DISK DOCTOR

DISK DOCTOR
COPYRIGHT AJM SOFTWARE

This program is copyrighted and it is illegal to give or sell copies without the approval of the author. You should, however, make a backup of the program using any media copy utility, and store it in a safe place.

This utility is similar in many respects to FILEMANAGER which is a comprehensive disk and file management system. If you are familiar with FILEMANAGER, it should be relatively easy to use this utility. Disk Doctor has only one purpose in mind: help rebuild damaged EOS directories. It has been carefully designed to be error free, smooth flowing, and user friendly. Since it does write to disks or tapes, it is recommendable to work from a backup copy, even when working with damaged disks. The program has the following features:

Analysis of disk or tape contents
Echo REM statements from programs
Blank out damaged directories
Initializing up to 6K of directory
Compatibility with single/double/quad density
Modification of directory entries
Simultaneous viewing of disk contents
Automatic marking of end-of-file
Forward referencing directory entries

The Program will NOT reformat disks or tapes, or repair damaged tracks/sectors. It will only be helpful if the Directory or CATALOG of the medium has been trashed and the rest of the medium still contains valid programs/files. Because of the way that Disk Doctor calculates allocation blocks, it should not be used for routine catalog editing functions. Use FILEMANAGER for that.

WARNING: This utility will write to disks or tapes ignoring standard protection features (except write protect tabs). It is essential to carefully read all instructions, especially those on directory structure before using Disk Doctor.

Address any questions about the program to:

Guy Cousineau
1059 Hindley Avenue
OTTAWA Canada
K2B 5L9

MENU

To start the utility, simply turn your ADAM on, insert your copy of DISK DOCTOR, and pull the reset switch. A title screen will appear and will remain while the program is being loaded. After a few seconds, a menu screen will appear:

```
          I     Edit Directory
         II     Read Block
        III     Analyze Medium
         IV     Change Default Setup

        source:
          V     Disk Drive 1 / C / D5
         VI     Single density
```

Remove the system disk at this point, it is no longer required. The last 2 entries denote the default setting for editing operations. The drive is shown as 'C' as is sometimes referred to in programs, and also by the 'D5' designation used by SMARTBASIC. To change this selection, simply press Smart Key V. To change from single/double/quad density, press Smart Key VI. Disk Doctor will run from either disk drive or either tape drive; the source drive selection will skip over any drives that are missing or inactive.

Note that many option settings can be toggled simply by pressing the appropriate Smart Key.

From any level of menu, pressing <ESCAPE> will back up one level. Pressing <ESCAPE> from the opening menu above will exit the program.

THE PROGRAM WILL WRITE TO DISK IN ONLY 2 CASES:

When you ask to INIT a bad EOS directory
When you ask to save changes after an editing session

The program will only write to the directory block(s) of the medium. Since it is possible to INIT up to 6K of directory, users should be careful not to overwrite data blocks.

EDIT DIRECTORY

This function is selected with Smart Key I from the opening menu. It should be used after obtaining a printout of the medium analysis. You should NOT start with READ BLOCK since it defeats the following check.

Each time EDIT CATALOG is entered, the entire directory (up to 6K) is read into memory and remains until the next restart. All directory information is available throughout the session. The first time you edit a damaged directory, you will probably get the following message:

Bad EOS Directory
INIT medium (Y/N)?

Be absolutely sure that you have the correct disk in the correct drive before you answer yes to this option -- you don't want to INIT the Disk Doctor system disk! You will be provided with an INIT menu selection screen from which you can select the volume name, disk size, and directory size. As a further protection, it will not be possible to change the medium to INIT. If you have made an error, press <ESCAPE> and change the drive from the main menu.

It is advisable to INIT only 1K of directory and modify the size later; the INIT option actually writes to all directory blocks. INIT will write a standard Header with VOLUME BOOT DIRECTORY and BLOCKS LEFT. The rest of the directory will be filled with 1's (not 0's), to set the BLOCKS LEFT attribute in all positions.

The EDIT DIRECTORY function is self correcting to save on calculations:

Next file start is calculated as previous_file_start + previous_file_length. This is essential for full compatibility.

Setting file length also sets used length. Used length must be set after if it is to be different. Note that it is essential to correctly set both the reserved_length and the used_length.

The user is not allowed to change the Start Block of a file; it is always calculated by Disk Doctor.

EDIT DIRECTORY FUNCTIONS

EDIT DIRECTORY displays one directory entry at a time and use the following function keys:

-<UP> <DOWN> to move forward or back one entry. You cannot move back before entry 0 or the past the last possible entry in the directory.
-<HOME UP> moves to entry 0.
-<HOME DOWN> moves to the BLOCKS LEFT entry (if found)
-<HOME> lets you select the entry number to go to. It won't let you advance past the BLOCKS LEFT entry. Note that this entry is calculated when EDIT DIRECTORY is selected from the menu and it is not affected by editing.
-<ESCAPE> aborts to main menu.
-<RETURN> returns to menu after checking for save option.
-<WILD> sends you to READ BLOCK. Note that this function will assign the start block of the file currently being edited to the read block routine. This makes it easy to examine the current file from the start in order to determine the end-of-file. Consult the directory structure information in the last section of this manual. You should NOT enter read block from the VOLUME entry or any other that will assign an illegal block number to the read block function.

EDIT FUNCTIONS

I Change name: may be up to 11 characters for entry 0 (VOLUME) or 10 characters for a file name. After 10 characters are entered or a <CR>, the cursor will move over and wait for a file type (A a H h); just press a key for the file type, don't follow with <CR>.

II Set Attributes: cursor up and down and press 'y/n' to toggle the bits. Pressing <CR> will exit this function. See the DIRECTORY STRUCTURE section for more details on attributes.

The start Block is shown but cannot be modified. It will be calculated automatically by DISK DOCTOR for all entries. It is therefore essential to properly set the 2 length bytes.

III Set Reserved Length: tells the directory the maximum size of the file. Setting the reserved length will also set the used length. Used length must be set after if it is to be smaller. DISK DOCTOR will not check if it is bigger. Be careful!

IV Set used length. This is the actual size of the file in 'K'.

V Set used length in last block. This is used to determine the end-of-file.

VI Set Date in YY/MM/DD format with ranges from 0-99, 0-12, 0-31. Zero values are allowed for special purposes and there is no check for valid day-month selection.

II READ BLOCK

Read Block can be entered from the menu with Smart Key II or by the WILD-CARD key in Edit Catalog. While in read block, the current file marker is preserved and will be restored when WILD-CARD is pressed again. Read block does not allow any disk writing. It is only used to verify file types and to determine the end_of_file. It uses the following function keys:

UDLR move around the sector being displayed.

<TAB> toggles between the ASCII and HEX section

<CR> beginning of next line

<HOME> top of sector

<HOME UP> previous sector. If at sector 0, sector 7 of the previous block is displayed.

<HOME DOWN> next sector. If at sector 7, sector 0 of the next block is displayed.

<CONTROL-UP> previous block. Will not go beyond 0.

<CONTROL DOWN> next block. Will not advance beyond disk size.

<WILD-CARD> returns to Edit Catalog with the current cursor position in memory. If you say yes, the current file size and used_in_last_block will be transferred to the directory entry.

Note that a BLOCK consist of 8 128-byte sectors numbered from 0 to 7. To 'read' through a file, press <HOME DOWN> until you get to the end-of-file. To jump ahead or back several blocks, press <ESCAPE> and Smart Key II from the edit block menu. Enter a decimal number and press <CR> twice to read in the selected block.

Many ASCII characters cannot be shown on the console and may clutter up the display. For this reason, control characters and values above 7FH will be represented with a '.'. This will make it easier to look for ASCII data in Binary files. Unlike FILEMANAGER, Disk Doctor will not show any inverse video characters.

Read block should also be used if you accidently get bounced to the main menu by pressing <ESC> from either read block or edit catalog. Choosing function I will re-read the directory and destroy all editing done since the last save. Instead, chose function II, then press WILD-CARD to resume the editing at the last file. Note that the SAVE CHANGES option is only available from the Edit Catalog routine and it is essential to re-enter this way. To cancel the current changes or log in a new disk, return to the main menu and press Smart Key I. It is advisable to save changes after every few entries to prevent accidental loss.

III ANALYZE MEDIUM

This feature of the program will be invaluable in decoding the files on your disks or tapes. It will distinguish between Word Processing files, Binary data files, shape tables, fast loaded programs, and ASCII files. In addition, it will search all ASCII files and fast loaded programs for REM statements. If you have taken the precaution of writing REM statements at the start and end of your programs, most of the hard work will take care of itself. Pressing Smart Key III from the opening menu selects this function.

You will be prompted for printer/screen options. It is recommendable to make a paper copy of the analysis for use with functions I and II. You will get a 3 page printout for single-sided disks. Use the Smart Keys to select the type of printer and the type of page end (<FF> or wait). Note that form-feed on the ADAM printer will send multiple line feeds. Consult the printer setup options at the end of this manual. The printout will list each block starting at 2. Consult the next page for explanations of the analysis. The following partial list will show its use:

```
002  WP file
003  ASCII
004  ASCII
005  Fast Load Program REM POKER/REM deal hand/REM evaluate/
006  Binary file REM players move/
007  Binary file REM POKER/
008  Binary file at 30000
009  Fast Load Program
010  Binary file
011  ASCII
012  ASCII REM move up/REM move down/
013  ASCII REM
014  ASCII REM calculate/REM jump/
015  ASCII
016  Shape Table at 29000 with 36 shapes
017  Binary file
018  Binary file
019  Fast Load Program
```

A considerable amount of information is available even from this printout. We know, for example that the first file is probably a word processing file with a maximum length of 3K. It is possible that it is only 1 or 2 K and is followed by a short BASIC program or an ASCII data file. We know for sure that there is and 'H' type BASIC program starting at block 5, and it is a POKER program. We also see the last REM statement (actually the start of the program) in block 7 marking the end of the program in that block. There is some sort of 'H' data file at block 8 and it is definitely 1K long loading at 30000. We are sure because there is a fast load program in the next block (note that FAST LOAD is the most reliable interpretation). The Fast Load Program in block 9 appears to be 1 or 2 K long; we forgot to put REM statements in that one. From 11 to 15 is hard to tell, there may be several files or programs but we are sure that there is at least one program because of the REM statements. The rest is easy since this section is all ASCII -- we'll use READ BLOCK to decode the text. From block 16 to probably 18, we have a shape table. Note that there is no load address shown at 17 or 18, so those blocks did not contain binary file headers -- the file must be 3K long.

MEDIUM ANALYSIS INFORMATION

A WP File

If the first three bytes of a block are 00 01 00 then the file is presumed to be in SMARTWRITER format. This can be verified by looking at that corresponding block of the disk: the first ASCII byte of the file will be in sector 2 byte 3 (one page header plus 3 bytes of pre-header).

B Binary File

If the first three bytes of a block are 01 00 01 then the file is presumed to be another 'type H' and the following checks are performed. Note that Binary is also the default file type. Binary file by itself only means that no other data type was recognized. Note also that the analysis has no way of determining the END-OF-FILE. You will have to do that manually.

B1 Fast Load Program

This is an 'H' type program created by TURBOLOAD, FAST LOAD, CRUNCH, and other similar utilities. It is easily recognized by looking for the variable commands (SPC TAB ERRNUM) in the first block of the binary file.

B2 Shape Table at aaaaa with nn shapes

If further analysis suggest data consistent with shape table definitions, the above line will be shown with the load address and the presumed number of shapes. Note that there are a variety of binary data files which would be interpreted as shape tables: 1,0,1,1,2,3,4,5,6,7,8,9,10... for example would be incorrectly interpreted as a shape table at 513 with 3 shapes. It would be evident from the load address that this information is erroneous. If you are dealing with a disk of your own programs, you will probably remember things like shape table sizes and load addresses and quickly spot the correct entries. Despite its limitations, the interpretation is correct most of the time.

B3 Binary file at aaaaa

If the analysis does not reveal a fast load program or a shape table, then the load address only is shown. Note again that this interpretation is not totally reliable. The load address is shown only in the first block of binary files. A binary data file containing many ones and zeroes might confuse the analysis.

C ASCII

Whenever the first 40 bytes of a block are composed of ASCII characters including <CR> <FF> ^D, this analysis is reported. It will happen under 4 conditions:

- -'A' file type with ASCII data or ASCII BASIC program
- -Subsequent blocks of a Smartwriter file
- -Long ASCII data imbeded in binary data file
- -Long REM, DATA, or PRINT statements in Fast Load programs

Note the first 2 occurrences are most common and can usually be trusted.

The ASCII file check can fail in 3 instances:

- -Fewer than 40 bytes used in last block of file
- -Imbeded control characters in PRINT or REM statements
- -Special characters in WP files like super/sub script

CHANGE DEFAULT SETUP

Pressing Smart Key IV from the main menu allows you to change display colours and the default drive for operations. From the setup menu, press the appropriate Smart Keys and see the colours change as you go along. You can set the border, character, and cursor colours.

The default setup also allows you to change the source drive and specification. Note that these changes will be in effect whenever you re-boot Disk Doctor; it is therefore a good idea to set the drive type to the configuration of your system.

To make changes that will affect only the current session, press ESCAPE after your selections. For permanent changes, press <CR>: You will be prompted for a destination drive. Check what drive your Disk Doctor system disk is in and press the appropriate SMart Key.

DIRECTORY STRUCTURE

Before using Disk Doctor, it is a good idea to be familiar with the EOS directory structure. Whether using tapes or disks, the EOS treats the media as a sequential device. This means that all 'files' are written to contiguous blocks. While this approach does not take full advantage of available disk space, it reduces the amount of directory information required.

Each directory entry is 26 bytes long, which allows a total of 39 entries per 'K' of directory. Both the EOS and DISK DOCTOR automatically handle the few extra bytes at the end of each block for Directories of more than 1K. Each file entry has the following format:

```
0                   1                   2
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
|    file name     | | | |       |   |   |   |     |
                    ^ ^ ^    ^     ^   ^   ^    ^
     file type   - -/ | |    |     |   |   |    |
     end-of-name - - -/ |    |     |   |   |    |
     attributes  - - - -/    |     |   |   |    |
     start block - - - - - - /     |   |   |    |
     reserved size - - - - - - - - /   |   |    |
     used size - - - - - - - - - - - - /   |    |
     used in last block  - - - - - - - - - /    |
     date  - - - - - - - - - - - - - - - - - - -/
```

The file name is limited to 10 characters. The program will let you enter ILLEGAL characters as file names, check your user manual for legal characters.

The file name is followed with the type; its maximum position (10) is shown above. Disk Doctor will correctly place the type. Enter a name and press <CR>, the cursor moves over and waits for a file type byte (a A h H), or any other character. Again, you are allowed to use special file types.

The file type is followed by a CHR$(3) which is the signal to the EOS that the end-of-name has been reached. The file type is located just before the end-of-name. All data from the end-of-name to position 11 is ignored and does not need to be blanked out.

The attributes byte is a series of bits that identify access to the file:

BIT	Meaning
7--LOCKED:	This bit is affected by BASIC's LOCK and UNLOCK functions which prevent a file from being deleted. Using EDIT CATALOG, a file can be deleted whether locked or not by setting the deleted bit.
6--WRITE PROTECT:	This bit prevents appending or deleting a file This bit cannot be set from BASIC.
5--READ PROTECT:	This bit prevents the OPENing, READing CLOSEing, and LOADing, of files from BASIC or SmartWriter. It can be useful for protecting programs or data that will be loaded in via READ BLOCK.
4--USER FILE:	This bit must be set if the file is to be shown by the CATALOG command. It can however be overridden by setting bit 3. Regardless, setting the user bit allows normal opening, closing, renaming, etc.
3--SYSTEM FILE:	Setting this bit disables the listing of the file by normal DIRECTORY or CATALOG functions. It has no other effect on file operations.
2--DELETED FILE:	A file may be un-deleted by resetting this bit. Make sure, however, that the same file name has not been used elsewhere by an active file, or you may confuse the EOS.
1--EXECUTE PROTECT:	This bit prevents UNLOCKing a file but will not prevent LOCKing. Any file LOCKed while this bit is set will not be UNLOCKable using conventional means.
0--BLOCKS LEFT BIT:	If this bit is set, all other information is ignored along with all directory entries following it. See BLOCKS LEFT ENTRY for more information.

The START BLOCK bytes identify the start of the file on the media. It uses 4 bytes which are placed in registers BCDE to address, in theory, a drive of over 4000 MEG.

The RESERVED SIZE bytes show the size of the file originally placed in the HOLE. A smaller file may reside in the same hole later, but it will always occupy the same amount of disk space. A reserved size may be up to 65535K since 2 bytes are used for this data. Note that START BLOCK + RESERVED SIZE must equal the START BLOCK of the next directory entry for proper management.

The USED SIZE bytes show the actual length of the file. When a small file is placed in a big HOLE, these bytes tell the EOS how many K of file to load in when getting the file.

The USED IN LAST BLOCK bytes are needed since the EOS does not use an END-OF- FILE marker in ASCII files. When a file is saved, its exact length is computed and the 'remainder' placed in these 2 bytes. The EOS will know when re-loading the file, exactly how many bytes of the last block to actually read in. All information past the USED IN LAST BLOCK pointer will be ignored.

The last 3 bytes of a directory entry are reserved for a creation date. There is a date in the EOS which appears to be the birth date of one of the programmers in 3 BC numbers YY-MM-DD. There is no function in BASIC, SmartWriter, SmartFiler, or any other COLECO software that I know of that makes use of this date. You may reset the system date by poking directly in the EOS and all files created today will have today's date. Furthermore, jumping to Smartwriter with a JP_WP instruction will also preserve the system date.

SPECIAL DIRECTORY ENTRIES

There are four special entries which look like just like file entries but are used by the EOS to work with the directory and MEDIUM.

VOLUME

The Volume entry is set by default to FIRST DIR. This is the name that will be reported by BASIC when a CATALOG command is issued. It is also the name passed by the INIT function in BASIC. Since the volume name does not need a file type, this entry may be up to 11 characters. The attribute byte consists of 80H + the number of K reserved for DIRECTORY (see DIRECTORY). This value may be set by placing a 'Y' in the PERMANENT PROTECT bit and setting the appropriate number of low-end bits (usually 1). The Next 4 bytes are 55H AAH 00H FFH; these are simply a series of check bytes for a valid directory and Edit Catalog will report this as a start block of 43605. Bytes 17 and 18 reflect the size of the medium, 255 for tapes, 160 for disks. The other bytes are non-significant and usually zero except for the date which may or may not be filled in.

BOOT

This entry tells the directory how many K have been reserved for a BOOT block. It points by default to Block 0 with a length of 1K and 1024 bytes used. The exact use of this entry is unclear since all EOS media have this entry filled out in the same way. SMARTBASIC and other systems reserve another entry in the directory for a BASICPGM file which is read and loaded from the BOOT block. The entry should not, however, be modified.

DIRECTORY

This entry tells the EOS the size of the directory by the 'size used' information. All other data in this entry appears to be insignificant but should not be changed, just in case. Note that the VOLUME entry also has a directory size, but it only indicates the maximum directory size. To change a directory from 1 to 2 K, both these entries must be modified.

BLOCKS LEFT

The Blocks left entry indicates the end of the disk or tape. Although it is handled slightly differently by BASIC and BASIC II, it only requires 3 parameters:

Setting the SYSTEM FILE and BLOCKS LEFT bit in the attributes regardless of the file name. BASIC II does not even bother writing BLOCKS LEFT in the name.

Setting the start block as for any other entry as the sum of previous-file-start and previous-file-size.

Setting the reserved-size by subtracting the start block from DISK SIZE. If Blocks Left starts at 100, the size should be set to 60 for a single sided disk (not 59).

HOW TO USE DISK DOCTOR

1 Start with a medium that has a bad directory: one that results in an I/O error when you try to get a CATALOG from BASIC or one that Smart Writer refuses to read.

2 Boot Disk Doctor and select EDIT CATALOG (Function I).

2a If the Directory reads in OK, then you probably just had a bad block read or other erratic problem. Press Smart Key II (change attributes) and simply press <CR> to change nothing. This makes Disk Doctor think that you have made a change. Now Press <CR> again to exit and answer 'Y' to the save question. Sometimes this is all that is needed. Try to read the disk again from BASIC or WP.

2b If the Directory Reads OK and some of the information is damaged, then a few minor adjustments may be required. Check VOLUME BOOT DIRECTORY and BLOCKS LEFT. Damage to some of these entries may invalidate the directory.

2c If the first block is not recognized as a directory, you will be asked whether or not to INIT the medium. Always answer 'N' to this question the first time through. You will be placed in EDIT CATALOG mode with whatever information was read in. Proceed as in 2b.

2d If there is no valid directory information, press <ESC> and select READ BLOCK. Scan through the first few blocks to check for valid 2 or more blocks of directory, for an EOS disk, and any other valuable information about the medium before you erase what is there.

2e Re-select EDIT CATALOG. This time answer 'Y' to the INIT question. Unless you are absolutely sure, you should always INIT a 1K Directory. You may indeed have a 2K Directory of which only the first K is damaged. After the INIT, proceed to ANALYZE MEDIUM.

3 In any case where Directory information is missing, ANALYZE MEDIUM (function III) may help you rebuild. The first step is to select printer options; it is well worth the wait as the information supplied will be invaluable later on.

4 Once the VOLUME BOOT DIRECTORY entries have been properly set, you can proceed to rebuild. Start at the directory entry and set the reserved_length to the correct directory size. Then press the down arrow key to check the first file entry.

4a Press WILD-CARD and switch to READ BLOCK at the start of the file. Go through the file (press HOME-DOWN) and set the cursor where you think the end of the file is. Then Press WILD-CARD again; answer 'Y' to the save question. This automatically calculates the file length and the size used in the last block.

4b Fill in the presumed file name, file type, and attribute. In most cases just setting the USER FILE bit to YES and the others to NO will be sufficient.

4c Repeat the above procedure for all files, assigning a unique name to each file including dummy names for those files that you can't figure out. If you get to a block that is full of E5's, you have certainly reached the end of the used space on the medium. E5 is the value written to every block of disks by the Format Program. Tapes on the other hand are filled with zeroes. Remember also that the last several blocks of the medium may contain invalid or incomplete files, especially if you have KRUNCHED the directory and re-arranged the files on the medium to recover space.

5 Review all your work, especially if you have changed any reserved_length bytes or if you have used the go-to-entry (HOME) function. The automatic recalculation of the directory is done as you scroll through the list. Check the start block of each file to make sure it is still properly set by consulting the MEDIUM ANALYSIS information or by pressing WILD_CARD to see the block. Don't forget to press <CR> from the Edit Catalog screen to save the changes.

6 Go to Smart Writer and load in all the 'A' files and the 'H' word processing files (you can't load the other 'H' files in WP). Check the end of the files to see if they have been correctly placed. If there is extraneous information at the end, simply remove it and re-save the file, preferably to another medium. You don't want to mess with your original medium until you have recovered as many files as possible.

7 Boot BASIC and start working with the FAST LOAD PROGRAMS if any. It is usually better to abort the programs as soon as they start running with a ^C. List the programs to see if they appear to be all there, and run them to see if they work correctly. If the program LISTs ok but does not work, save it in ASCII format (on another medium), re-boot BASIC and try the ASCII version. There may have been some extraneous bytes at the end of the file which trashed your BASIC.

HINTS AND SUGGESTIONS

There are some special situations which may run you into some difficulty. I will try to outline some of the scenarios, but it would be difficult to imagine all the possibilities. After you have lost a disk due to some error or hardware failure, any amount of information which can be recovered is a bonus. Don't expect to get everything back.

My Disk information seems to be there but I am unable to write to the directory block. This might be caused by a spike which has destroyed that block of the disk. Use a selective media copy utility like FILEMANAGER to copy from block 2 to the end of the media to another disk, INIT the other disk, and proceed to analyze and rebuild a directory for the new media.

FAST LOAD PROGRAMS crash into FATAL SYSTEM ERROR. This is usually caused by improper setting of file size. As a general rule, setting the file size bigger than it should be is not a major problem as it will overwrite parts of the bottom of the stack when loaded in. Setting the file size too low or much too high is disastrous. Adjust your file size up or down by about 1 sector (128 bytes) until you can at least get the program to load. Then, save an ASCII version of the program before testing it.

I have reached entry 39 (the last one) in my directory and still have more files to recover. This may happen if you have used a CRUNCH program to repack directory entries and recover unused disk space. The files that you are now trying to read may either be incomplete or older versions of programs which have been updated. The other possibility is the small holes left by frequent directory usage. You may also have created dummy entries which as far as you are concerned are garbage but were needed to keep the directory intact (see reserved-size used-size). Make that garbage part of the previous entry by setting the reserved length to the sum of the 2 entries and setting the used length to the length of the first file.

My ASCII BASIC program is missing some lines. If you are missing lines at the end, you have likely set your length too short. If the missing lines seem erratic, the file length may be too long. Re-load the file from SmartWriter and read it through again. The line numbers should be sequential. If a line number is followed by a smaller line number, the new line will replace the old one. In some cases this may not be a problem, but it is often disastrous if the next line happens to be a leftover from another program that used to be stored in that block.

My Program loads ok and lists ok but does not work after loading a binary data file. This is almost always due to resetting a data file's length greater than it was before. Check your LOMEM or HIMEM settings along with the load address and length of the file. Overwriting the LOMEM setting almost always results in destroying variable commands. Try a PRINT FRE(0) or PRINT TAB(10);"?". If these don't work, you have written to a RAM area below the LOMEM setting.

HINTS AND SUGGESTIONS (continued)

I can't figure out which data file belongs with which program. Sorry, I can't help you there. Neither can I tell you which version of a program is the most recent one. Note that repeated SAVEs of a BASIC program will create 3 files, an 'A' file, an 'a' file, and a deleted 'a' file. You may get even more each time the file size increased by 1K. Some of the following suggestions may help you avoid problems in the future:

Put a REM statement at line 0 of your programs (both ASCII and FAST LOAD). This one will include program name, version number, and creation date. If you change the version number each time you save the drafts of the program, each file will be unique.

Put a REM statement at line 65535 of your program, again with the program name. When you run across this line in READ BLOCK mode, you will know exactly where your program ends. Be sure to use the first occurrence of the REM statement. Note that FAST LOAD programs are stored top down and that your REM statement at the beginning of the program will mark the end of the file.

Waste a few bytes in DATA files to write the name of the program that they belong to or the function of the file. even if it is a Machine Language Code file, add a few ASCII data bytes after the RET instruction at the end. Programs that routinely write to data files should start by writing the name of the file and also write a visible end-of-file marker.

Word Processing files present less of a problem, especially if they are letters, reports, etc. It should be very obvious when scanning through the file where the end is. Do not look at any ASCII data in sector 0 and 1 of the first block of a WP file. All that area is reserved for headers, margins, tabs, etc. The file starts at byte 3 of sector 2.

This may seem to be a lot of work, but why not protect yourself. If you go through the trouble of writing a program, it must be because you want to keep it. It is also a good idea to keep a WORK DISK on which you have only prototypes of programs. Once they are debugged, they can be moved to another medium. Using this approach will cut down on your losses if one of your prototypes decides to start writing to disk at random.

PRINTER SETUP

Disk Doctor, like FILEMANAGER, is configured to be compatible with other printers hooked up through the EVE, CENTRONICS, or other similar interface. There are default sequences for horizontal pitch and letter quality; if they don't work with your printer, use a block editor to patch in the changes on your system disk.

Look at block 8, you should see the following data

PICA->	10 pitch sequence
ELITE->	12 pitch sequence
CONDON->	16.5 or condensed setting
NLQOFF->	Draft mode
NLQON->	Near Letter Quality mode

Each of these pointers denotes the start of the control sequences sent to the printer for the functions indicated. The first byte is the length of the sequence (maximum of 7) and it is followed by the control/escape sequence. Patch in the changes as required and save them to your system disk.

Note that all sequences are padded with nulls for additional safety. The routine that sends the sequences exits when it encounters a null or when the length of sequence is reached. For full compatibility, all sequences should be preceded with a correct length byte.

AJM SOFTWARE - REGISTRATION AND EVALUATION

NAME ______________________

ADDRESS ______________________________________

CITY ______________________ STATE/PROV _________ ZIP/CODE __________

NAME OF PROGRAM ______________________ REVISION ___________ SERIAL ____________

PURCHASED FROM ______________________________________

After you have used your new software item, please take a few moments to answer the following questions. Your comments may be valuable in the preparation of upgrades to the program, or in the design of new software items. If you take the time to register your software, you will be advised of upgrades, and problems that may arise from the use of the software. The program name and revision date should appear on the boot screen, and the serial number should be on the original disk label.

PLEASE RATE THE FOLLOWING ITEMS FROM **0** TO **5** WHERE **0** IS WORSE AND **5** IS BEST

	RATING	COMMENTS
WRITTEN INSTRUCTIONS	___	______________________________
		______________________________
ON SCREEN INSTRUCTIONS*(menus)*	___	______________________________
		______________________________
USE OF COLOUR	___	______________________________
		______________________________
USE OF SOUND*(if any)*	___	______________________________
		______________________________
IS THE PROGRAM SMOOTH FLOWING?	___	______________________________
		______________________________
IS THE PROGRAM EASY TO USE?	___	______________________________
		______________________________
DOES IT SERVE YOUR PURPOSE?	___	______________________________
		______________________________
WHAT ELSE WOULD YOU LIKE IT TO DO?		______________________________
		______________________________

____

____

____

____

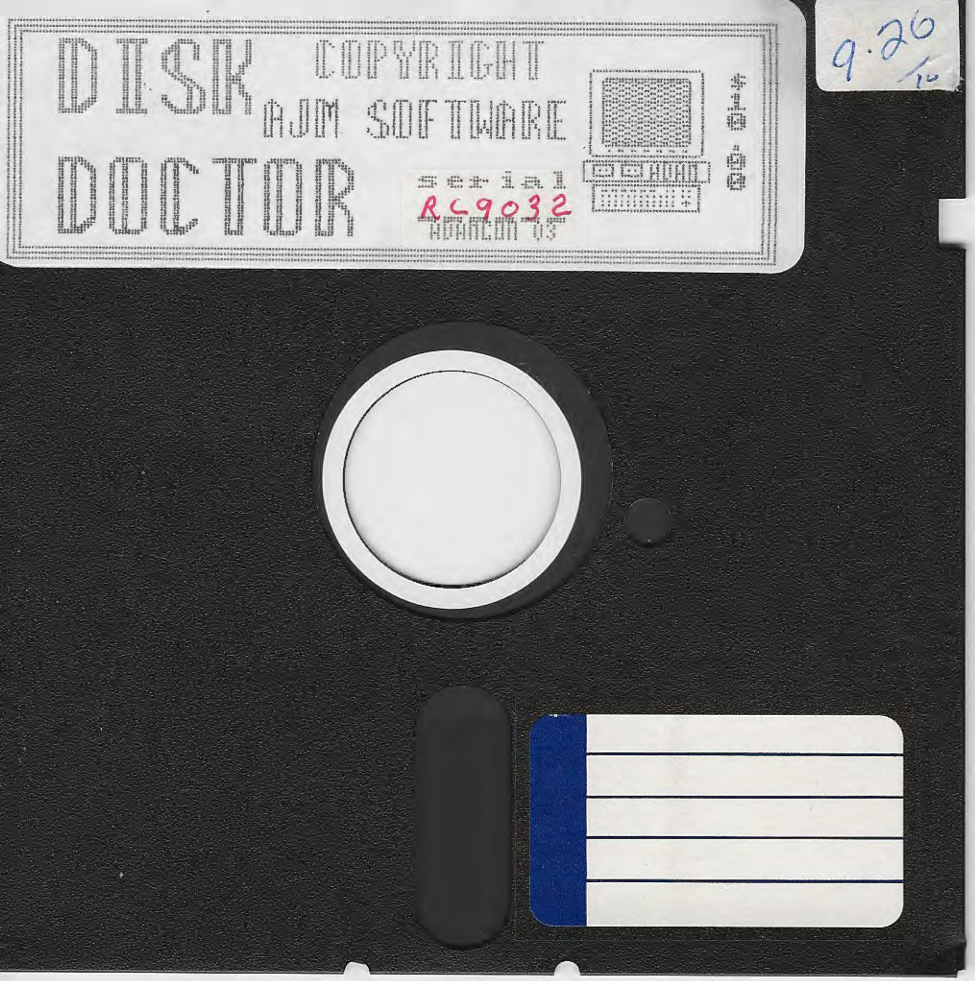
____

____

____

SEND YOUR REGISTRATION/EVALUATION TO:

Guy Cousineau
1059 Hindley Street
OTTAWA Canada
K2B 5L9



DISK DOCTOR

COPYRIGHT
AJM SOFTWARE

serial
RC9032
ADANCOM 03

$10.00

9.26
1/10