POWERSOFT

Products from Breeze/QSD, Inc.

THE TOOLBOX

The TOOLBOX for LDOS

3

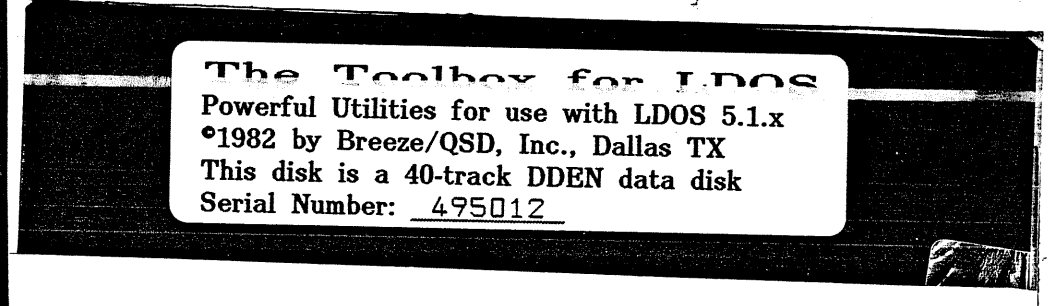
The information on this sheet supersedes that on page 5 of the Toolbox manual under the heading "Loading Instructions."

The disk you have received is a standard 40-track, double density data diskette formatted by LDOS. It contains all of the Toolbox utilities as described in the user's manual.

You are urged to make at least two backups of the distribution disk for normal, day to day use. Do not use the distribution disk in your day to day operations. Keep it in a safe place as a hedge against that inevitable day when you find that all your working copies have been bombed.

If your computer does not have double-density capability, send this disk back along with $5.00 to PowerSOFT, at the address found on page 5 of the manual and request that the programs be placed on single-density diskettes. You will receive back two single-density diskettes containing all of the programs.

The Toolbox utilities will work on LDOS versions 5.1.3 and later. The file called PBOOTM/FIX is for MAX-80 LDOS 5.1.4 only. Do not attempt to use these programs under any other operating system, including TRSDOS 6.x. If you are running TRSDOS 6.x exclusively, order the Toolbelt from PowerSOFT.



PowerSOFT's TOOLBOX FOR LDOS(tm)

OVERVIEW

The two disks you have received contain the utility programs which make up PowerSOFT's Toolbox for LDOS operating systems versions 5.1.3 and later. These are a powerful set of utilities which enable you to do a variety of tasks on disk files and memory, such as direct modification of disk sectors (PMOD), checking and fixing directory errors (PCHECK and PFIX), testing your disks for CRC and other formatting errors (PVU), reformatting your LDOS disks without losing any data already there (PREFORM), searching for a particular string, byte or word value in a file or in an entire disk (PFIND), and comparing two files or disks for differences (PCOMPARE).

In addition, PFILT/FLT is a general-purpose filter program which is user-definable, and which adapts itself for either input or output, depending on the device to which it is attached. The filter files may be created using the BUILD library command. Two example files, CODE/JCL and DECODE/JCL, are given to demonstrate how PFILT can be used.

Have you ever wanted to use the DVORAK keyboard layout everyone is talking about? DVORAK/FLT will remap your keyboard into that layout. Alternatively, you may make use of PFILT/FLT to achieve the same results using the filter file DVORAK/JCL.

Disk drive head movement may be tested and exercised with PEX, the all-purpose disk stepper exerciser. You may use this program in conjunction with a cleaning disk for more effective head cleaning (but use it sparingly!).

Tired of using a bulk eraser you don't dare bring close to the computer? PERASE will do the job for you. It will absolutely remove all traces of data and formatting information from a diskette.

If you just want to kill files from a diskette as fast as possible, the versatile PKILL utility will let you do so, in a variety of ways. You may specify files to be killed by class, category, or else provide a list. Much more versatile that the PURGE library command, and much faster.

If you need to know where on a diskette a certain file resides, PMAP will tell you. Conversely, if you want to know what file a certain sector on the disk has been assigned to, if any, PSS, the Sector Status utility, will provide the information.

PMOVE will allow you to move several files from one disk to another at the fastest possible speed. You may specify a list of files to move. This utility fills in the gap between a

Backup-by-class and a full Backup -- that is, the case when you have several files which don't have any common characteristics in their filenames that can be specified for a Backup-by-class.

You've just received a TRSDOS 1.3 disk that you want to CONVert over to LDOS, but you don't know exactly what's on it. Why bother booting it up to do a DIR, when PDIRT will show you the directory without having to leave LDOS at all?

Model I users know that a single-density LDOS disk can't be read by TRSDOS. This can be an annoying problem sometimes, especially when you want to send a diskette to someone who doesn't have LDOS. PUN, the UNREPAIR utility, will take care of that problem in a few seconds. It will write out the directory track of a single density Model I disk with the correct data address marks that will make it readable by TRSDOS. Now you don't have to bother installing patches in the operating system itself just to reverse the action of the REPAIR utility. Just UNREPAIR the disk!

Passwords can be removed from one or all files on a disk in one fell swoop by the PASSGO utility. Very handy for those cases when you don't want to keep typing the LDOS master password after each file. And this one won't even ask you for a master password!

Directories and unassigned sectors on your disk can be cleaned up with the use of PCLEAR. This utility will zero out the unused directory slots and unused sectors if desired. If you want to make sure absolutely no trace of a killed file remains on a disk for a hacker to find, use PCLEAR.

PHELP provides you with on-line help for the LDOS library commands and utilities, so you don't have to keep the manual nearby at all times. Just say "PHELP DIR" or "PHELP COPY" and it will tell all.

For those of you who use an EPSON-MX-80 printer, you'll have noticed that trying to perform a screen dump of graphics can do very strange things to your printer. This is because the MX-80's graphics codes aren't the same ones used by the TRS-80. PMX/FLT will solve that problem for you. Applied to the *PR device it will massage all graphics codes so that the MX-80 can understand them.

Tired of looking at the big LDOS logo on bootup? Want to personalize your disks? Apply the PBOOT patch and give your disks a new logo with your own 3-line personalized message. If we do say so ourselves, this logo looks a lot better.

The following pages will give full detailed instructions on the use of each of these utilities and filters. Please read the instructions carefully before trying anything drastic.

see change on pg 1.

LOADING INSTRUCTIONS

The disks you have received do not have an operating system on them. They are readable as ordinary LDOS data disks. However, they also contain a special transfer system which will move the files onto your own disks. THE MASTER DISKS CANNOT BE BACKED UP!! May we suggest that you move the files onto your own formatted diskettes and use those as your everyday working master. Put the original master disks away in a safe place place against that inevitable time when all your working disks are bombed out.

To use the special transfer system, simply prepare two single density data disks to receive the files. Then boot up each master in drive 0. You will be shown which files are on the disk, and how much space is needed on the destination drive to hold them. Place one of the disks you just prepared in any drive and tell the program which drive it is on. It will transfer the files over one by one. Repeat the process with the second master disk. Then put the masters away in a safe place.

REMEMBER! The destination disks must be formatted in single density!!

If you have only one drive, you may specify drive 0 as the destination disk and the file transfer system will tell you when to swap disks.

If you do NOT wish to use the transfer system, merely use the COPY command to move the files from the master onto your own disks. Or you may use BACKUP with the (VIS) parameter. Please don't specify (INV) or BACKUP will choke.

USE WITH LDOS VERSIONS EARLIER THAN 5.1.3

The TOOLBOX utilities are written for use with LDOS version 5.1.3. Under no circumstances should they be used with the LDOS 5.0 series. PowerSOFT will not take responsibility for damage arising from using these programs under LDOS 5.0.x.

Model I owners may use these programs on any 5.1 series LDOS system. Model III owners must apply corrective patches if they wish to use these programs under versions of LDOS earlier than 5.1.3. These patches are listed below.

```
PATCH PMOD/CMD (D00,6A=90)
PATCH PCHECK/CMD (D01,03=90)
PATCH PFIX/CMD (D01,69=90)
PATCH PFIND/CMD (D07,07=90)
PATCH PCOMPARE/CMD (D02,18=90)
PATCH PREFORM/CMD (D00,C6=90)
PATCH PVU/CMD (D00,CA=90)
PATCH PERASE/CMD (D00,99=90)
PATCH PCLEAR/CMD (D01,CE=90)
PATCH PSS/CMD (D01,2A=90)
PATCH PMAP/CMD (D00,E7=90)
PATCH PMOVE/CMD (D02,83=90:D02,A8=90)
PATCH PASSGO/CMD (D01,45=90)
PATCH PEX/CMD (D00,61=90)
PATCH PKILL/CMD (D01,C3=90)
```

Any file which does not appear on this list does not need patching. Remember, these patches are for MODEL III only.

PMOD

The PMOD Utility is a comprehensive Disk/File/Memory Modification Utility. It allows easy user interaction between memory and disk storage systems. All LDOS supported disk devices may be operated on, including Single/Double Density, Single/Double Sided, 5" and 8" floppies and fixed/removable rigid disk systems. It has a full screen display of 256 bytes at a time with both HEX and ASCII formats, and makes use of dual cursors for easy modification. The user can go to the heart of a disk and modify in HEX, ASCII, DECIMAL, BINARY, or OCTAL input. The syntax of the PMOD command is:

```
PMOD,aaa
PMOD,filespec,bbbb
PMOD,:c,d,e

aaaa = memory address to modify
filespec = any valid LDOS filespec
bbbb = relative file sector (optional)
:c = drive to modify (0-7, colon mandatory)
d  = cylinder number (defaults 0)
e  - sector number (defaults 0)

abbr: NONE
```

NOTE: It is recommended that you SET *KI KI/DVR (jkl) before using entering PMOD so the control keys and screenprinter will be active.

There are 3 modes of operation of the program as indicated by the command line:

Memory Modify

Enter the memory address where modification is to begin. You may enter the number in Hexadecimal, Decimal, Octal, or Binary by PRECEDING the number with H, D, O(or Q), or B respectively (for example, H4030). Decimal is the default value if none of the above are specified.

After entering the address and pressing <ENTER>, you will view the contents of 1 page of memory (256 bytes) starting at the specified address. Please refer to the following screen dump:

```
     3000.C35E 32C3 9B32 C374 32C3 DA32 C3C0 31C3..^2..2.t2..2..1.
HEX  3010.D131 C3AB 34C3 5534 C3C2 35C3 FB35 C35A..1..4.U4..5..5.Z
MEM  3020.36C3 8036 C38E 33C3 3937 C3F7 31C3 7B37.6..6..3.97..1.{7
     3030.C399 37C3 BB35 C3A0 35DB E4CB 6FC3 1C35...7..5..5...o..5
     3040.18D3 C3B5 3740 6162 6364 6566 6768 696A.....7@abcdefghij
     3050.6B6C 6D6E 6F70 7172 7374 7576 7778 797A.klmnopqrstuvwxyz
     3060.3AEA 37B7 C930 3132 3334 3536 3738 393A.:.7..0123456789:
     3070.3B2C 2D2E 2F0D 1F01 5B0A 0809 2021 DC05.;,-./...[... !..
     3080.22FF 41AF C960 4142 4344 4546 4748 494A.".A..@ABCDEFGHIJ
     3090.4B4C 4D4E 4F50 5152 5354 5556 5758 595A.KLMNOPQRSTUVWXYZ
     30A0.77AF C9AA AA00 2122 2324 2526 2728 292A.w.....!"#$%&'()*
     30B0.2B3C 3D3E 3F0D 1F01 1B1A 1819 203E 0121.+<=>?....... >.!
     30C0.1940 AE18 DB40 4142 4344 4546 4748 494A..@...@ABCDEFGHIJ
     30D0.4B4C 4D4E 4F50 5152 5354 5556 5758 595A.KLMNOPQRSTUVWXYZ
     30E0.CDD9 01AF C930 3132 3334 3536 3738 393A......0123456789:
     30F0.3B2C 2D2E 2F0D 1F01 5B0A 0809 2028 E1A6.;,-./...[... (..
```

The far left column of the screen indicates the current modification BASE (HEX, DECimal, ASCii, OCTal, BINary), and the MEM indicates that you are in the memory modify mode. The next column is the 4 character HEX address where the data is that you are viewing. The center columns of the display is the HEX representation of the 16 bytes starting at the address indicated at the row beginning. There is a space between each quad (2 bytes, 4 HEX ASCII characters) for easier viewing. On the far right is the representative ASCII character of the 16 bytes on that row. All bytes less than 20H (32 decimal) are displayed as a period (.) to indicate a non-printing character. Graphics characters are also printed as periods in this illustration, however they will show up as graphics on your screen. You are now in the memory paging mode. Several keys are active at this point:

BREAK - terminate the program
SHIFT CLEAR - prompt for a new address
M - enter MODIFY mode
A - set ASCII modify
B - set BINARY modify
D - set DECIMAL modify
H - set HEXIDECIMAL modify
O - set OCTAL modify
X - prompt for new source data
RIGHT ARROW - increment displayed address by 1 byte
LEFT ARROW - decrement displayed address by 1 byte
UP ARROW - increment displayed address by 1 page (256 bytes)
DOWN ARROW - decrement displayed address by 1 page

Disk Modify

Enter a Drive, Cylinder, and Sector where modification is to begin. You must precede the drive number with a colon (:) to distinguish it from a memory address. The cylinder and sector numbers default to 0 if not issued. The cylinder may be answered with the at symbol (@) to page directly to the directory track.

After answering the prompt and pressing <ENTER> the specified sector will be read from disk and displayed on the video. Please refer to the following screen dump:

```
     00.0300 FDCB 035E 2803 210C 62ED B0FD CB03.....^(.!.b.....
HEX  10.5E20 353E 53FD CB03 7628 023E 0102 0057.^ 5>S...v(.>...W
DRV  20.4432 413F 3E44 3242 3F3E 53FD CB04 6E28.D2A?>D2B?>S...n(
 0   30.023E 4432 813F 3E53 3282 3FFD CB03 7EC8..>D2.?>S2.?...~.
CYL  40.210F 6211 C03F 0103 00ED B0C9 1140 3F01.!.b..?.......@?.
 11  50.0300 FDCB 0356 2192 6020 0321 9560 EDB0.....V!.@ .!.@..
SID  60.18D9 2104 3C11 4164 0140 003E 10F5 1A13...!.<.Ad.@.>....
 0   70.FE3A 2816 FE03 2812 FE0D 280E FE2E 280A..:(...(...(...(.
SEC  80.B728 0777 09F1 3D20 E4F5 F121 F461 1180..(.w..= ...!.a..
 05  90.3C01 0300 EDB0 DD21 5F64 DD7E 06C6 3032.<.....!_d.~..02
STD  A0.C13C 21BA 6111 003F 0104 00ED B0DD 213F..<!.a..?......!?
 5"  B0.3FDD 7E00 F52A C55E CDC9 54F1 323F 3F21.?.~..*.^..T.2??!
RIG  C0.BE61 1180 3F01 0400 EDB0 DD21 BF3F DD7E..a..?......!.?.~
FIX  D0.00F5 2A65 5FCD C954 F132 BF3F C921 F857...*e_..T.2.?.!.W
     E0.1180 3C01 0300 EDB0 2AC0 5511 4000 FD21...<.....*.U.@..!
     F0.033C 3E10 F57C CDB0 54FD 7100 FD70 017D..<>..|..T.q..p.}
```

The screen format is similar to Memory Modify except for the far left column of data, which indicates the source of the information. You are given the drive, cylinder, and sector being viewed. You are supplied with the size of the drive (5" or 8"), and if it is a Floppy or Rigid drive. If a floppy, you are given the density (single or double), and the number of sides (single or double). If a Rigid drive, you are told if it is Fixed or Removable. You are now in the sector paging mode. The following keys are active:

BREAK - terminate program
SHIFT CLEAR - prompt for new drive, cylinder, and Sector
M - enter MODIFY mode
A - set ASCII modify
B - set BINARY modify
D - set DECIMAL modify
H - set HEX modify
O - set OCTAL modify
X - prompt for new source data
0-9 - page to corresponding sector

@ - page to current sector on directory track
R - restore drive to cylinder 0, sector 0
RIGHT ARROW - increment sector
LEFT ARROW - decrement sector
UP ARROW - increment track
DOWN ARROW - decrement track
SHIFT RIGHT ARROW - page to highest sector on current track
SHIFT DOWN ARROW - page to lowest sector on current track
SHIFT UP ARROW - page to highest cylinder on disk, current sector

File Modify

Enter a filespec that you wish to view. You may optionally follow the filename with the relative sector that you wish to start with. After locating the file, the specified relative sector (default zero) will be read from the disk and displayed to the video. Please refer to the following screen dump:

```
    D00.0506 4445 4C20 2020 1F18 5665 7273 696F...DEL   ..Versio
HEX E10.6E20 3228 3130 2920 2D20 3039 2F32 382F.n 2(10) - 09/28/
DRV L20.3832 0102 0052 7EFE 0DCA B252 FE2F 2007.82...R~....R./ .
 0  /30.114E 5323 CD1C 44DD 214D 533A 4038 FE04..NS#..D.!MS:@8..
    C40.CADD 5211 5153 CD1C 4428 0AFE 0D20 ECCD...R.QS..D(... ..
    M50.E952 C32D 40CD 4252 CD24 44C2 BB52 CD2C..R.-@.BR.$D..R.,
    D60.4420 7EDD 3400 18D3 3A4E 53B7 C8E5 CD70.D ~.4...:NS....p
     70.5221 4E53 3A54 003D 2010 CD73 443A 4C53.RINS:T.= ..sD:LS
     80.B7C4 9052 E1AF 324C 53C9 CD4B 443A 4C53....R..2LS..KD:LS
     90.B7C4 9052 18EE 2151 537E FE3A 2806 FE03....R..!QS~.:(...
     A0.C823 18F5 3E01 324C 53D5 C511 4953 0102..#..>.2LS...IS..
     B0.00ED B0C1 D1C9 D511 5153 1AFE 0328 07FE.........QS...(..
RSEC C0.3A28 0313 18F4 E5C5 2149 5301 0300 EDB0.:(......!IS.....
0000 D0.AF32 4C53 C1E1 D1C9 2110 53CD 6744 C330..2LS....!.S.gD.0
EOFS E0.40E5 F53A 5153 CB7F 2803 CD28 4421 5153.@..:QS(..(D!QS
0002 F0.CD67 4421 2F53 CD67 44F1 F6C0 CD09 44E1..gD!/S.gD.....D.
```

The screen layout is similar to that of Memory Modify, except for the far left column. You are given the drive number that you are working on. The filename that you are viewing is displayed vertically starting just to the right of the cursor. Near the bottom you are given the relative sector you are viewing, and the end of file sector. Normally, you will only be able to view up to 1 less than the EOFS. You are now in the file paging mode. Several keys are active:

BREAK - terminate program
SHIFT CLEAR - prompt for a new relative sector
M - enter Modify mode

A - set ASCII modify
B - set BINARY modify
D - set DECIMAL modify
H - set HEX modify
O - set OCTAL modify
X - prompt for new source data
RIGHT ARROW - increment to the next relative sector
UP ARROW - same as right arrow
LEFT ARROW - decrement to next relative sector
DOWN ARROW - same as left arrow

NOTE: When using the FILE MODIFY option on DIR/SYS, the sectors will NOT be written as Read-Protected, making it unreadable to LDOS for all intents and purposes and requiring a "Repair (alien)" after updating back to the disk. Use the DISK MODIFY option to correctly handle the directory track.

Modify Mode

The MODIFY mode is common to all of the above 3 modes of operation, and is entered by pressing M when the desired data is being viewed in one of the above paging modes. The cursor will move into the HEX and ASCII portions of the display (dual cursors), and will flash very rapidly. Refer to the following screen dump:

```
00] D00.0506 4445 4C20 2020 1F18 5665 7273 696F...DEL   ..Versio
HEX E10.6E20 3228 3130 2920 2D20 3039 2F32 382F.n 2(10) - 09/28/
DRV L20.3832 0102 0052 7EFE 0DCA B252 FE2F 2007.82...R~....R./ .
 0  /30.114E 5323 CD1C 44DD 214D 533A 4038 FE04..NS#..D.!MS:@8..
    C40.CADD 5211 5153 CD1C 4428 0AFE 0D20 ECCD...R.QS..D(... ..
    M50.E952 C32D 40CD 4252 CD24 44C2 BB52 CD2C..R.-@.BR.$D..R.,
    D60.4420 7EDD 3400 18D3 3A4E 53B7 C8E5 CD70.D ~.4...:NS....p
     70.5221 4E53 3A54 003D 2010 CD73 443A 4C53.RINS:T.= ..sD:LS
     80.B7C4 9052 E1AF 324C 53C9 CD4B 443A 4C53....R..2LS..KD:LS
     90.B7C4 9052 18EE 2151 537E FE3A 2806 FE03....R..!QS~.:(...
     A0.C823 18F5 3E01 324C 53D5 C511 4953 0102..#..>.2LS...IS..
     B0.00ED B0C1 D1C9 D511 5153 1AFE 0328 07FE.........QS...(..
RSEC C0.3A28 0313 18F4 E5C5 2149 5301 0300 EDB0.:(......!IS.....
0000 D0.AF32 4C53 C1E1 D1C9 2110 53CD 6744 C330..2LS....!.S.gD.0
EOFS E0.40E5 F53A 5153 CB7F 2803 CD28 4421 5153.@..:QS(..(D!QS
0002 F0.CD67 4421 2F53 CD67 44F1 F6C0 CD09 44E1..gD!/S.gD.....D.
```

At the top left of the display, you are given the relative byte within the displayed data where the cursor lies. The following keys are active at this point:

BREAK - abort the modify mode, re-read the source data as it was before any changes were made, and return back to the paging mode you came from. NOTE: In the MEMORY mode, the changes are immediate, and cannot be re-read. In this case, the BREAK key is identical to pressing the ENTER key.

```
22  X00.01FE FC52 0321 7403 0102 00E0 B021 6AD3.A..R.!t.AB∂..!j
HEX D10.363F 2336 0023 3E02 32AF 402A 8AD3 ED5B.6?#6∂#>B2.∂*...[
DRV 120.6AD3 ..D2 0B22 7603 1182 0321 8AD3 0102.j...K"v.Q..!..A
    R30.00ED B011 82D3 1A6F 131A 677E 2172 D377.∂..Q..ZoSZg~!r.
    /40.2336 0021 6AD3 3600 2336 0023 2172 D35E.#6∂!j.6M#6∂#!r.
    C50.2356 216A D34E 2346 D5E1 B7ED 427C B5CA.#V!j.N#F....B¦.
    M60.AF53 2172 037E 2168 A536 0123 7721 68A7..S!r.~!h.6A#w!h
    D70.4E79 0600 0311 D8D6 EDB0 2168 A54E 8123.NyF∂CQ....!h.N.
     80.F579 B728 0406 00ED B021 D8D6 F177 4F06..y.(DF∂..!...wO
     90.0003 1168 A7ED B011 82D3 1A6F 131A 6723.∂CQh...Q..ZoSZg
     A0.7C12 1B7D 12E5 1176 D31A 6F13 1A67 D1AF.¦R[]R.Qv.ZoSZg.
     B0.ED52 CB7C CA2B 53C3 B353 2021 68B7 3601..R.¦.+S..S !h.6
RSEC C0.23E5 0121 B253 0E01 0600 EDB0 2100 0022.#..!.SNAF∂..!∂∂
0000 D0.DCD6 2168 B77E FE00 284D 22DA D621 68A7...!h.~.∂(M"..!h
0001 E0.4F7E 9138 4E4E 22D8 D606 0023 ED5B DAD6.O~.8NN"..F∂#.[.
     F0.131A EDB1 3E00 2031 2BC5 E5ED 5BD8 D61A.SZ..>∂ 1+...[..
```

At the top left of the display, you are given the relative byte within the displayed data where the cursor lies. The following keys are active at this point:

BREAK - abort the modify mode, re-read the source data as it was before any changes were made, and return back to the paging mode you came from. **NOTE:** In the **MEMORY** mode, the changes are immediate, and cannot be re-read. In this case, the **BREAK** key is identical to pressing the **ENTER** key.

ENTER - terminate the modify mode, and write the modified data back to the source with all changes, and return to the paging mode you came from. **NOTE:** Changes are immediate in the **MEMORY** mode, and control is merely passed back to the paging mode.

SHIFT CLEAR - allows you to change the modification mode BASE without leaving modify mode. You will have a new flashing prompt in the left column of the display. Enter A, B, D, H, or O (or Q) to set ASCII, BINARY, DECIMAL, HEX, or OCTAL bases respectively. Any other keys will be ignored. You will then return back to the modify mode at the same location you came from.

ARROW KEYS - move the cursor 1 byte in the corresponding direction without affecting any of the bytes.

SHIFT ARROW KEYS - move the cursor to extreme row or column ends in the associated direction without affecting any of the bytes.

NOTE: In the **DISK MODIFY** and **FILE MODIFY** modes you will not be able to move the cursor beyond the edges of the displayed data. In **MEMORY** modify, if you attempt to move the cursor beyond the edges, additional data is brought to the screen to supply the desired information.

If you are in the ASCII MODIFY mode, and the key you enter does not apply to any of the above conditions, then that character is entered at the current cursor location, and the cursor is advanced by one byte.

If you are NOT in ASCII modify, several additional features are available:

Q - position the cursor to the last byte of the currently displayed page.

S - position the cursor to the first byte of the currently displayed page.

G - followed by a number moves the cursor immediately to the relative byte in the page that you are viewing.

L - followed by a number moves the cursor to the next occurrence of the specified byte. The cursor will move to the last byte on the screen if the requested byte is not located.

+ - followed by a number positions the cursor that many bytes FORWARD from its current position.

- - followed by a number positions the cursor that many bytes BACK from its current position.

P - followed by a number duplicates (propagates) the current byte located under the cursor to the following indicated number of bytes.

< - will shift all bytes from the cursor to the page end by one byte, and place a zero at the last location on the screen. This allows you to delete text at the cursor.

> - will shift all bytes from the cursor to the page end by one byte, and place a zero at the current cursor location. This allows you to insert text at the cursor.

Z - fills the data buffer with ZEROES starting at the current cursor position.

The above commands are disabled in the ASCII MODIFY mode, and will merely insert the character into the text.

If you are not in ASCII modify, and none of the above conditions are met, then the input is interpreted as numeric input. If you are in HEX mode, for example, you must enter two HEX digits (0-9,A-F) to change a single byte. Decimal mode expects 3 digits (0-9), Octal mode expects 3 digits (0-7), and Binary mode expects 8 digits (0-1). You may pre-terminate the numerical input by pressing ENTER. Thus, in HEX mode, 3 <ENTER> is the same as entering 0 3, in Binary mode, 1 1 1 <ENTER> is the same as entering 0 0 0 0 0 1 1 1. When you are in the middle of entering a number,

the cursor will change to indicate that more input is required to complete the current operation. If you enter incorrect digits before completing the number, entering an invalid character will terminate the entry. Thus, in HEX modify, you type a 3, but meant to enter a 4, just hit an invalid key (X for example) to terminate the operation and leave the byte unchanged. In the above special commands, G, L, P, + and -, the numeric input following the command must be in the current base. If you are in HEX modify, then G10 is a valid command to move the cursor to relative byte 10H, but in DECIMAL modify, you must enter G016 to achieve the same results.

Any key that does not meet any of the above conditions will be ignored.

When first entering this program, the data source information may be entered directly from the LDOS™ READY command line, or will be prompted for if not supplied with the command. If you wish to examine a file that could be interpreted as an address (for example, AFCH/CMD or D0123/TXT), you may precede the filename with an exclamation point (!) to force PMOD to interpret it as a filespec. For example, PMOD,F000H will display memory address F000H, but PMOD,!F000H will display the file F000H. Alternatively, you may attach a drive specification to the file name to force it to be treated as such, for example, F000H:0 will be seen as the file F000H on drive zero rather than the address F000H.

PCHECK

The PCHECK Utility is a comprehensive directory check utility. It allows the user to completely check the integrity of disk storage systems. All LDOS™ supported disk devices may be operated on, including Single/Double Density, Single/Double Sided, 5" and 8" floppies and fixed/removable rigid disk systems. This program will locate and report any inconsistencies in the directory information of a disk. In conjunction with PFIX, most directory problems can be easily corrected without extensive knowledge of how directories are formatted. The syntax of the PCHECK command is:

```
        PCHECK, *,:a
        PCHECK,*,filespec

  :a = drive number to check (colon optional)
  filespec = any valid LDOS™ filespec

  an asterisk (*) preceding the parameters will
  send the output to the video and printer

  abbr: NONE
```

After selecting the mode of operation (full disk or file), the entire directory of the indicated disk is read into memory. If not enough memory is available, it will be reported, and the program will abort. If any errors occurring during the directory

read, they will be reported, and the program will pause and allow you to select a R>etry, S>kip, or A>bort. Although it is possible to check even an unreadable directory, unpredictable errors may be reported. For the normal operation of this program, it is assumed that all sectors of the directory are readable, even though the data contained therein may be incorrect.

Each file specified is passed through a three stage check. This will either be the single file specified, or all files if an entire disk is specified. In the first stage, the integrity of the files' directory entry is checked. In the second stage, the associated HIT table bytes are checked. In the third stage, the associated GAT table bytes are checked. When all specified files have been checked, and all errors have been reported, a total error count will be issued and the program will exit back to LDOS™.

The utility PFIX/CMD is capable of fixing most errors that may be reported by PCHECK.

The following is a list of the possible error messages that PCHECK may report. A technical section will follow that will describe the logic used by PCHECK, and probable causes for the error conditions.

ERROR MESSAGES

1. Cylinder xxx has an invalid GAT table byte
2. HIT byte at xxH invalid or extraneous
3. Filename contains non-ASCII characters
4. End of File Sector beyond allocated sectors
5. No terminator for extent field

6. Directory links to record not linking back to it
7. Track assigned that is beyond diskette boundary
8. Extension assigned before end of extents
9. Forward link to inactive entry
10. Forward link to non-extension entry
11. Extension record not assigned to any files
12. Multiple files assigned to single granule
13. Directory record has invalid HIT byte
14. Directory record has a zero HIT byte
15. Extended directory record has invalid HIT byte
16. Extended directory record has a zero HIT byte

Directory Check Logic

This section contains the logic that PCHECK uses to determine the correctness of the directory. The novice user need not be concerned with this information as it is offered only as technical reference for the advanced user.

During passes 1-3, if an error is found, it is reported in the following format:

filename/ext @ Directory Sector xx, Relative Byte xxH.
"error message string"

If a non-ASCII character is found in the filename, its position will be highlighted with a graphic block. In all errors reported in the above format, the directory location will refer to the primary directory record, regardless of where the error occurred along the file.

Pass #1

Check to be sure that all 11 bytes of the filename contain characters within the range of 20H to BFH (printable ASCII characters). If any are found to be outside this range, report error 3. The offending characters are displayed as a graphic block to aid in identifying their positions.

The 5 extents of the associated record are each checked on a sector by sector basis. If any extent points to a track that is beyond the diskette boundary, an error is reported.

If all 5 extents have been parsed, and no terminator is found (FEH or FFH) then error 5 is reported and the program terminates.

If an extended record is found, and it is not at the 5th position in the extent field, error 8 is reported and the error scan continues.

If the extended record is found, locate its position in the directory. If bit 4 of the first byte in the record is not set (inactive file), error 9 is reported and the program terminates. If bit 7 is not set (not an extension), error 10 is returned and the program terminates. The second byte in the extended record is a link back to the previous record. If this byte is incorrect, error 6 is reported and the program terminates.

This process continues until an FFH terminator is reached, or a terminal error is found.

When processing is completed, the number of sectors assigned to the file is calculated and compared to the end of file sector in the directory

record. If the end of file is larger than the allocated records, error 4 is reported.

Pass #2

This pass checks the integrity of the associated HIT byte for the current file. Each file is traced from its primary directory entry through all extents until an FFH terminator or a terminal error is reached.

If the primary directory entry contains an invalid HIT byte, PCHECK returns error 13 and continues. If the primary directory entry contains a zero HIT byte, error 14 is returned.

If an extended directory record contains an invalid HIT byte, error 15 is returned. If an extended record contains a zero HIT byte, then error 16 is reported.

Pass #3

This pass checks the integrity of the associated GAT bytes for the current file. As with pass 2, each file is traced through all of its extents.

If the current granule has already been allocated (either by itself or another file), then error 12 is displayed. NOTE: Only a single error 12 will be reported for a single file even though there may be several occurrences of it.

When these steps have all been completed, and a single file has been specified, the number of errors will be reported and an exit will be made

back to LDOS™. If an entire disk is specified, three additional checks are made on the directory.

Check #1

When Pass #1 is made in the above process, it checks every directory record that has bit 4 set and bit 7 NOT set (active primary directory record). As the current file is being checked, bit 5 of all the associated directory primary and extended records is set to indicate a completed file. During the Check #1 phase, the entire directory is re-scanned. If any files have bit 4 set, and not bit 5, then bit 7 MUST be set or error 11 is reported. This condition indicates an extended directory record that has not be linked to any primary record (an extraneous directory record). This condition could result if one of the above passes terminates on a fatal error and an extension is not checked. If no terminal type errors are reported, but error 11 was issued, then there is an extra record in the directory that is not associated with any files.

Check #2

As the program processes pass #2, it completely builds a second HIT table in another buffer. During the check #2 phase, the computed table is compared byte for byte with the HIT table as read from the diskette. If any of the bytes do not match, then error 2 along with the relative byte of the mismatch is displayed. If none of the pass #2 errors are reported, but error 2 is issued, then there is an extra byte in the HIT table at the corresponding location.

Check #3

As the program processes pass #3, it completely builds a second GAT table in another buffer. Before check 3 is made, the disk lockout table is overlaid onto the built GAT table so that locked out tracks will not be counted. During the check #3 phase, the computed table is compared byte for byte with the GAT table as read from the diskette for as many tracks as there are on the disk. If any of the bytes do not match, error 1 and the relative cylinder of the mismatch is returned. If none of the pass #3 errors are reported, but error 1 is issued, then there are extra grans allocated on the disk that are not assigned to any files.

Sometimes, one type of an error will inherently cause another type of error. In these cases, fixing even one error may correct several others. As an example, if there is no terminator for a record, chances are that error 12 will be reported also. If a terminal error is found in a directory link to an extension, error 2 may also be reported.

When using this utility on Rigid drives, there is one error that may be reported that should be considered normal. LDOS™ sets aside an extra track right next to the directory, and shows this track as allocated in the GAT table. This track is used by the Laredo and Radio Shack hard drive systems as overhead, and should not be used. PCHECK may report an invalid GAT Table Byte at this track location, and it should be considered normal. The Laredo or Radio Shack rigid drive user should be aware of this problem, especially as it relates to the PFIX utility, which may de-allocate this track, or allocate it to an unassigned file.

Similarly, using PCHECK on a Model I double density system disk with a SOLE track will result in an "Invalid GAT byte" error. This is because the SOLE modification system allocates track 0 to prevent the system from attempting to use non-existent sectors on it. This error should be considered normal and should not be fixed.

PFIX

The PFIX program is a comprehensive director repair utility. It allows the user to completel repair most directory problems, and also has th ability to transfer a faulty boot sector from a good disk. All LDOS™ supported disk devices ma be operated on, including Single/Double Density Single/Double Sided, 5" and 8" floppies and fixe or removable rigid disk systems. This program wil locate and repair most inconsistencies in th directory information of a disk. In conjunction wit PCHECK, most directory problems can be easil located and corrected without extensive knowledg of how directories are formatted. The syntax o the PFIX command is:

```
PFIX:a,G,H,F,B=:d,L,R,P

:a = drive number to fix (colon optional)
 G = repair GAT table
 H = repair HIT table
 F = repair missing/invalid File data
 B = repair BOOT sector
:d = take boot sector from this drive
     (mandatory if B is specified)
 L = indicates a LAREDO rigid disk
     system
 R = indicates a RADIO SHACK
     rigid disk
 P = prompt for cyls, density, sides
     (should be last parameter
     specified)
```



```
At least ONE parameter MUST be specified.

abbr: NONE
```

After selecting the desired parameters, the entire directory of the indicated disk is read into memory. If the P option is not specified, the program will use the information supplied by LDOS™ defining the diskette cylinder count, number of sides, and density. This information is located in the GAT sector of the diskette at format time. If it is possible that this particular information is invalid, use the P option, and you will be prompted to enter these parameters. If, for example, the entire GAT sector is zeroed out, LDOS™ will report this to be a "xxxxxx density, single sided, 35 track" disk. If the GAT is fixed using these parameters, and the disk is REALLY a 40 track disk, the corrections will be inaccurate.

If not enough memory is available, it will be reported, and the program will abort. If any errors occurring during the directory read, they will be reported, and the program will pause and allow you to select a R>etry, S>kip, or A>bort. Although it is possible to repair even an unreadable directory, unpredictable errors may be generated. For the normal operation of this program, it is assumed that all sectors of the directory are readable, even though the data in them may be incorrect.

Each file of the directory is passed through a three stage process. In the first stage, the integrity of the files directory entry is fixed. In the second stage, the associated HIT table bytes are fixed. In the third stage, the associated GAT

table bytes are fixed. When all specified files have been checked, and all errors have been corrected, the corrected data will be written back to disk per the command parameters. Either the GAT, HIT, file records, and/or BOOT sector will be updated to the disk. Although the entire directory is repaired in memory, only the requested data to be fixed is written back out to the disk. The utility **PFIX/CMD** is capable of fixing most errors that may be reported by **PCHECK**.

If only the GAT and/or HIT are specified, and an error is found in any of the files directory records, the fix will abort and a message will be returned requesting you to use the "F" option. This is because it would be unreliable to repair a GAT or HIT when the root information is invalid. This is to prevent a user from inadvertently destroying any relevant data.

The following is a list of the errors that **PFIX** can fix. A technical section will follow that will describe the logic used by **PFIX**, and probable causes for the error conditions.

1. Cylinder xxx has an invalid GAT table byte
2. HIT byte at xxH invalid or extraneous
3. Filename contains non-ASCII characters
4. End of File Sector beyond allocated sectors
5. No terminator for extent field
6. Directory links to record not linking back to it
7. Track assigned that is beyond diskette boundary
8. Extension assigned before end of extents
9. Forward link to inactive entry
10. Forward link to non-extension entry
11. Extension record not assigned to any files
12. Multiple files assigned to single granule

13. Directory record has invalid HIT byte
14. Directory record has a zero HIT byte
15. Extended directory record has invalid HIT byte
16. Extended directory record has a zero HIT byte

Directory Repair Logic

This section contains the logic that PFIX uses to repair the directory. The novice user need not be concerned with this information as it is offered only as technical reference for the advanced user.

Pass #1

Check to be sure that all 11 bytes of the filename contain characters within the range of 20H to BFH (printable ASCII characters). If any are found to be outside this range, replace with an "X". (If this occurs, a HIT repair will be required to access this file from LDOS™)

The 5 extents of the associated file record are each checked on a sector by sector basis. If any extent points to a track that is beyond the diskette boundary, a terminator (FFH) is entered at the current location.

If any granules are indicated in a file's extents, and the corresponding GAT bit is set (assigned to a previous file), the record will be terminated at the current location.

If all 5 extents have been parsed, and no terminator was found (FEH or FFH) then a terminator is entered at the last position.

If an extended record is found, and it is not at the 5th position in the extent field, the error is ignored and the program continues.

If the extended record is found, its position in the directory is located. If bit 4 of the first byte

in the record is not set (inactive file), the extension is unlinked. If bit 7 is not set (not an extension), it is also unlinked. The second byte in the extended record is a link back to the previous record. If this byte is incorrect, the extension is unlinked.

The un-linking of the files is to protect against possible overlaying another file that does not in fact link back to the primary. By unlinking a file, data past the link will be lost, but the data will not accidentally be force-linked to an incorrect entry.

This process continues until an FFH terminator is reached, or a terminal error is found and a terminator is forced.

When processing is completed, the number of sectors assigned to the file is calculated and compared to the end of file sector in the directory record. If the end of file is larger than the allocated records, then force the number of allocated records into the end of file sector.

Pass #2

This pass constructs the HIT table. Each file is traced from its primary directory entry through all extents until an FFH terminator is reached. The entire HIT table will be reconstructed starting with a page (256 bytes) of zeroes. All active primary and extended directory records are logged correctly into this table.

Pass #3

This pass constructs the GAT table. As with pass 2, each file is traced through all of its

allocated records, then force the number of allocated records into the end of file sector.

Pass #2

This pass constructs the HIT table. Each file is traced from its primary directory entry through all extents until an FFH terminator is reached. The entire HIT table will be reconstructed starting with a page (256 bytes) of zeroes. All active primary and extended directory records are logged correctly into this table.

Pass #3

This pass constructs the GAT table. As with pass 2, each file is traced through all of its extents. The entire GAT table will be re-constructed starting with all null characters. All active primary and extended directory records are logged correctly into this table.

Sometimes, one type of an error will inherently cause another type of error. In these cases, fixing even one error may correct several others. As an example, if there is no terminator for a record, repairing this will sometimes correct a multiple files assigned to a single granule error.

When using this utility on Rigid drives, there is one error that may be repaired that the user should be made aware of. LDOS sets aside an extra track (the location can vary), and shows this track as allocated in the GAT table. This track is used by the Laredo and Radio Shack hard disk systems as overhead, and should not be used. The PFIX utility will de-allocate this granule, as it is not assigned to any files. This is only on rigid drive systems, and cannot be detected via the DCT parameters. Due to this situation, PFIX will allocate this track if the following conditions are met:

1. The drive being fixed must be a Rigid drive.
2. The GAT fix option must have been specified.
3. The GAT entry for the required cylinder (middle track for the Laredo, Track 1 for the Radio Shack rigid disk) must be a null byte (not allocated to any files) after the GAT re-build.
4. The L or R option must have been specified.

If these conditions are met, then the extra track is automatically allocated. This could essentially lock-out an available track on a non-Laredo or non-Radio Shack rigid disk system, but will cause no other problems. If you are not sure if your rigid system uses this cylinder or not, then please consult the manufacturer or Logical Systems Inc. It is better to lock out a cylinder than crash the system.

Using PFIX on a Model I double-density system disk with the SOLE modification will cause Track 0 to be de-allocated, which is undesirable. If this is the case, then the PMOD utility must be used to re-allocate Track 0 in the GAT table to prevent the system from attempting to use non-existent sectors in the single density track 0 created by SOLE. Using the command PMOD :d,@, the GAT table can be displayed. Change the first byte of the displayed sector to FFH and write the sector back to the disk.

PREFORM

The PREFORM utility is a disk Re-Formatter program. It allows the user to easily repair "sector not found" and "CRC errors". Only 5" Floppy Disks may be operated on, including Single/Double Density, Single/Double Sided disk systems. The syntax of the PREFORM command is:

```
!    PREFORM,:a,b,c                                    !
!                                                      !
!    :a = drive to be re-formatted (Ø-7)               !
!     b = Y or N (pause on faulty sectors)             !
!     c = starting cylinder (default Ø)                !
!                                                      !
!    abbr: NONE                                        !
```

When the desired parameters have been selected, the specified drive will be read to determine the diskette type (cylinders, density, sides). If there is not enough memory for the re-format utility to operate, the program will abort with an error message. If memory is sufficient, then the re-format operation will proceed as follows:

The current track will be read into memory sector by sector.
The track will be formatted.
The track will be re-written back sector by sector.

If any errors occur during any of the 3 steps, and the Y option was specified, you will be issued an error message, and allowed to R>etry, S>kip, or A>bort. If the N option was specified, processing will continue with no error messages. When the entire disk has been re-formatted, the directory track will be read into memory, and written back with the proper read-protect data address marks.

If you wish the program to abort, you may press the **BREAK** key, and the routine will terminate after the current track has been completed. This will allow normal termination of the program. Pressing the RESET button to abort the program is not recommended, as the current track may be totally lost. If the program is aborted, and the directory track has already been re-formatted, it will not be read-protected. In this case, you must issue a "repair (alien)" command to gain access back to the diskette.

If any CRC errors existed on the diskette, they will be corrected (if the media is OK). If any NOT FOUND errors existed

on the diskette, they also will be corrected, but the previous contents of the sector will be completely lost. This will allow the diskette to be reused with minor losses. If the faulty sector was on the directory track, it is possible that some files records may have been lost also. It is strongly recommended that the user make frequent backups of valued data in order to minimize losses resulting from disk errors. If sectors are lost during the re-formatting process, it is advised that the user recopy the programs that were assigned to the faulty sectors. The PSS utility will assist the user in resolving this.

This program will also "freshen" a diskette. Formatted diskettes have a "shelf" life of about 5 years according to most manufacturers. By re-formatting a diskette, the data will be strengthened, and the shelf life will be extended. If all sectors are readable before re-formatting, then no loss of data will occur.

PVU

The PVU program iw a comprehensive disk verification utility. It allows the user to easily locate and identify faulty sectors on a disk. All LDOS supported disk devices may be operated on, including Single/Double Density, Single/Double Sided, 5" and 8" floppies and fixed/removable disk systems. The syntax of the PVU command is:

```
=======================================================
!    PVU,:a,b,c                                       !
!                                                     !
!     :a = drive to be verified (0-7)                 !
!      b = Y or N (pause on faulty sectors)           !
!      c = starting cylinder (default 0)              !
!                                                     !
!     abbr: NONE                                      !
=======================================================
```

When the desired parameters have been selected, the specified drive will be read to determine the diskette type (cylinders, density, sides). Each cylinder will then be read sector by sector. If any errors are found, and the Y (pause) option was given, you will be given the option to R>etry, S>kip, or A>bort.

When the disk verification process has been completed, you will be given the number of sectors checked, and the number of sectors that were not read successfully. If there were bad sectors, you will be prompted to press ENTER for the list of faulty sectors. Each bad sector will then be listed to the video.

PCLEAR

The PCLEAR program is a comprehensive disk clean-up utility. It allows the user to easily erase unassigned granules and directory records. All LDOS supported disk devices may be operated on, including Single/Double Density, Single/Double Sided, 5" and 8" floppies and fixed/removable disk systems. The syntax of the PCLEAR command is:

```
=======================================================
!    PCLEAR,:a,D,S,b                                  !
!    PCLEAR,filespec,b                                !
!                                                     !
!    :a = clear this drive (colon optional)           !
!    D = clear unused directory records               !
!    S = clear unassigned sectors                     !
!    filespec = any valid LDOS filespec               !
!    b = fill command (default 00H)                   !
!    fill = #cc  (numeric value)                      !
!         = "cc"  (ASCII string)                      !
!           (numbers and ASCII can be mixed)          !
!                                                     !
!    abbr: NONE                                       !
=======================================================
```

After selecting the desired parameters, one of the following routines will be executed.

If a disk was specified to be cleared:

If the S option was given, the directory GAT table will be read into memory. All non-allocated sectors will be written to with the defined fill data.If the D option was given, then all directory records that do not have bit 4 set (i.e., inactive records) will be set to all zeroes.

If a file was specified to be cleared:

All sectors assigned to the file will be written to with all the defined fill data. This will destroy all data belonging to the file.

The user may optionally define what data is to be written into the empty sectors. The default value is zero (00H). A single pound sign (#) followed by a one byte numerical value (0-255) to be used as the fill byte. The number may be entered in Binary, Decimal, Hex, or Octal by preceding the number with B, D, H or O respectively (default decimal). A quotation mark (") followed by

a string will copy this string repetively into the empty sectors. Numerical values and strings may be intermixed.

Examples:

```
PCLEAR,:0,d,s,#h40
```

Fill all unassigned granules with sectors filled with the byte 40H, and clear all unused directory records.

```
PCLEAR,deadfile/cmd:3,"message"
```

Fill all sectors currently assigned to file "deadfile/cmd" with the repetitive string "message".

```
PCLEAR,:7,s,#95,"Kim was here"
```

Fill all unassigned granules with the string "Kim was here" preceded by the cursor character (ASCII 95 decimal). The terminating quote may be left out if it is the last character on the line.

When a file is killed with LDOS, several actions occur. First the first byte of all primary and extended directory records is set to 00H. All granules assigned to the file are released and made available in the GAT table. All primary and extended directory records have their corresponding HIT bytes set to 0.

This will KILL a file, and make it inaccessible to LDOS, but all the information is still there. The directory record is still in the directory, but it is indicated as inactive by the system. All data that was assigned to the file is also still on the disk. Only the directory record has been changed, none of the data has been touched. At this point, a couple of minor changes to the directory can reinsatate the file and make it totally available again with full access to the data.

By running PCLEAR on a drive, you may safely remove all traces of both the files data, and it's associated directory record. If you have a disk that has been used several times, chances are that several "killed" files are still on the disk. If this disk was given to a programmer, chances are that the data could be "unkilled". PCLEAR will make sure that even the best programmer cannot access any of the killed files on your disk, because the information has been completey removed from the disk.

The FILE option of PCLEAR is a very powerful feature, but could be disastrous if you are not careful. This will totally erase all data assigned to the file. Each sector will contain the specifed fill data. This routine could be useful to set all bytes of a file to a known initial value to be used as a data file.

PSS

The PSS program is a Sector Status Utility. It allows the user to easily identify which file is assigned to any sector on a disk. All LDOS supported disk devices may be operated on, including Single/Double Density, Single/Double Sided, 5" and 8" floppies and fixed/removable disk systems. The syntax of the PSS command is:

```
PSS,:a,b,c

:a = drive number (0-7, colon optional)
 b = cylinder number
 c = sector number

abbr: NONE
```

When the specified parameters have been entered, PSS will read the designed directory. If not enough memory is available, PSS will abort with an error message. Otherwise, the drive type will be displayed along with the name, date, and number of free granules on the mounted diskette. If the sector is not assigned to any files, an appropriate message will be displayed, and an exit made back to LDOS. If it is assigned, the filename, and the relative sector number in the file will be returned.

PMAP

The PMAP utility is a comprehensive disk/file mapping program. It allows the user to easily locate files' sectors, and determine the status of all granules on a disk. All LDOS supported disk devices may be operated on, including Single/Double Density, Single/Double Sided, 5" and 8" floppies and fixed/removable disk systems. The syntax of the PMAP command is:

```
PMAP,*,:a
PMAP,*,filespec

* = send output to printer
:a = map this drive
filespec = any valid filespec

abbr: NONE
```

Mapping a drive:

The GAT table of the specified diskette will be read into memory. Each cylinder will be mapped granule by granule. The display will shkw the cylinder number, and a liwt of all the granules on a sector basis. An example of a single density, single sided disk could be:

Cylinder 00: 00-04xx 05-09xx

The first x position will indicate the status of the Allocation Bit associated with gran, and the sectors 00-04 indicate the range of sectors in the first gran. This position will display a graphic block if the corresponding granule is assigned, and a period if the granule is currently available. The second x position indicates the status of the corresponding Lockout Bit for the gran. This position will display an X if the corresponding granule is locked-out, or a space if it is available. All granules indicating locked-out status should also have the corresponding allocation bit set.

NOTE: Rigid drives do not have a lockout table, so that information will not given.

The display will pause each screenful and wait for you to press a key to continue. Press BREAK to abort at any time, or SHIFT @ to pause the display manually. This procedure will continue till all cylinders have been displayed.

Mapping a File:

The associated directory record for the given file will be read into memory. All sectors assigned to the file will be traced through the directory manually. The current sector that is being operated on will be displayed to the video (or printer if specified). An attempt will be made to read each sector as it is computed. If the sector was read successfully, it will be surrounded by a thin line (ASCII 95). If a read error was issued by the system, the cylinder and sector number will be surrounded by graphic blocks. At any time, you may abort by pressing BREAK, or pause by pressing SHIFT @. This procedure will continue until the end of file is reached, and an exit is made back to LDOS.

PASSGO

The PASSGO program is a comprehensive password removal utility. It allows the user to easily remove passwords on a single file or an entire diskette. All LDOS supported disk devices may be operated on, including Single/Double Density, Single/Double Sided, 5" and 8" floppies and fixed/removable disk systems. The syntax of the PASSGO command is:

```
PASSGO,:a,V,I,S,N
PASSGO,filespec

:a = remove all passwords this drive
 V = include VISIBLE files (default)
 I = include INVISIBLE files
 S = include SYSTEM files
 N = exclude visible files
 filespec = any valid LDOS filespec

abbr: NONE
```

Once the specifed parameters have been entered, one of the following 2 routines will be performed:

If an entire disk is specified, all files that are included by the command will have both their ACCESS and UPDATE passwords set to nil (8 spaces), and the protection level will be set to 0. The diskette master password will also be set to PASSWORD. The user should note that if the file CONFIG/SYS is affected by this command, it will no longer be loaded by the system. This may be corrected by:

```
ATTRIB CONFIG/SYS(acc=ccc,upd=ccc,inv)
```

If a single file has been specified, then that file will have its ACCESS and UPDATE passwords set to nil, and the protection level will be set to 0 (full access). If a password currently exists for the file, it must be supplied with the command for the password to be stripped off. In this case the following two commands would be identical:

```
PASSGO filename/ext.password:d
ATTRIB filename/ext.password:d (acc=,upd=,prot=full)
```


PUN

The PUN utility is a Diskette Un-Repair program. It allows the user to easily reverse the effects of a "repair (alien)" command that was used to convert the data address marks used by TRSDOS to that used by LDOS. This will allow TRSDOS I to access a directory written to by LDOS. This program will only work in a Model I (the Mod III cannot reproduce the desired address mark), and will only operate on a single density, single sided, 5" floppy disk. The syntax of the PUN command is:

```
PUN,:a

:a = drive number to un-repair
```

When the appropriate parameter has been specified, the directory will be read into memory, and immediately written back with the Mod I read-protect data address marks. This will allow TRSDOS I to access the directory again.

In order to gain compatibility between the models I and III, LDOS has chosen to write the single density directories on a Mod I using the User Defined Address Mark. This will cause TRSDOS to report a read error on the directory, or at least cause a lot of thrashing while the system does a re-seek 10 times for each sector. One pass through PUN, and no more problems. The diskette will become readable to a TRSDOS I system, and will also remain readable to LDOS.

PKILL

PKILL is a comprehensive multiple-file purge utility which allows the user to quickly and simultaneously kill several files from a disk. Files to be killed may be classified using partspecs or the standard categories of Visible, Invisible or System. Alternatively, a list of files to be killed may be supplied on the command line when the utility is invoked. PKILL may be used on all LDOS supported disk devices including 5" or 8" single/double sided, single/double density floppy disks and rigid drives. The syntax of the PKILL command is:

```
PKILL :a,!,V,I,S,N,A,L,R,/ext,$wild
PKILL :a,*,/ext,file1,file2,file3 .....

:a = drive number, 0 through 7 (colon optional)
 ! = erase directory entry of affected files
 V = Include VISIBLE files (default)
 I = Include INVISIBLE files
 S = Include SYSTEM files
 N = Exclude visible files
 A = Include ALL files
 L = Indicates a LAREDO rigid disk
 R = Indicates a RADIO SHACK rigid disk
/ext = include files with this extension
$wild= Include files starting with the charac-
       ters "wild"
* = kill all files in the list following this
    asterisk (supply default extension if
    needed).

Abbr: NONE
```

PKILL allows the user to quickly and simultaneously kill several files from a disk. The files to be killed may be classified according to the standard categories V(isible), I(nvisible) and S(ystem), or may be files with a common extension, or files which begin with a particular set of characters. Files which have NO common characteristics may also be killed by entering an asterisk (*) followed by a list of filenames. The filenames may be assigned a default extension if needed. All LDOS-supported disk devices are supported by PKILL, up to and including hard drive systems. When using PKILL on a hard drive system, the L (for Laredo systems) or R (for Radio Shack systems) parameter must also be supplied to inform PKILL that it is dealing with a rigid disk system that requires a track

locked out for system use. A colon is optional when specifying a particular drive number.

After the user selects the desired parameters, PKILL reads the directory track of the target disk into memory and proceeds to kill the necessary files. It then writes the now-modified directory back onto the disk. This procedure allows a very fast kill of multiple files.

If the "!" parameter is not specified, the files are merely flagged as inactive in the directory, leaving the possibility of later recovering a killed file. However, if "!" is specified, the entire directory entry for a killed file is zeroed out, making later recovery impossible.

Any combination of the parameters V, I, and S are permitted by PKILL. N and V are mutually exclusive. The use of the "A" parameter is a very good way to produce a data disk with a totally blank directory and the user should exercise caution when using it.

When a list of files is supplied, an asterisk should be entered as the first item to inform PKILL that what follows is a list of files and not additional parameters. PKILL will kill each file in the list.

Default extensions for files to be killed may be supplied by entering the extension after any file classification identifiers (S, I, V, etc) or in front of a list of files. In the latter case, the extension will be added to any filenames on the list which do not have explicitly typed extensions. To avoid having the default extension added onto a file, it must be entered as FILE/ to force a blank extension.

If PKILL is typed in without any parameters, the user will be prompted to supply them when the program starts. At this point the user may type in the necessary information. Up to 255 characters will be accepted. It may be useful, therefore, when entering a long list of filenames to simply enter PKILL and then type the list of names at the prompt. This is because the LDOS command buffer limits keyboard entry to only 64 characters.

EXAMPLES:

```
PKILL :0,I,/CIM
```

This command will cause all visible and invisible files on drive 0 with the extension /CIM to be killed. Note that since V defaults to ON, it is not necessary to type it in.

```
PKILL :3,!,/DAT
```

All visible files on drive 3 with the extension /DAT will be killed. The slots they occupied in the drive 3 directory will be zeroed out.

```
PKILL :1,!,A
```

This command will produce a totally blank data diskette on drive 1. All files except for BOOT/SYS and DIR/SYS will be killed and their directory entries zeroed out.

```
PKILL :0,*,/cmd,test,test2,prog/bas,Foo/
```

The files TEST/CMD, TEST2/CMD, PROG/BAS and FOO will be killed. All the files must exist on drive 0.

PCOMPARE

The PCOMPARE program is a comprehensive disk/file compare utility. It allows the user to locate inconsistencies between disks or files. All LDOS supported disk devices may be operated on, including Single/Double Density, Single/Double Sided, 5" and 8" floppies and fixed/removable rigid disk systems. The syntax of the PCOMPARE command is:

```
PCOMPARE,*,:a,b,c,d,:e,f,g
PCOMPARE,*,file1,file2,h

 * = output to printer
:a = source drive (0-7)
 b = source track (default 0)
 c = source sector (default 0)
 d = sector count (default to disk end)
:e = destination drive (0-7)
 f = destination track (default 0)
 g = destination sector (default 0)
 file1, file2 = any valid LDOS filespec
 h = starting relative sector (default 0)

 abbr: NONE
```

After selecting the desired parameters, the source data will be loaded from disk one sector at a time, and compared to the specified destination sector. If any bytes do not match, their locations will be reported in the following format:

```
:x, Cyl xxx, Side x, Sect xxx <<==>> :x, Cyl xxx, Side x, Sec xxx
At Relative xxx (xxH) to yyy (yyH) for xxx (xxH) byte(s).

Data Mismatch File Relative Sector xxxxx.
At Relative xxx (xxH) to yyy (yyH) for xxx (xxH) byte(s).
```

This will continue until all bytes in the sectors have been compared. When all sectors have been compared, the following will be displayed:

```
xxxxx Total Sectors Compared.
xxxxx Sectors Did Not Match.
xxxxx+Bytes Did Not Match.
```

The plus sign at the byte mismatch message will appear if more than 65535 (FFFFH) bytes total did not match, otherwise it will be absent.

PFIND

The PFIND program is a comprehensive string search and replace utility. It allows the user to locate and replace strings in memory, in a file, or the disk on a sector by sector basis. All LDOS supported disk devices may be operated on, including Single/Double Density, Single/Double Sided, 5" and 8" floppies and fixed/removable rigid disk systems. This program will locate strings in several different formats, and upper/lower case independent searches may be specified. The syntax of the PFIND command is:

```
=====================================================
!    PFIND,*,@,:a,b,c,d,s,>t                        !
!    PFIND,*,@,filespec,e,s,>t                      !
!    PFIND,*,@,ffff,gggg,s,>t                       !
!                                                   !
!     * = output to printer                         !
!     @ = display match locations                   !
!    :a = drive number (0-7, colon optional)        !
!     b = starting cylinder (default 0)             !
!     c = starting sector (default 0)               !
!     d = sector count (defaults to disk end)       !
!     filespec = any valid LDOS filespec            !
!     e = starting relative sector (default 0)      !
!     ffff = memory start address                   !
!     gggg = ending address (default topmem)        !
!     s = string to locate                          !
!     >t = replacement string (optional)            !
!     string = "abcde????fgh"                       !
!            (literal string, case dependent)       !
!            = 'abcde????fgh'                       !
!            (literal string, case independent)     !
!            = abcde????fgh                         !
!            (? matches anything, case independent) !
!            = #,a,b,c (byte list)                  !
!            = ##,a,b,c (word list)                 !
!                                                   !
!    abbr: NONE                                     !
=====================================================
```

After selecting the desired parameters, the search will begin at the specified starting location. If a disk or file search is requested, the entire string must be contained wholly within a sector, as no spanning across sectors is done. If any matches are found, and the @ option is specified, their locations will be returned to the video, or the printer if the * option is given. If the @ was not issued, the location will be counted, but not displayed. If a replacement string was given, it will be written

back to the source data. If any errors occur during either the read or write cycle, you will be issued an error message, and given the option to R>etry, S>kip, or A>bort. If a replacement string was specified that is longer than the source string, it will be truncated to the length of the source string.

The following types of strings will be accepted by the system:

Literal string, case dependent: "string"

By surrounding the string with quotation marks ("), it will be interpreted by PFIND as a literal string, no conversion of case will be made. Each letter specified must match exactly. This means that upper case and lower case are treated as separate characters. If the source string is "Kim" for example, then KIM will not match because the i and m are considered different characters.

Literal string, case independent: 'string'

By surrounding the string with apostrophes ('), it will be interpreted by PFIND as a literal string, but all characters will be converted to upper case. When scanning for matches, each character of data is converted to upper case before the comparison is made. This will allow you to locate matches even if the upper/lower case characters do not match exactly. If the source string is 'Kim' for example, then KIM, kIM, kim, etc. will all match. The upper and lower case characters are considered equal in this type of search.

? matches anything, case independent: str?ng

This type of search is identical to the above mentioned example, except that any question mark characters (?) will match with any characters in the data searched. If there are no ? characters in the string, then this and the above would be identical in all respects. If the source string is K?m for example, then KIM, kIM, kim, kam, kxm, etc. will all match. The upper and lower case characters are considered equal.

Byte list: #,a,b,c

This type of search will allow you to specify any string of bytes. The single pound sign (#) tells PFIND that a list of one byte numerical values will be following. The values may be made in Binary, Decimal, Hex, or Octal by preceding the number with B, D, H or O respectively (default decimal). All numbers indicated must be in the range of 0-255 (0-FFH).

Word list: ##,a,b,c

This search allows a string of 2 byte words. The double pound sign (##) tells PFIND that a list of 2 byte numerical values will follow. The values may be in any number base following the rules in byte list. All numbers must be in the range of 0-65535 (0-FFFFH). The numbers are placed into the string in LSB, MSB order.

Sample Strings:

"Kim Watt" becomes 4BH,69H,6DH,20H,57H,61H,74H,74H
'Kim Watt' becomes 4BH,49H,4DH,20H,57H,41H,54H,54H
Kim Watt becomes 4BH,49H,4DH,20H,57H,41H,54H,54H
#,1,2,3 becomes 01H,02H,03H
##,1,2,3 becomes 01H,00H,02H,00H,03H,00H

If you are searching memory, the locations will be reported as follows:

Memory Match at xxxxH

If you are searching disk sectors, locations are reported as follows:

Disk Match, Drive x, Cylinder xxx, Sector xxx, Relative Byte xxH.

If you are searching a file, then matches are reported as:

File Match at Relative Sector xxxxx, Relative Byte xxH.

The following are sample command lines:

PFIND,*@:0,'kim watt',>"Kim Watt"

Searches drive 0 from start to end. If the string 'Kim Watt' appears in ANY case, replace with the string "Kim Watt" in upper/lower case. If any matches are found, they will be displayed to the video and printer.

PFIND,:0,'kim watt'

Searches drive 0 from start to end. Reports the total number of occurences of the string 'Kim Watt'. Locations are not displayed, only the total number of matches.

PFIND,@pfind/cmd,kim watt

Searches the file PFIND/CMD for all occurrences of the string 'Kim Watt' in either upper or lower case, and list heir locations on the video.

PFIND,@sys0/sys,##,h4411

Searches the file SYS0/SYS for all occurrences of the binary word 4411H (topmem Mod III), and reports their locations on the video.

PFIND,@h4000,##,h4411

Searches memory from 4000H to TOPMEM for the binary word 4411H, and reports the locations of all matches on the video.

PMOVE

The PMOVE utility is a comprehensive file copy program. It allows the user to easily transfer multiple files from one disk to another with a minimum amount of keystrokes. All LDOS supported disk devices may be operated on, including Single/Double Density, Single/Double Sided, 5" and 8" floppies and fixed/removable disk systems. The syntax of the PMOVE command is:

```
=====================================================
!    PMOVE,:a,:b,/cde,file1,file1>file2             !
!                                                   !
!    :a = source drive number (0-7)                 !
!    :b = destination drive (0-7)                   !
!    /cde = default extension for all files         !
!    file1,2 = any valid LDOS filespec              !
!    > = rename file to following filename          !
!    followed by a list of files                    !
!                                                   !
!    abbr: PMOVE,ab/cdefile1,file2                  !
=====================================================
```

Once the desired parameters have been entered, the source and destination directories will be read to establish the disk types. The command line may be specified sith the program name from the DOS command level, or you will be prompted for it. If entered from LDOS, you will be limited to the 63 character input buffer. If prompted from the program, you will have a full 255 character buffer to specify the files. You may also optionally enter the asterisk character (*) as the first command, and the screen will not be cleared when the program initializes. If you have a directory on the screen, type PMOVE,* and you will have access to the 255 character input, and the filenames will still be on the screen.

PMOVE may be used to copy a single file:

PMOVE,01pmove/cmd

but the time saving is really noticed when several files are to be copied:

PMOVE,01/cmdpmove,pmod,pvu,pfilt/flt,passno,pmap

the above command will copy 6 files from drive 0 to 1, and the default extension of /CMD will be added to all files except PFILT/FLT.

PMOVE may also be used to duplicate the same source file to several destination files using the rename capabilities:

```
PMOVE Ø1/cmd,pmove,pmove>pmove1,pmove>pmove2,pmove>pmove3
```

the above command will take PMOVE/CMD from drive Ø, and create 4 identical files on drive 1 named PMOVE/CMD, PMOVE1/CMD, PMOVE2/CMD, and PMOVE3/CMD.

PERASE

The PERASE utility is a Software Bulk-Erase program. It allows the user to easily remove all traces of data on a diskette, and return it back to a blank state. Only 5" Floppy Disks may be operated on, including Single/Double Density, Single/Double Sided disk systems. The syntax of the PERASE command is:

```
+-------------------------------------------------+
!  PERASE,:a                                      !
!                                                 !
!  :a = drive to be erased (Ø-7)                  !
!                                                 !
!  abbr: NONE                                     !
+-------------------------------------------------+
```

When the desired parameter has been selected, the specified drive will be read to determine the diskette type (cylinders, density, sides). The diskette must be currently formatted. If LDOS cannot establish the type of diskette, the program will abort with a "device not available" message. If there is not enough memory for the erase utility to operate, the program will abort with a corresponding message. If memory is sufficient, then the erase operation will proceed. You will be asked if you are ABSOLUTELY SURE if you want to proceed with this operation. All data will be lost, so be sure that the correct diskette is mounted before you answer Yes.

Each track will be formatted with all bits off (zeroes). This will effectively remove all traces of sectors and data on the diskette. If you wish to abort the program, press the BREAK key, and the routine will abort at the completion of the current track. When the entire disk has been erased, it may be considered a completely blank disk. The program will essentially perform the same task as using bulk-erase magnets to delete all information from the disk.

PDIRT

The PDIRT program is a directory utility with the ability to read Mod III TRSDOS formatted diskettes from within the LDOS operating environment. It allows the user to easily list files that are normally not readable via LDOS commands. This program requires that the disk to be read is previously formatted by TRSIII. On a Mod I, this program will only operate if the appropriate double density hardware is present, and corresponding LDOS double density disk drivers are in memory. The syntax of the PDIRT command is:

```
=====================================================
!    PDIRT,:a,I,S,O,P                               !
!                                                   !
!    :a = drive number (0-7, colon optional)        !
!     I = include INVISIBLE files                   !
!     S = include SYSTEM files                      !
!     O = include system OVERLAYS                   !
!     P = output to printer                         !
!                                                   !
!    abbr: NONE                                     !
=====================================================
```

When the desired parameters have been specified, the diskette will be read into memory. All VISIBLE files will automatically be listed, along with invisible and system files if requested. Because TRSDOS Mod III does not log the system overlays into normal directory records, but instead logs them into the last 32 bytes of the HIT table, if the O option is specified, a list will be displayed of the system overlays that are currently active in the directory. Each will be listed by a 2 digit decimal number from 00-15 if the corresponding overlay position contains an active pointer.

PEX

The PEX utility is a diskette exerciser program. It allows the user to move the head to any cylinder (for alignment purposes), or to move the head in a continuous motion (for cleaning disk purposes). The syntax of the PEX command is:

```
=====================================================
!    PEX,:a,bbb                                     !
!                                                   !
!    :a = drive number (0-7, colon optional)        !
!    bbb = cylinder count                           !
!          (optional if formatted disk mounted)     !
!                                                   !
!    abbr: NONE                                     !
=====================================================
```

Once the specified command has been entered, the specified drive head will be restored (moved to cylinder 0), and a sub-menu will be displayed. Several keys are recognized by the program at this point:

D	allows you to specify a new drive number
C	allows you to move the head to any cylinder
S	set a new step rate for the drive
A	automatically step from cylinder 0 to the top cylinder
R	move the head to cylinder 0
T	move the head to the top cylinder

If the automatic mode has been set, any of the other commands will terminate the operation. Pressing BREAK at any time will abort the program and return control to LDOS.

If you have a head cleaning diskette, the automatic mode may be used to move the head across the diskette. Use directions supplied by the head cleaning manufacturer. You may turn the system clock on prior to entry into the program to have a seconds timer available. It is not advisable to run PEX on any one drive for more than 15-30 seconds at a time while the head cleaning disk is in place. Although most manufacturers claim that the kits are non-abrasive, too much of a good thing could mean trouble, and this means head wear. Please follow your kit's instructions. We suggest that if you only use your computer a few hours a day, then one cleaning a year should be sufficient. Even heavy users should only clean once a month or less. You'll notice that the head cleaning kit does not tell you how to actually clean the head. Most people simply insert the disk, and boot the system.

This does not do the job, and of course, will only work on the drive 0 disk.

PMX

This filter will adjust graphic characters so that they are printed out as normal TRS80 graphic when using an Epson MX80 printer. The syntax of the command is:

FILTER *PR,PMX

The MX80 printer will not print standard TRS80 graphics unless it is set in the TRS80 mode, in which case you lose some of the special printer capabilities of the machine. This filter will allow use of these special features while correctly printing graphic characters identical to the TRS80 computer.

PHELP

This utility will allow easy access to the syntax associated with the LDOS library commands and several utilities. The syntax of the PHELP command is:

```
PHELP *
PHELP *,?
PHELP *,command

* sends output to printer
? displays syntax for PHELP
command - displays syntax for
the associated command

abbr: commands may be abbreviated
```

When the desired command has been issued, the associated syntax for the command will be displayed on the video, and also sent to the printer if the * was issued.

PFILT

This is a comprehensive, user definable conversion filter that may be used for either input or output devices. The program will read an ASCII text file created by the user (via the BUILD command) to determine the parameters. The syntax of the PFILT utility is:

```
FILTER *aa PFILT,filespec

aa = any LDOS defined device
filespec = any valid LDOS filespec
           default extension /JCL

abbr: NONE
```

This command will read the designated file and create a table of the parameters in the file. The entire program will then relocate itself to high memory, and protect itself. The format of the data file is as follows:

A numerical value, followed by an equal sign (=), followed by the value to be converted to.

If the first character is a colon (:), then the following character is interpreted in its ASCII form.

If the first character is a period (.), then the following data on the line is considered a remark, and is not executed, although it will be displayed to the video.

All numerical values may be entered in Binary, Decimal, Hex or Octal by preceding the number with B, D, H or O respectively (default decimal).

Following are examples of acceptable syntax:

```
31=37
31=h44
o37=d44
bl10110ll=o177
hff=hfe
:A=a
:Q=B
```

Two sample filter files for use with PFILT are included on the disks, called CODE/JCL and DECODE/JCL. These sample files show how PFILT may be used for coding/decoding purposes. These are not supplied as sophisticated techniques, but merely as examples demonstrating the power of the PFILT filter. The syntax of the commands are:

```
FILTER *KI,PFILT,CODE
FILTER *DO,PFILT,DECODE
```

(The PFILT program uses the default extension /JCL which is the default extension supplied to files by the BUILD library command of LDOS)

Normally, you would only want to have one driver in memory at a time. When the *KI is being filtered, only upper case input will be changed, thereby allowing the user to enter DOS commands in lower case. Use the SPACEBAR to enter commas while the CODE filter is active. Following is a sample session demonstrating the capabilities of the system:

(Enter the case exactly as entered below)

```
filter *ki,pfilt,code
build test/fil:0
THIS IS A TEST OF A CODED MESSAGE. <ENTER>
<BREAK>
reset *ki
LIST TEST/FIL:0
(SHOULD NOT BE READABLE!)
filter *do,pfilt,decode
list test/fil (SHOULD NOW BE READABLE!)
reset,*do
```


DVORAK/FLT

This is the filter file for the famous DVORAK keyboard. It contains drivers for both DVORAK and QWERTY, and each driver may easily be toggled to the other. The program also allows a caps lock function. The program may be terminated at any time, and the memory it is using will be reclaimed if it was the last driver into high memory. The syntax of the DVORAK command is:

```
FILTER *KI,DVORAK,a

a = Q or D if the program will
    initialize as QWERTY or DVORAK
    (default DVORAK) (optional)

abbr: NONE
```

This will load the driver into memory, at which time it will relocate itself to high memory and protect itself. It is recommended that the user SET *KI KI/DVR before setting up the DVORAK filter so that the control keys will be active.

You may toggle between DVORAK and QWERTY by pressing the CLEAR 0. The SHIFT 0 toggles the upper case lock feature. The driver may be disabled with CLEAR BREAK. If it was the last file into high memory, then the memory that it occupies will be reclaimed to the system and a corresponding message will be displayed.

There is another DVORAK file on your disk, called DVORAK/JCL. This is the DVORAK keyboard filter file that can be used in conjunction with the PFILT filter. It is a pure ASCII data file, and although it does not provide the user with the full features of the DVORAK/FLT utility, it does require much less memory. The syntax of the command is:

```
FILTER,*KI,PFILT,DVORAK

abbr: NONE
```

After this command, the DVORAK filter will be installed, moved to high memory, and protected.

PBOOT

This patch will allow you to change the screen graphics displayed by LDOS at bootup, without affecting any of the copyright information. Space is provided for 3 lines of 22 characters each of user definable text. The syntax for applying PBOOT is:

```
PATCH SYSØ/SYS.RSØLTØFF,PBOOT
```

This will patch the code directly into the SYSØ system overlay, and will appear each time the system is booted.

You may use a text editor or word processor (one capable of producing pure ASCII files) to insert your text into the patch. The text lines are clearly defined. They may not be shortened or lengthened. If your text does not take up the full 22 characters on a line, you must pad it out with blanks.

NOTICE

This software is sold on an AS-IS basis only, and Breeze/QSD shall not be liable for any damage or loss, whether real or alleged, arising from the use of this software. No warranties of merchantability or fitness are made, expressed, implied or imagined. Installation and determination of fitness for a particular purpose is the sole responsibility of the user.

Acknowledgements