLOR= AA:
    HPLOT UX%, UY% TO UX%, ZH%:
    GOSUB 320: NEXT : GOSUB 5200:
    GOSUB 300: IF ZH% = 5 THEN GOTO
    700
680 RETURN
699 REM SATELLITE FALLS
700 ROT= 0: SCALE= 1
710 HCOLOR= 0: DRAW 8 AT PN% * 7 , 5
720 ROT= 32: FOR A = 0 TO
    HEIGHT%(PN%) STEP 2
730 FOR AA = 1 TO 2
740 XDRAW 8 AT PN% * 7 + 7, A: POKE
    6, A: POKE 7, 6: CALL 768: N =
    PEEK (BUZZ)
750 NEXT : NEXT : GUN% = GUN% - 1 :
    GOSUB 5000: IF GUN% = 0 THEN 850
799 REM GROUND EXPLOSION
800 HCOLOR= 0
```

```
810 IF HEIGHT%(PN%) > = EARTH THEN
    FOR AA = 1 TO 2: FOR A = 2 TO 6:
    SCALE= 3 * A: FOR R = 0 TO 20
    STEP 2: FOR RR = - 1 TO 1 STEP 2:
    ROT= 112 - R * RR: XDRAW 6 AT PN%
    * 7 + 3, EART H: NEXT : N = PEEK
    (BUZZ): NEXT : NEXT : NEXT :
    GOSUB 350
819 REM WHOLE BLDG. GONE
820 FOR AA = HEIGHT%(PN%) TO EAR TH
    STEP 2: HPLOT PN% * 7 - 1 , AA TO
    PN% * 7 + 7, AA: HPLOT PN% * 7 -
    2, AA - 1 TO PN% * 7 + 8, AA - 1:
    N = PEEK (BUZZ ): NEXT
830 HEIGHT%(PN%) = EARTH: GOSUB 350
840 ADD = - 1000 * LEVEL%: GOSUB
    9020: RETURN
849 REM END OF GAME
850 A$ = " END OF GAME !": FOR B = 1
    TO 15: FOR AA = 1 TO B: VTAB 10:
    HTAB 13: PRINT LEFT$ (A$, B)
860 POKE - 16303, 0: GOSUB 310: POKE
    - 16304, 0: NEXT AA: A = B * 13:
    GOSUB 5300
870 NEXT B: TEXT : VTAB 1: HTAB 10:
    PRINT "ANOTHER GAME? <Y> <N> ";
880 GET A$: IF A$ = "Y" THEN RUN
890 HOME: END
899 REM BUILDING LOSES A LEVEL
900 FOR L = 1 TO LEVEL%: HCOLOR= 0:
    ROT= 0
910 SCALE= 7: DRAW 6 AT ZX% * 7, ZH%
    - 1: GOSUB 320
920 SCALE= 9: DRAW 6 AT ZX% * 7 -
    1,ZH%: GOSUB 320
930 HCOLOR= 5: SCALE= 7: DRAW 6 AT
    ZX% * 7,ZH% + 1
940 ADD = - 10 * LEVEL%: GOSUB 9020:
    ZH% = ZH% + 2: IF ZH% < EARTH
    THEN NEXT: RETURN
950 IF ZH% > EARTH + 4 THEN ZH% =
    EARTH + 4: CC = CC + 1: IF CC > 9
    THEN GOSUB 6000
960 ADD = - 20 * LEVEL%: GOSUB 9
    020: RETURN
999 REM MOVE SATELLITE
1000 PN% = PDL (1) / 5 + 1: IF P N% >
    38 THEN PN% = 38
1050 ROT= 0: SCALE= 1
1100 IF PN% < > PO% THEN HCOLOR= 0:
    DRAW 8 AT PO% * 7,5
1200 HCOLOR= 3: DRAW 8 AT PN% * 7,5
1300 PO% = PN%
1400 RETURN
1999 REM MOVE ZYPHYRS
2000 Z = RND (1) * 10 + 1
2050 IF YUFO%(Z) < 1 THEN RETURN
2100 Q% = RND (1) * 37 + 1
2200 IF COL%(Q%) > 0 THEN GOTO 600
2300 X1% = XUFO%(Z) * 7: X2% = Q% * 7
2400 IF RND (1) * 10 < LEVEL% THEN
    GOSUB 1000: SF% = 1: GOSUB 600
2500 C = 5
2600 IF X1% > X2% THEN C = - C
2700 FOR AA = 5 TO 4 STEP - 1: HCOLOR
    = AA: HPLOT X1%,YUFO%(Z) TO
    X2%,YU FO%(Z): NEXT: GOSUB 1000:
    GOSUB 5400
2750 COL%(XUFO%(Z)) = 0
2800 HCOLOR= 0: GOSUB 200: XUFO%(Z)
    = Q%: HCOLOR= 3: GOSUB 200
2900 COL%(Q%) = Z: RETURN
2999 REM SATELLITE SHOOTS
3000 TX% = PN% * 7 + 4
3010 IF COL%(PN%) = 0 THEN TY% =
    HEIGHT%(PN%) - 2: GOSUB 400:
    HCOLOR= 0: GOSUB 820: RETURN
3020 TY% = YUFO%(COL%(PN%)): GOSUB
    400: HCOLOR= 0: GOSUB 260
3030 YUFO%(COL%(PN%)) = 0: COL%(PN%)
    = 0: GOSUB 9000: GOSUB 550:
    RETURN
4999 REM SOUND OF DESTRUCTION
5000 FOR A = 1 TO 10: POKE 6, RND (1)
    * 50 + 1: POKE 7,250: CALL 768:
    NEXT: RETURN
5100 FOR A = 100 TO 200 STEP 20:
    POKE 6,A: POKE 7,2: CALL 76 8: N
    = PEEK (BUZZ): GOSUB 1000: NEXT:
    RETURN
5200 FOR A = 200 TO 250 STEP 10:
    POKE 6,A: POKE 7,4: CALL 768:
    NEXT: RETURN
5300 FOR AA = 251 TO 1 STEP - 1 0:
    POKE 6,A: POKE 7,5: CALL 768:
    NEXT: RETURN
5400 FOR A = 1 TO 30 STEP 3: POKE
    6,A: POKE 7,2: CALL 768: NEXT:
    RETURN
5999 REM CRATER COUNTER
6000 CC = 0: FOR I = 1 TO 38
6010 IF HEIGHT%(I) < EARTH THEN 6090
6050 CC = CC + 1
6060 IF CC > 5 THEN GOSUB 8000: GOTO
    850
6090 NEXT: RETURN
8000 FOR B = 1 TO 10: IF YUFO%(B) < 1
    THEN 8050
8010 HCOLOR= 1: HPLOT XUFO%(B) * 7 +
    4,YUFO%(B) TO 140,90: GOSUB 5400
8050 NEXT B: SCALE= 50: FOR A = 0 TO
    64: ROT= A: XDRAW 6 AT 140,90:
    NEXT A
8099 REM BIG BOMB
8100 FOR A = 1 TO 50
8110 HCOLOR= RND (1) * 5 + 1
8120 X = RND (1) * 280: Y = RND (1) *
    190
8130 HPLOT 140,90 TO X,Y
8140 POKE 6,A: POKE 7,3: CALL 768
8150 NEXT A
8190 RETURN
8999 REM <DISPLAY SCORE>
9000 HOME: FUFO% = 1
9010 ADD = LEVEL% * 100: VTAB TY% /
    8 + 1: HTAB PN%: PRINT ADD
9020 PTS = PTS + ADD: SC$ = STR$
    (PTS): VTAB 21: HTAB 17: INVERSE
9030 FOR A = 1 TO LEN (SC$)
9040 IF FUFO% THEN POKE - 1630 3,0:
    N = PEEK (BUZZ)
9050 PRINT MID$ (SC$,A,1);
9070 POKE - 16304,0
9080 NEXT: PRINT "": NORMAL: IF PTS
    < - 100000 THEN GOSUB 8000: GOTO
    850
9090 HOME: FUFO% = 0: RETURN
9999 REM INIT VARIABLES
10000 EARTH = 152: LEVEL% = 1: GUN%
    = 5: PTS = 0: CC = 0
10010 BUZZ = - 16336: KEY = - 16384:
    KBOARD = - 16368: B1 = - 16286
10020 HC = - 1
10030 Z = 1
10070 G$ = CHR$ (7) + CHR$ (7)
10080 SCALE= 1: ROT= 0
10090 IF AGAIN > 0 THEN 12000
10100 DATA 166,7,164,6,173,48,19
    2,136,208,253,202,208,245,96 ,0
10999 REM SHAPE TABLE
11000 DATA 9,0,20,0,24,0,29,0,33
    ,0,37,0,41,0,43,0,45,0,0,0
11010 DATA 77,45,45,0
11020 DATA 45,9,45,5,0
11030 DATA 45,77,45,0
11040 DATA 45,109,41,0
11050 DATA 77,77,5,0
11060 DATA 5,0
11070 DATA 32,0
11080 DATA 44,44,53,54,46,36,36,
    45,46,54,0
11090 DATA -1
11100 START = 768
11110 MEM = START
11120 READ QUANTITY
11130 IF QUANTITY < 0 THEN 11200
11140 POKE MEM,QUANTITY
11150 MEM = MEM + 1
11160 GOTO 11120
11200 POKE 232,15: POKE 233,3
11999 REM SET UP GAME
12000 GOSUB 13000
12010 GOSUB 14000
12020 GOSUB 14100
12030 GOSUB 15000
12040 GOSUB 350
12090 GOTO 100
12999 REM MAKE EARTH
13000 HCOLOR= 3
13010 FOR A = EARTH TO EARTH + 1 0
13020 HPLOT 0,A TO 279,A
13030 NEXT
13100 INVERSE
13110 FOR A = 21 TO 23: FOR AA = 1 TO
    40
13120 VTAB A: HTAB AA: PRINT " ";
13130 NEXT: NEXT
13140 VTAB 23: HTAB 12: PRINT
    " > ZYPHYR WARS! <";: NORMAL :
    VTAB 24: HTAB 3: PRINT "COPY
RIGHT 1982 BY SOFTKEY PUBLISHING";
13180 POKE 35,19
13190 RETURN
13999 REM CALCULATE BLDGS
14000 FOR A = 1 TO 38
14010 HEIGHT = ( RND (1) * 20) * 2
14020 HEIGHT%(A) = EARTH - HEIGHT
14030 FOR B = EARTH TO HEIGHT%(A )
    STEP - 2
14040 R% = RND (1) * 5 + 1
14050 SCALE=7: XDRAW 6 AT A*7,B-1
14060 SCALE= 1: XDRAW R% AT A * 7,B
14070 POKE 6,B: POKE 7,5: CALL 7 68
14090 NEXT : NEXT : RETURN
14099 REM DRAW STARS
14100 FOR A = 0 TO EARTH
14110 HC = - HC: HCOLOR= 3: IF H C <
    0 THEN HCOLOR= 7
14120 STAR = RND (1) * 279 + 1
14130 HPLOT STAR,A
14140 POKE 6,A + 1: POKE 7,9: CALL
    768
14150 NEXT
14190 RETURN
14999 REM CALCULATE UFOS
15000 FOR A = 1 TO 10
15010 T% = RND (1) * 37 + 1: IF
    COL%(T%) > 0 THEN 15010
15020 XUFO%(A) = T%
15030 YUFO%(A) = A * 8 + 12
15040 HCOLOR= 3:Z = A: GOSUB 200
15050 COL%(T%) = A
15060 NEXT
15090 GOSUB 360: RETURN
```

Zyphyr Wars Checksums

```
1   - $97CE     940   - $4651     10999 - $A2FA
2   - $AFC9     950   - $4271     11000 - $F4E0
3   - $2A5F     960   - $FB9F     11010 - $47E6
10  - $51D9     999   - $134E     11020 - $A154
99  - $0A89     1000  - $6C31     11030 - $0816
100 - $91A7     1050  - $04A3     11040 - $2DE9
110 - $C2C8     1100  - $81A7     11050 - $3969
120 - $9CB5     1200  - $73DC     11060 - $482B
190 - $2D28     1300  - $3E0D     11070 - $1897
199 - $4FFD     1400  - $904D     11080 - $9A13
200 - $E8DE                       11090 - $0008
205 - $5803     1999  - $7B6D     11100 - $2489
210 - $AA9B     2000  - $1F15     11110 - $C96E
220 - $B03B     2050  - $E52E     11120 - $33A8
225 - $1155     2100  - $2912     11130 - $301F
230 - $C568     2200  - $DE15     11140 - $1F18
235 - $CFD3     2300  - $2833     11150 - $86DF
240 - $0E6A     2400  - $7DE0     11160 - $81B2
250 - $EB33     2500  - $92DB     11200 - $BD3A
260 - $5481     2600  - $A93B     11999 - $BE1B
                2700  - $9717
299 - $389D     2750  - $D059     12000 - $3E11
300 - $CA1D     2800  - $8677     12010 - $9CA6
310 - $DA63     2900  - $76EC     12020 - $2423
320 - $5D5F     2999  - $9091     12030 - $8431
349 - $0EDB     3000  - $5F7A     12040 - $2C44
350 - $297D     3010  - $C658     12090 - $669A
359 - $EC52     3020  - $2689     12999 - $10CC
360 - $B364     3030  - $FD18     13000 - $12AA
399 - $E136     4999  - $10BE     13010 - $FB0D
400 - $E3AD     5000  - $7934     13020 - $DC52
410 - $392C                       13030 - $9DC4
420 - $7D58     5100  - $1527     13100 - $6D53
440 - $834C     5200  - $3BE2     13110 - $5CA3
549 - $4694     5300  - $4870     13120 - $8F5E
550 - $2B40     5400  - $8C2A     13130 - $D4A1
560 - $6C25     5999  - $C8CF     13140 - $80FA
599 - $3BC8     6000  - $CD81     13180 - $1F4F
600 - $11CF     6010  - $2D62     13190 - $5DD9
610 - $3556     6050  - $E6FB     13999 - $2D7D
620 - $A89B     6060  - $2798     14000 - $FD12
                6090  - $F3D0
640 - $9701     8000  - $BC47     14010 - $9763
650 - $5F7F     8010  - $D4D6     14020 - $F31D
660 - $487B     8050  - $4832     14030 - $945B
670 - $1590     8099  - $B4DD     14040 - $3B7C
680 - $6C3F     8100  - $C822     14050 - $9F58
699 - $F952     8110  - $6A05     14060 - $9F3B
700 - $6445     8120  - $4B1D     14070 - $5F79
710 - $9FDD     8130  - $BCB9     14090 - $2B71
720 - $B4F3     8140  - $8EBC     14099 - $67A4
730 - $9B74     8150  - $B8F7     14100 - $D6BF
740 - $372A                       14110 - $13EE
750 - $CA21     8190  - $9372     14120 - $9AED
799 - $3D4B     8999  - $5A9C     14130 - $0158
800 - $4182     9000  - $B0A6     14140 - $3943
810 - $0997     9010  - $FD6D     14150 - $6882
819 - $FBA6     9020  - $7CB6     14190 - $B648
820 - $5C05     9030  - $C38F     14999 - $5E78
830 - $DA51     9040  - $6E93     15000 - $2D43
840 - $F41E     9050  - $A095     15010 - $9CB9
849 - $3648     9070  - $6820     15020 - $4AE2
                9080  - $7BA3
850 - $2016     9090  - $CAFA     15030 - $E94D
860 - $0105     9999  - $90F1     15040 - $028F
870 - $2193     10000 - $8F23     15050 - $0ADF
880 - $428C     10010 - $9429     15060 - $4825
890 - $CB01     10020 - $4A27     15090 - $61AF
899 - $E578     10030 - $A70D
900 - $F24F     10070 - $DD68
910 - $80CF     10080 - $F066
920 - $01F2     10090 - $CA52
930 - $1374     10100 - $F45C
```

INVADERS continued from page 57

```
12000 VTAB 5: HTAB 7: PRINT "WHAT IS
    YOUR SKILL LEVEL?";: VTAB 7:
    HTAB 9: PRINT "1....2....
    3....4....5";: VTAB 9: HTAB 14:
    PRINT "< PICK ONE >";
12010 A = 5
12020 KE% = PEEK ( - 16384): IF KE%
    < 127 THEN 12050
12030 IF KE% < 177 OR KE% > 181 THEN
    GOTO 12080
12040 SK% = KE% - 176: GOTO 12100
12050 VTAB 7: HTAB A * 5 + 4:
    INVERSE : PRINT A;:ZZ = PEEK
    (NO) - PEEK (NO): FOR C = 1 TO
    15: NEXT C: NORMAL : HTAB A * 5 +
    4: PRINT A;
12060 A = A + 1: IF A > 5 THEN A = 1
12070 GOTO 12020
12080 VTAB 2:INVERSE: FOR B=1 TO LEN
    (ER$): HTAB B+5: PRINT MID$((ER$
    ),B,1);: Z = PEEK (NO): NEXT B
12090 PRINT CHR$ (7);: POKE ST, 0:
    HTAB 5: PRINT RIGHT$ (SP $, LEN
    (ER$) +1): NORMAL : GOTO 12020
    12100 VTAB 5: HTAB 1: CALL - 86
    8: VTAB 7: CALL - 868: VTAB 9:
    CALL - 868:
12200 GOSUB 13000: VTAB VINVADER %:
    HTAB HINVADER%: PRINT
    INVADER$(1)
12210 GOSUB 400
12900 GOTO 1000
12999 END
13000 IF SK% < 6 THEN SK% = SK% + 1
13010 FOR A = 0 TO 2:IN$(A) = " ":
    NEXT A
13020 N% = SK% + 2: FOR A = 1 TO
    N%:IN$(1) = IN$(1) + "]" + I $ +
    CHR$ (91) + " "
13030 IN$(0) = IN$(0) + "/ " + CHR$
    (220) + " ":IN$(2) = IN$(2) +
    "< > "
13040 NEXT A: RETURN
13200 INVERSE : GOSUB 220: GOSUB
    13290: VTAB IV%: HTAB IH%: PRINT
    ".";: GOSUB 13290: NORMAL :
    GOSUB 220
13210 VTAB VG%: HTAB HG% - 2: PRINT
    "=:#:=";: GOSUB 13290: VTAB VG%
    - 1: HTAB HG%: PRINT "'" ;:
    GOSUB 13290
13220 VTAB VG%: HTAB HG% - 2: PRINT
    "- * -";: GOSUB 13290: VTAB VG%
    - 1: HTAB HG%: PRINT "." ;:
    GOSUB 13290
13230 VTAB VG%: HTAB HG% - 2: PRINT
    "  -  ";: GOSUB 13290: VTAB VG%
    - 1: HTAB HG%: PRINT " " ;:
    GOSUB 13290
13240 VTAB VG%: HTAB HG% - 2: PRINT
    "";: GOSUB 13290
13250 FOR E = 1 TO 100: NEXT E
13260 HG% = 19: GOSUB 400: RETURN
13290 FOR E = 1 TO 10:BU = PEEK
    (NO): NEXT E: RETURN
20000 AG% = AG% + 1
20010 VTAB VI%: HTAB 1: CALL - 868:
    VTAB VI% + 1: HTAB 1: CALL - 868
20020 VI% = SK% +AG%: HI% =3: IT% =1
20030 GOSUB 13000
20090 GOTO 1000
21000 HOME : VTAB 5: HTAB 5: PRINT
    "YOUR FINAL SKILL LEVEL WAS "SK%
21010 VTAB 7: HTAB 5: PRINT "YOUR
    FINAL SCORE WAS "PT%
21070 VTAB 13: HTAB 5: PRINT "DO YOU
    WISH TO TRY AGAIN? (Y/N)": GET
    AN$: IF AN$ = "Y" THEN GO% = 1:
    GOTO 10000
21080 IF AN$ = "N" THEN TEXT : HOME
    : END
21090 GOTO 21070
```

Checksums for Text Invaders

```
9   - $A726     790 - $2300     10550 - $FCDE
10  - $49B4     800 - $07C3     10560 - $1DF0
20  - $F890     810 - $888C     10570 - $9B68
30  - $20A7     820 - $B0DE     10580 - $2E96
35  - $EB1D     830 - $22DC     10590 - $68BA
40  - $BE01     840 - $0EC7     10600 - $2BFA
45  - $DB40     850 - $DDED     10800 - $53EF
50  - $A838     860 - $9262     10900 - $7738
52  - $4155     880 - $7575     10910 - $522D
55  - $1C42     890 - $57D9     11000 - $E960
60  - $FD78     900 - $AB0B     11010 - $E659
65  - $080A     905 - $F699     11020 - $DD6E
70  - $EA60     910 - $F7C3     11030 - $FF0D
72  - $FEA6     915 - $9B4B     11050 - $B73D
75  - $C683     920 - $B160     11060 - $6CA0
80  - $3AAF     930 - $55E0     11070 - $B84B
85  - $BB54     935 - $DB08
90  - $AAC0     940 - $42EF     11100 - $97A0
100 - $0379                     11110 - $C6BE
110 - $71B9     945 - $18E4     11120 - $B789
                950 - $C1F8     11200 - $5CC3
120 - $89F9     960 - $14FE     11210 - $B02E
150 - $DA2D     970 - $CCE8     11220 - $C30F
155 - $F374     980 - $0612     11230 - $0537
160 - $79D6     990 - $2B62     12000 - $B135
170 - $5B7D     1000 - $C2DE    12010 - $FAEC
190 - $1A60     1010 - $04CC    12020 - $A3CB
200 - $8CB1     1020 - $9564    12030 - $4276
210 - $E4C0     1040 - $5737    12040 - $F0C4
220 - $D8D9     1050 - $ACF4    12050 - $3935
230 - $4EFF     1080 - $E3F3    12060 - $4928
300 - $4F3F     1090 - $DB61    12070 - $8958
350 - $3002     1500 - $BE8C    12080 - $6165
400 - $027C     1510 - $4A95    12090 - $FE3E
410 - $6DF8     1520 - $5341    12100 - $BDFF
420 - $B090     1530 - $84B1    12200 - $BE20
430 - $7415     1600 - $FD18    12210 - $EC27
450 - $98EE     1610 - $7AF9
500 - $50B3     1999 - $4F47    12900 - $39CE
510 - $7A89                     12999 - $DEDE
520 - $437D     2000 - $C0E7    13000 - $AECA
                2010 - $D827    13010 - $E6AC
530 - $B720     2400 - $6913    13020 - $7E06
540 - $1ACE     2410 - $8EB2    13030 - $F3D0
550 - $695D     2420 - $29E5    13040 - $C1C3
570 - $7EA2     2430 - $96AE    13200 - $D5DE
580 - $1514     2500 - $D30C    13210 - $D461
590 - $5EBD     2510 - $34EC    13220 - $8476
600 - $CAB3     2520 - $22D7    13230 - $DFFC
605 - $108C     10000 - $9D2A   13240 - $AFF0
610 - $9B0E     10100 - $2C1F   13250 - $67B3
615 - $2333     10110 - $F29E   13260 - $FB1D
620 - $86C2     10120 - $0014   13290 - $B8D9
625 - $C049     10130 - $7002   20000 - $CFB8
630 - $FB28     10140 - $A7E6   20010 - $0421
640 - $A278     10150 - $AD67   20020 - $D6D5
650 - $1750     10160 - $3FB0   20030 - $890F
660 - $5684     10170 - $7E62   20090 - $CF68
670 - $79F0     10180 - $0FA6
690 - $841A     10500 - $0103   21000 - $AB0A
700 - $EE7E                     21010 - $0414
750 - $D459     10510 - $A600   21070 - $2834
                10520 - $8C9D   21080 - $B161
770 - $DDE4     10530 - $ACFF   21090 - $80A8
780 - $5D6D     10540 - $7D8E
```

Check your binary and Applesoft listings for errors with

Checksoft and Checkbin

By Robb Canfield

For the benefit of those who will be typing in the listings in this volume, we included two utilities (Checksoft and Checkbin) from the first issues of "The hardcore COMPUTIST" (not Hardcore computing...confused?). Properly entered and used, they will help readers identify their typographical errors in listings from our magazines.

How to enter Checksoft

Checksoft must first be typed into memory as a series of bytes.

1. Enter the monitor.

```
CALL -151
```

The machine language prompt should appear (as an asterisk "*"). If it doesn't, try CALL -151 again. If the prompt still refuses to appear, then something is definitely wrong with your Apple (you are using an Apple, aren't you?).

2. Type the following:

```
0300:A9 4C 8D F5 03 A9 10 8D
0308:F6 03 A9 03 8D F7 03 60
0310:20 8E FD A9 10 8D 96 03
0318:A9 FB 8D 97 03 A9 14 85
0320:0A A5 67 8D C8 03 85 0B
0328:A5 68 8D C9 03 85 0C A2
0330:00 A0 00 20 C7 03 E0 02
0338:90 22 E0 04 B0 03 48 90
0340:0E C9 00 F0 1D C0 FF F0
0348:13 C9 B2 D0 02 A0 FF 6A
0350:45 0B 2A 45 0C 85 0B 45
0358:0C 6A 85 0C 20 BF 03 E8
0360:D0 D4 68 A8 68 AA 98 20
0368:24 ED 38 A9 06 E5 24 AA
0370:20 4A F9 A0 02 B9 CB 03
0378:20 ED FD 88 10 F7 A5 0B
0380:A6 0C 20 41 F9 20 8E FD
0388:C6 0A D0 1F A9 14 85 0A
0390:20 8E FD AD 00 C0 10 FB
0398:8D 10 C0 C9 83 F0 1C C9
03A0:9B D0 08 A9 EA 8D 96 03
03A8:8D 97 03 A2 00 A0 00 20
03B0:BF 03 D0 82 E0 01 F0 03
03B8:E8 10 F4 20 8E FD 60 EE
03C0:C8 03 D0 03 EE C9 03 AD
03C8:FF FF 60 A4 A0 AD
```

3. Double check what you have typed and fix all errors before continuing.

```
300.3CD
```

4. Return to BASIC

```
3D0G
```

5. Save Checksoft to your disk.

```
BSAVE CHECKSOFT,A$300,L$CE
```

How to Use Checksoft

Checksoft is our program to inspect your Applesoft listings for any typographical errors. Using Checksoft is easy if you follow the three steps below.

1) Enable the program.

```
BRUN CHECKSOFT
```

2) Type any Applesoft listing or load one into memory.

3) Press the ampersand key (&) and Return. The checksums for the first twenty lines will appear. Compare the checksums on your screen with the checksums we've published for that program.

If they match exactly, your program has been typed in correctly up to the last line number listed, so press the space bar to display the next twenty lines.

If the checksums don't match, then there is an error in the first line in which your checksums disagree with ours. Press ctrl C to exit Checksoft and make your corrections. Press ampersand and Return to restart Checksoft.

How to enter Checkbin

Checkbin is entered the same way as Checksoft.

1. Enter the monitor.

```
CALL -151
```

2. Type the following:

```
0300:20 58 FF BA CA BD 00 01
0308:18 69 1F 8D F9 03 85 62
0310:E8 BD 00 01 69 00 8D FA
0318:03 85 63 A9 4C 8D F8 03
0320:60 20 8E FD A9 0A 85 0A
0328:A0 00 84 31 20 A7 FF A9
0330:FF 85 31 A5 3C 85 0B A5
0338:3D 85 0C 20 A7 FF A0 55
0340:A9 10 91 62 A9 FB C8 91
0348:62 A0 00 F0 45 A5 3C 29
0350:07 D0 42 38 A9 1F E5 24
0358:AA 20 4A F9 A9 A4 20 ED
0360:FD A5 0B A6 0C 20 41 F9
0368:C6 0A D0 26 20 8E FD A9
0370:0A 85 0A AD 00 C0 10 FB
0378:8D 10 C0 C9 83 F0 48 C9
0380:A0 F0 BB C9 9B D0 0B A9
0388:EA A0 55 91 62 C8 91 62
0390:A0 00 20 92 FD A9 A0 20
0398:ED FD B1 3C 48 20 DA FD
03A0:68 6A 45 0B 2A 45 0C 85
03A8:0B 45 0C 6A 85 0C 20 BA
03B0:FC 90 9A A9 1F E5 24 AA
03B8:20 4A F9 A9 A4 20 ED FD
03C0:A5 0B A6 0C 20 41 F9 20
03C8:8E FD 8D 10 C0 60
```

3. Double check what you have typed.

```
300.3CE
```

Compare this to the printed list and correct any differences.

4. Return to BASIC

```
3D0G
```

5. Save Checkbin to your disk.

```
BSAVE CHECKBIN,A$300,L$D0
```

How to Use Checkbin

Checkbin is our program to inspect your binary listing for typographical errors. Like Checksoft, Checkbin is very easy to use.

1) First BRUN the Checkbin program. Many of our machine code listings are loaded into the same area of memory which contains Checkbin (300-3CE). Thus, you should always BRUN Checkbin at some out-of-the-way location, so that the listing you are checking does not overwrite the Checkbin routine. To do this, simply

```
BRUN CHECKBIN,A$8000
```

2) Now type in one of our hex dumps (not source code). Of course, if you have already typed it in, you may BLOAD the binary file from your disk.

3) The last step is to call the Checkbin routine. To do this, you must specify where the binary program begins and ends in memory. These values are contained in each checkbin checksum listing.

You must be in the monitor to call the Checkbin routine, enter

```
300.3CE ctrl Y
```

The ctrl Y works in the same way as the ampersand does for Checksoft.

The first ten lines of the hex dump will appear with the checksums printed on the extreme right of every line. Use these checksums exactly the same way you used the checksums for Checksoft. Press the space bar to examine the next ten lines.

That's all there is to it.