LDOS (tm)LSI
UTILITY DISKS

By: KIM WATT

from

POWERSOFT

a division of Breeze/QSD, Inc.

UTILITY DISK #2 (c)1982

$29.95

Disk #2 Includes:

PMOD/CMD - A Disk-File-Memory Modification Utility.

PMOD,aaaa (display memory address)
PMOD,filespec,aaaa (file to display,starting rel. sector)
PMOD :b,c,d (drive,cylinder,sector to display)

LOADING INSTRUCTIONS

- MOD I or III -

1) Prepare a SINGLE density LDOS formatted diskette. This disk may be 35, 40, or 80 tracks, but SINGLE density!
2) Boot the MASTER disk you received in this package in drive 0. You will be prompted to insert the destination diskette, and the files will be "moved" to YOUR diskette with no effort on your part. Single drive owners will need to swap disks a few times.
3) LX-80 users should treat the enclosed MASTER disk as a 35 track single density data diskette.

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,aaaa                                        !
!  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 (Ø-7, colon mandatory) !
!       d  = cylinder number (defaults Ø)           !
!       e  - sector number (defaults Ø)             !
!                                                   !
!  abbr: NONE                                       !
=====================================================
```

NOTE: It is recommended to 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 Hex, Decimal, Octal, or Binary by PRECEDING the number with H, D, O, or B respectively. 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:

```
    0000 F3AF C315 30C3 0040 C300 40E1 E9C3 1230
HEX0010 C303 40C5 0601 182E C306 40C5 0602 1826
MEM0020 C309 40C5 0604 181E C30C 4011 1540 18E3
    0030 C30F 4011 1D40 18E3 C312 4011 2540 18DB
    0040 C3D9 05C9 0000 C374 06CD 2B00 B7C0 18F9
    0050 11E5 4118 BE11 ED41 18C1 11F5 4118 BC00
    0060 C3FB 0120 FBC9 C339 30C3 5204 111D 4218
    0070 AA00 C3CC 0611 8040 21F7 1801 2700 EDB0
    0080 21E5 4236 3A23 7023 362C 2322 A740 112D
    0090 0106 1C21 5241 36C3 2373 2372 2310 F706
    00A0 1536 C923 2323 10F9 21E8 4370 31F8 42CD
    00B0 BF1B 0000 0021 0501 CDA7 28CD B31B 38F5
    00C0 D7B7 2012 214C 4423 7CB5 281B 7E47 2F77
    00D0 BE70 28F3 1811 CD5A 1EB7 C297 19EB 2B3E
    00E0 BF46 77BE 7020 CE2B 1114 45DF DA7A 1911
    00F0 CEFF 22B1 4019 22A0 40CD 4D1B 2111 01C3
```

The far left column of the screen indicates the current modify 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. 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 between 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 to the video. Please refer to the

```
     00 00FE 14F3 3A02 4357 1E04 CD74 4320 713A .▗.▝:.CW..▙tC q:
HEX  10 0051 E610 286E D92A 1651 557C 0707 07E6 .Q▚.(n▚*.QU¦...▚
DRV  20 0767 0784 075F 21FF 51D9 CD5D 433D 200C .g..._!▝Q▙]C= .
 0   30 CD4E 43CD 5D43 7723 10F9 18EE 3D28 0BCD ▙NC▙]Cw#.▚.▚=(.▙
CYL  40 5D43 47CD 5D43 10FB 18E0 CD4E 43E9 CD5D ]CG▙]C.▚.▚NC▚▚]
 00  50 4347 CD5D 436F 05CD 5D43 6705 C9D9 2C20 CG▙]Co.▙]Cg.▚▚,
SID  60 0FE5 CD74 4320 19E1 1C7B D612 2002 5F14 .▙▙tC .▚.{▚. ._.
 0   70 7ED9 C9C7 0605 C5CD 8E43 C1E6 1CC8 10F6 ~▚▚▚..▙▙C▚▚.▚.▚
SEC  80 21E6 43DD 21F4 43CD 1B02 CD40 00C7 01F4 !▚C▚!▙C▙..▙@.▚.▙
 00  90 B1ED 410D 3E18 ED51 CDDF 43DB F0CB 4720 ▚▙A.>.▙Q▙▙C▚▚▚G
STD  A0 FA7B D3F2 3E81 D3F4 21D3 4322 4A40 3EC3 ▙{▙▙>▚▙▚▚!▚C"J@>▙
 5"  B0 3249 4021 0051 D511 02C1 3E80 CDDF 433E 2I@!.Q▙..▙> ▙▙C>
FLP  C0 C0D3 E4DB F0A3 28FB EDA2 7AD3 F4ED A220 ▚▚▚▚▚▚{▚▚▚z▚▚▚▚▚
 DD  D0 FA18 FEE1 D1AF D3E4 3E81 D3F4 DBF0 C9D3 ▙▚▚▚▚▚▚▚>▚▚▚▚▚▚▚
 SS  E0 F006 0C10 FEC9 17E8 4469 736B 2065 7272 ▚...▚▚.▚Disk err
     F0 6F72 1F03 17E8 4E6F 2073 7973 7465 6D03 or...▚No system.
```

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:

```
     P00 0506 504D 4F44 2020 0102 0052 E521 A462 ..PMOD  ...R.!.b
HEX  M10 CD67 44E1 1807 2100 6436 0D18 EFCD 6F53 .gD...!.d6....oS
DRV  O20 DA48 53FE 3F20 08E5 CD7C 60E1 2318 EEFE .HS.? ...|`.#...
 0   D30 3A28 19FE 21CA EA52 E5CD 8253 E1DA EB52 :(..!..R...S...R
     /40 ED43 EE55 3E01 324C 56C3 2155 237E D630 .C.U>.2LV.!U#~.0
     C50 DA39 53FE 08D2 3953 3268 524F E53A 0400 .9S...9S2hRO.:..
     M60 FE30 2805 CDB8 4418 03CD 9042 3E08 E1C2 .0(...D...B>...
     D70 3B53 E53E 004F CD8F 47FD 220C 5EE1 23AF ;S.>.O..G.".^.#.
      80 32CF 5D32 CE5D FD7E 07E6 1F3C FDCB 046E 2.]2.]~...<...n
      90 2802 CB27 FDCB 035E 2811 4FFD 7E07 0F0F (..'...^(.O.~...
      A0 0F0F 0FE6 073C 47AF 8110 FD32 C354 CD6F ....<G...2.T.o
      B0 5338 23FE 4020 06FD 7E09 2318 07CD 8253 S8#.@ ..~.#...S
RSEC  C0 DA42 5379 32CF 5DCD 6F53 380A CD82 53DA .BSy2.].oS8...S.
0000  D0 4253 7932 CE5D 3E02 324C 56C3 9C5D 21B9 BSy2.]>.2LV..]!.
EOFS  E0 61CD 6744 061F 2100 64CD 4000 DA65 5378 a.gD..!.d.@..eSx
0018  F0 B7CA 0A52 1832 2311 5F64 CD1C 44C2 3B53 ....R.2#._d..D.;S
```

The screen layout is similar to the 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 that 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, and will require a "Repair (alien)" after updating back to 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:

```
67]  00 00FE 14F3 3A02 4357 1E04 CD74 4320 713A .[.=:.CW..tC q:
HEX  10 0051 E610 286E D92A 1651 557C 0707 07E6 .Q.(n.*.QU|....
DRV  20 0767 0784 075F 21FF 51D9 CD5D 433D 200C .g..._!.Q.]C= .
 0   30 CD4E 43CD 5D43 7723 10F9 18EE 3D28 0BCD .NC.]Cw#..=(..
CYL  40 5D43 47CD 5D43 10FB 18E0 CD4E 43E9 CD5D ]CG.]C..N C.]
 00  50 4347 CD5D 436F 05CD 5D43 6705 C9D9 2C20 CG.]Co..]Cg.
SID  60 0FE5 CD74 4320 19[illegible] 1C7B D612 2002 5F14 .[.tC .{. ._.
 0   70 7ED9 C9C7 0605 C5CD 8E43 C1E6 1CC8 10F6 ~..[..C..
SEC  80 21E6 43DD 21F4 43CD 1B02 CD40 00C7 01F4 !.C.!.C...@..
 00  90 B1ED 410D 3E18 ED51 CDDF 43DB F0CB 4720 ..A.>..Q..C..G
STD  A0 FA7B D3F2 3E81 D3F4 21D3 4322 4A40 3EC3 .{..>..!.C"J@>.
 5"  B0 3249 4021 0051 D511 02C1 3E80 CDDF 433E 2I@!.Q..>..C>
FLP  C0 C0D3 E4DB F0A3 28FB EDA2 7AD3 F4ED A220 ..(..z..
 DD  D0 FA18 FEE1 D1AF D3E4 3E81 D3F4 DBF0 C9D3 .....>.....
 SS  E0 F006 0C10 FEC9 17E8 4469 736B 2065 7272 ....D.. Disk err
     F0 6F72 1F03 17E8 4E6F 2073 7973 7465 6D03 or... No system.
```

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 - homes the cursor to the first relative byte being displayed.
CLEAR ENTER - allows you to change the current BASE you are modifying in without leaving the modify mode. You will have a new flashing prompt in the left column of the display. Enter A, B, D, H, or O 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 you will not be able to move the cursor beyond the edges of the displayed data. In the 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:

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 OCCURENCE of the specified byte. The cursor will move to the last byte on the screen if the requested byte is not located.
P - followed by a number DUPLICATES 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.

NOTE: The above five 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 part of NUMERICAL input. If you are in HEX mode, for example, you must enter TWO HEX DIGITS (Ø-9,A-F) to change a single byte. Decimal mode expects 3 digits (Ø-9), Octal mode expects 3 digits (Ø-7), and Binary mode expects 8 digits (Ø-1). You may pre-terminate the numerical input by pressing ENTER. Thus, in HEX mode, 3 ENTER is the same as entering Ø 3, in Binary mode, 1 1 1 ENTER is the same as entering Ø Ø Ø Ø Ø 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, and P, the numerical input following the command will be in the current BASE. If you are in HEX modify, then G1Ø is a valid command to move the cursor to relative byte 1ØH, but in DECIMAL modify, you must enter GØ16 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 DOS 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, you may PRECEDE the filename with an exclamation point (!) to FORCE PMOD to interpret it as a filespec.

PMOD,HFØØØ will display memory address FØØØH.
PMOD,!HFØØØ will display the file HFØØØ

LDOS (tm)LSI
UTILITY DISKS

By: KIM WATT

from

POWERSOFT

a division of Breeze/QSD, Inc.

UTILITY DISK
3 (c)1982

$29.95

Disk #3 Includes:

PCHECK/CMD - LDOS Directory Check Utility. Checks a drive or a file, (GAT, HIT, BOOT, etc.) for integrity.

PFIX/CMD - FIXES problem discovered by PCHECK/CMD. You may repair GAT or HIT table, repair BOOT sector, or repair FILE records. This program might fix that crashed disk that has you going nuts!

LOADING INSTRUCTIONS

- MOD I or III -

1) Prepare a SINGLE density LDOS formatted diskette. This disk may be 35, 40, or 80 tracks, but make sure it is SINGLE DENSITY!
2) BOOT the MASTER disk you received in this package in Drive 0. You will be prompted to insert the destination diskette, and the files will be "moved" to YOUR diskette with no effort on your part. Single drive owners will need to swap disks a few times.
3) LX-80 users should treat the enclosed MASTER disk as a 35 track single density DATA diskette.

PFIX

The PFIX Utility is a comprehensive Directory Repair Utility. It allows the user to completely repair most directory problems, and also has the ability to transfer a faulty boot sector from a good disk. 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 repair most inconsistencies in the directory information of a disk. In conjunction with PCHECK, most directory problems can be easily located and corrected without extensive knowledge of how directories are formatted. The syntax of the PFIX command is:

```
=====================================================
!  PFIX,:a,G,H,F,B=:d,L,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  = lockout cylinder above directory       !
!            (rigid systems only)                   !
!       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 occuring 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 they 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 file 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 reported 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 PCHECK 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

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 DOS)

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, enter a terminator (FFH) 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 found (FEH or FFH) then enter a terminator at the last position.

If an extended record is found, and it is not at the 5th position in the extent field, ignore and continue.

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), delink the extension. If bit 7 is not set (not an extension), delink the extension. The second byte in the entended record is a link back to the previous record. If this byte is incorrect, delink the extension.

The de-linking of the files is to protect against possible overlaying another file that does not in fact link back to the primary. By de-linking a file, data past the de-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 re-constructed 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 right next to the directory, and shows this track as allocated in the GAT table. This track is used by the Laredo system 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 the Laredo system, and cannot be detected via the DCT parameters. Due to this situation, PFIX will allocate this track if the following conditions are met:

The drive being fixed must be a Rigid drive.

The GAT fix option must have been specified.

The GAT entry for the cylinder immediately past the directory must be a null byte (not allocated to any files) AFTER the GAT re-build.

The L 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 system, but will cause no other problems. If you are not sure if your rigid system uses this cylinder or not, then use the L option. It is better to lock out a cylinder than crash the system.

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 (*) preceeding the command 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 occuring 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 they 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 DOS. The file 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.

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

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 occured 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, report error 7.

If all 5 extents have been parsed, and no terminator is found (FEH or FFH) then report error 5 and terminate.

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

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), report error 9 and terminate. If bit 7 is not set (not an extension), report error 10 and terminate. The second byte in the entended record is a link back to the previous record. If this byte is incorrect, report error 6 and terminate.

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, report error 4.

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, report error 13 and continue. If the primary directory entry contains a zero HIT byte, report error 14 and continue.

If an extended directory record contains an invalid HIT byte, report error 15 and continue. If an extended record contains a zero HIT byte, report error 16 and continue.

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), report error 12 and continue.

NOTE: Only a single error 12 will be reported for a single file even though there may be several occurances 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 DOS. If an entire disk is specified, 3 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 and 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 is 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, report error 2 and the relative byte of the mismatch. 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, report error 1 and the relative cylinder of the mismatch. 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 system 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 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.

LDOS (tm)LSI
UTILITY DISKS

By: KIM WATT

from

POWERSOFT

a division of Breeze/QSD, Inc.

UTILITY DISK #4

(c)1982

$29.95

Disk #4 Includes:

PFIND/CMD - FIND a string, replace a string, find a byte, find a WORD, replace a byte or word! FIND ANYTHING on the disk FAST! Replace it FAST!

PCOMPARE/CMD - Compare a FILE or a sector to any other file or sector. Detects the minutest differences! Send the outputs from above to video or printer.

LOADING INSTRUCTIONS

- MOD I or III -

1) Prepare a SINGLE density LDOS formatted diskette. This disk may be 35, 40, or 80 tracks, but SINGLE density!
2) Boot the MASTER disk you received in this package in drive 0. You will be prompted to insert the destination diskette, and the files will be "moved" to YOUR diskette with no effort on your part. Single drive owners will need to swap disks a few times.
3) LX-80 users should treat the enclosed MASTER disk as a 35 track single density data diskette.

PCOMP

The PCOMP Utility 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 PCOMP command is:

```
=====================================================
!  PCOMP,*,:a,b,c,d,:e,f,g                          !
!  PCOMP,*,file1,file2,h                            !
!                                                   !
!          *    = output to printer                 !
!          :a   = source drive (Ø-7)                !
!          b    = source track (default Ø)          !
!          c    = source sector (default Ø)         !
!          d    = sector count (default to disk end) !
!          :e   = destination drive (Ø-7)           !
!          f    = destination track (default Ø)     !
!          g    = destination sector (default Ø)    !
!          file1, file2 = any valid LDOS filespec   !
!          h    = starting relative sector (def Ø)  !
!                                                   !
!  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, Sect xxx
At Relative Byte xxx (xxH) for xxx (xxH) byte(s).

Data Mismatch File Relative Sector xxxxx.
At Relative Byte xxx (xxH) 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.

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 compare 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: string
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 PRECEEDING the number with B, D, H or O respectively (default decimal). All numbers indicated must be in the range of Ø-255 (Ø-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 Ø-65535 (Ø-FFFFH). The numbers are placed into the string in LSB, MSB order.

Sample Strings:

"Kim Watt"	becomes	4BH,69H,6DH,2ØH,57H,61H,74H,74H
'Kim Watt'	becomes	4BH,49H,4DH,2ØH,57H,41H,54H,54H
Kim Watt	becomes	4BH,49H,4DH,2ØH,57H,41H,54H,54H
#,1,2,3	becomes	Ø1H,Ø2H,Ø3H
##,1,2,3	becomes	Ø1H,ØØH,Ø2H,ØØH,Ø3H,ØØH

PFIND

The PFIND Utility 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 (Ø-7, colon optional)    !
!        b  = starting cylinder (default Ø)         !
!        c  = starting sector (default Ø)           !
!        d  = sector count (default to disk end)    !
!        filespec = any valid LDOS filespec         !
!        e        = starting relative sector (def Ø)!
!        ffff     = memory start address            !
!        gggg     = ending address (def 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 specified, the entire string must be contained wholly within a sector, no spanning 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.

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

Memory Match at xxxxH

Searching Disk Sectors:

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

Searching File Sectors:

File Match at Relative Sector xxxxx, Relative Byte xxH.

The following are sample command lines:

PFIND,*@:Ø,'kim watt',>"Kim Watt"

Searches drive Ø 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,:Ø,'kim watt'

Searches drive Ø from start to end. Report the total number of occurances of the string 'Kim Watt'. Do not display the locations, just the total.

PFIND,@pfind/cmd,kim watt

Searches the file PFIND/CMD for all occurances of the string 'Kim Watt' in either upper or lower case, and reports their locations to the video.

PFIND,@sysØ/sys,##,h4411

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

PFIND,@hØØØØ,##,h4411

Searches memory from ØØØØH to TOPMEM for the binary word 4411H, and reports all locations to the video.

LDOS (tm)LSI UTILITY DISKS

By: KIM WATT

from

POWERSOFT

a division of Breeze/QSD, Inc.

UTILITY DISK #5 (c)1982

$29.95

Disk #5 Includes:

PREFORM/CMD - Disk Re-Formatter Utility. Fixes CRC errors and "stengthens" the format on older disks. A FORMAT without ERASE utility(5" floppy only)

PVU/CMD - a Disk Verification Utility. Detects bad sectors. Fix them with PREFORM/CMD.

PERASE/CMD - Disk Bulk Eraser (5" floppy only)

LOADING INSTRUCTIONS

- MOD I or III -

1) Prepare a SINGLE density LDOS formatted diskette. This disk may be 35, 40, or 80 tracks, but SINGLE density!
2) Boot the MASTER disk you received in this package in drive 0. You will be prompted to insert the destination diskette, and the files will be "moved" to YOUR diskette with no effort on your part. Single drive owners will need to swap disks a few times.
3) LX-80 users should treat the enclosed MASTER disk as a 35 track single density data diskette.

PREFORM

The PREFORM Utility is a comprehensive Disk Re-Formatter Utility. 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 a corresponding message. If memory is sufficient, then the re-format operation will then 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. It is NOT suggested to press the RESET button to abort the program, 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 so minimal losses will be experienced. 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. (see utility disk #6)

This program will also "freshen" a diskette. The diskettes that are used 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 Utility is a comprehensive Disk Verify 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 (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). 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 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.

PERASE

The PERASE Utility is a Software Bulk-Erase Utility. 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 parameters have 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 the user wishes 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.

LDOS (tm)LSI
UTILITY DISKS

By: KIM WATT

from

POWERSOFT

a division of Breeze/QSD, Inc.

UTILITY DISK
#6 (c)1982

$29.95

Disk #6 Includes:

PCLEAR/CMD- Erases unassigned sectors and Directory slots. Removes ALL traces of "killed" files!

PSS/CMD - Sector Status Utility. Finds WHAT file is assigned to WHAT sector.

PMAP/CMD - Disk-File Mapping Utility. Map a FILE or a disk! Shows exactly WHERE all files are with extents.

LOADING INSTRUCTIONS

- MOD I or III -

1) Prepare a SINGLE density LDOS formatted diskette. This disk may be 35, 40, or 80 tracks, but SINGLE density!
2) Boot the MASTER disk you received in this package in drive 0. You will be prompted to insert the destination diskette, and the files will be "moved" to YOUR diskette with no effort on your part. Single drive owners will need to swap disks a few times.
3) LX-80 users should treat the enclosed MASTER disk as a 35 track single density data diskette.

PSS

The PSS Utility 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 (Ø-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 a corresponding message. The type of drive will be displayed along with the name, date, and number of free granules. 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.

PCLEAR

The PCLEAR Utility 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 granules          !
!          filespec = any valid LDOS filespec       !
!          b   = fill command (default ØØH)         !
!          fill = #cc   = numerical 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 granules will be written to with sectors of the defined fill data.
If the D option was given, then all directory records that do not have bit 4 set (inactive record) 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 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 (ØØH). A single pound sign (#) followed by a one byte numerical value (Ø-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.

Example:
PCLEAR,:Ø,d,s,#h4Ø

Fill all unassigned granules with sectors filled with the byte 4ØH 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 ØØH. 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 Ø.

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.

PMAP

The PMAP Utility is a comprehensive Disk/File Mapping Utility. It allow 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 show the cylinder number, and a list of all the granules on a sector basis. An example of a single density, single sided disk could be:
Cylinder ØØ: ØØ-Ø4xx Ø5-Ø9xx
The first x position will indicate the status of the Allocation Bit associated with gran, and the sectors ØØ-Ø4 indicate the range of sector in the first gran. This position will display a graphic block if th 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 unti the end of file is reached, and an exit is made back to LDOS.

LDOS (tm)LSI
UTILITY DISKS

By: KIM WATT

from

POWERSOFT

a division of Breeze/QSD, Inc.

UTILITY DISK #7

(c)1982

$29.95

Disk #7 Includes:

PMOVE/CMD - A SUPER FAST Multiple Copy Utility!
PDIRT/CMD - Read a Mod III TRSDOS directory from LDOS!
PASSGO/CMD - Removes passwords from A file or a WHOLE disk!
PUN/CMD - UN-REPAIR a disk. (Mod I only)
PEX/CMD - A disk exerciser for head cleaning kits.

LOADING INSTRUCTIONS

- MOD I or III -

1) Prepare a SINGLE density LDOS formatted diskette. This disk may be 35, 40, or 80 tracks, but make sure it is SINGLE DENSITY!
2) BOOT the MASTER disk you received in this package in Drive 0. You will be prompted to insert the destination diskette, and the files will be "moved" to YOUR diskette with no effort on your part. Single drive owners will need to swap disks a few times.
3) LX-80 users should treat the enclosed MASTER disk as a 35 track single density DATA diskette.

PUN

The PUN Utility is a Diskette Un-Repair utility. 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.

PASSGO

The PASSGO Utility 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 Ø. 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 Ø (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)

PMOVE

The PMOVE Utility is a comprehensive file copy utility. 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 (Ø-7)           !
!          :b  = destination drive (Ø-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 with the program name from the DOS command level, or you will be prompted for it. If entered from DOS, 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,Ø1pmove/cmd
but the time saving is really noticed when several files are to be copied:
PMOVE,Ø1/cmdpmove,pmod,pvu,pfilt/flt,passno,pmap
the above command will copy 6 files from drive Ø 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/cmdpmove,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.

PDIRT

The PDIRT Utility is a comprehensive directory utility with the ability to read Mod III TRSDOS formatted diskettes. 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 (Ø-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 ØØ-15 if the corresponding overlay position contains an active pointer.

PEX

The PEX Utility is a Diskette Exercise Utility. 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 (Ø-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 Ø), 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 Ø to the top cylinder
R move the head to cylinder Ø
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 15-3Ø 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 kits 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 Ø disk. Aren't you glad you have PEX?

LDOS (tm)LSI
UTILITY DISKS

By: KIM WATT

from

POWERSOFT

a division of Breeze/QSD, Inc.

UTILITY DISK
8 (c)1982

$29.95

Disk #8 Includes:
PMX/FLT-MX80 graphics
PHELP/CMD -Instant HELP! Very complete. Saves time!
PBOOT/FIX - Customize YOUR LDOS Boot graphics and message!
PFILT/FLT - A User Definable Filter. Very flexible!
DVORAK/FLT - DVORAK/QWERTY Keyboard Filter.
DVORAK/JCL - DVORAK Keyboard Table (used with PFILT/FLT)
CODE/JCL - Keyboard EN-Coding TABLE (used with PFILT/FLT)
DECODE/JCL - Keyboard DE-Coding Table (used with PFILT/FLT)

LOADING INSTRUCTIONS

- MOD I or III -

1) Prepare a SINGLE density LDOS formatted diskette. This disk may be 35, 40, or 80 tracks, but make sure it is SINGLE DENSITY!
2) BOOT the MASTER disk you received in this package in Drive 0. You will be prompted to insert the destination diskette, and the files will be "moved" to YOUR diskette with no effort on your part. Single drive owners will need to swap disks a few times.
3) LX-80 users should treat the enclosed MASTER disk as a 35 track single density DATA diskette.

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.

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 the PBOOT command is:

```
=========================================
!  PATCH,SYSØ/SYS.RSØLTØFF,PBOOT         !
!                                        !
!  abbr: NONE                            !
=========================================
```

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

DVORAK

This is the famous DVORAK keyboard driver that can be used in conjunction with the PFILT driver. 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.

DVORAK

This is the famous DVORAK keyboard driver. 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 Ø. The SHIFT Ø 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.

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                      !
!                                      !
!  abbr: NONE                          !
========================================
```

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.

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
b11011011=o177
hff=hfe
:A=a
:Q=B
```

CODE/DECODE

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 command is:

```
==============================================
!  FILTER,*KI,PFILT,CODE                      !
!  FILTER,*DO,PFILT,DECODE                    !
!                                             !
!  abbr: NONE                                 !
==============================================
```

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

LDOS (tm)LSI UTILITY DISKS

By: KIM WATT

from

POWERSOFT

a division of Breeze/QSD, Inc.

UTILITY DISK #2 (c)1982

$29.95

Disk #2 Includes:

PMOD/CMD - A Disk-File-Memory Modification Utility.

PMOD,aaaa (display memory address)
PMOD,filespec,aaaa (file to display,starting rel. sector)
PMOD :b,c,d (drive,cylinder,sector to display)

UTILITY DISK #3 (c)1982

$29.95

Disk #3 Includes:

PCHECK/CMD - LDOS Directory Check Utility. Checks a drive or a file, (GAT, HIT, BOOT, etc.) for integrity.
PFIX/CMD - FIXES problem discovered by PCHECK/CMD. You may repair GAT or HIT table, repair BOOT sector, or repair FILE records. This program might fix that crashed disk that has you going nuts!

UTILITY DISK #4 (c)1982

$29.95

Disk #4 Includes:

PFIND/CMD - FIND a string, replace a string, find a byte, find a WORD, replace a byte or word! FIND ANYTHING on the disk FAST! Replace it FAST!
PCOMPARE/CMD - Compare a FILE or a sector to any other file or sector. Detects the minutest differences! Send the outputs from above to video or printer.

UTILITY DISK #5 (c)1982

$29.95

Disk #5 Includes:

PREFORM/CMD - Disk Re-Formatter Utility. Fixes CRC errors and "stengthens" the format on older disks. A FORMAT without ERASE utility(5" floppy only)
PVU/CMD - a Disk Verification Utility. Detects bad sectors. Fix them with PREFORM/CMD.
PERASE/CMD - Disk Bulk Eraser (5" floppy only)

UTILITY DISK #6 (c)1982

$29.95

Disk #6 Includes:

PCLEAR/CMD- Erases unassigned sectors and Directory slots. Removes ALL traces of "killed" files!
PSS/CMD - Sector Status Utility. Finds WHAT file is assigned to WHAT sector.
PMAP/CMD - Disk-File Mapping Utility. Map a FILE or a disk! Shows exactly WHERE all files are with extents.

UTILITY DISK #7 (c)1982

$29.95

Disk #7 Includes:

PMOVE/CMD - A SUPER FAST Multiple Copy Utility!
PDIRT/CMD - Read a Mod III TRSDOS directory from LDOS!
PASSGO/CMD - Removes passwords from A file or a WHOLE disk!
PUN/CMD - UN-REPAIR a disk. (Mod I only)
PEX/CMD - A disk exerciser for head cleaning kits.

UTILITY DISK #8 (c)1982

$29.95

Disk #8 Includes:
PMX/FLT-MX80 graphics
PHELP/CMD -Instant HELP! Very complete. Saves time!
PBOOT/FIX - Customize YOUR LDOS Boot graphics and message!
PFILT/FLT - A User Definable Filter. Very flexible!
DVORAK/FLT - DVORAK/QWERTY Keyboard Filter.
DVORAK/JCL - DVORAK Keyboard Table (used with PFILT/FLT)
CODE/JCL - Keyboard EN-Coding TABLE (used with PFILT/FLT)
DECODE/JCL - Keyboard DE-Coding Table (used with PFILT/FLT)