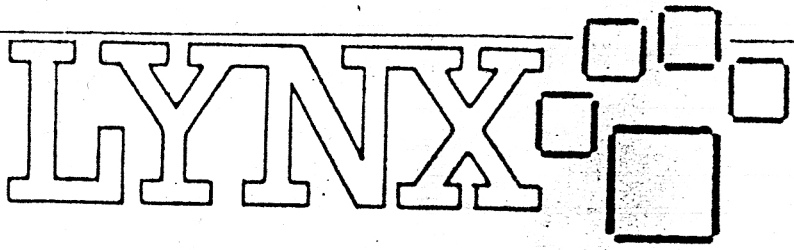
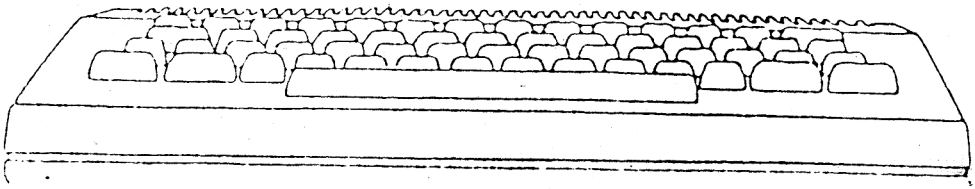


LYNX UTILITIES DISK 1



LYNX UTILITIES DISK

The disk contains the following files.

U — Loader for extra DOS commands contained in UTIL.OBJ. To load these commands, the utilities disk must be on the default drive. Then type:-

```
EXT MLOAD "U"
```

This will move HIMEM down to &EC00 and place the extra code above it. The extra Basic commands are :-

```
EXT SCOPY
EXT COPY
EXT CREATE
EXT OPEN
EXT CLOSE
EXT PUT
EXT GET
```

UTIL.OBJ — Code for the above routines. It will be loaded via EXT MLOAD "U".

DISKED — Disk editor.

TAPEDISK — Tape-to-disk copy program.

U — Disk Utility Program

COMMANDS FOR DISK UTILITIES

SCOPY and COPY

These two programs copy files in a single (SCOPY) and a double drive (COPY) system. Syntax is as follows: (assuming that the disk utility program "U has been loaded)

1. EXT SCOPY "*filename*"

Where *filename* is a compulsory file-name parameter. If *filename* is omitted (i.e. EXT SCOPY"") then all the files in drive A will be copied to another disk in drive A. The user will be prompted when it is necessary to change disks.

2. EXT COPY "*source:filename*", "*object:*"

'*source*' is the source drivename (i.e A, B, C or D). '*object*' is the destination drivename. '*source*' must not be the same as '*object*'. '*filename*' is the filename parameter; if '*filename*' is omitted, then the entire source disk will be copied to the entire destination disk.

'Wildcard' characters

It is not necessary to fully specify the filenames in either of these commands. The following two rules are used:

1. A '?' character matches any character in the corresponding position of the filename spec.

2. A '*' character terminates the comparison process. For example, if '*filename*' is specified as GRAPH*, then any of the following files will be treated as the source file: GRAPHIC, GRAPHABC, GRAPH, and so on.

If, when using either "COPY" or "SCOPY", a duplicate file name is found on the object disk, the following option is presented:

File already exists : Overwrite (Y/N)?__

Typing 'Y' results in the object file being overwritten.
Typing 'N' results in the source file being skipped over.

3. EXT CREATE 0, <*filename*>, record size (, no. of records)

Only channel 0 is implemented ; (0 can be an expression which evaluates to zero
Record size in range 0 - 32767
No. of records defaults to zero.

Filename may be represented by a string, string variable or a string array.

The command closes the file after creating it. It may be used in calculator or program mode.

4. EXT OPEN 0, <*filename*> (,NVAR1(,NVAR2))

Filename: as above
NVAR1: Numeric variable to receive record size.
NVAR2: Numeric variable to receive number of records.

The file remains open after execution of this command. A wrong mode error will arise in immediate mode.

5. EXT CLOSE 0
Rewrites the header with updated random file information so an already open file is assumed; the file is then closed. Immediate mode is allowed.

6. EXT PUT 0, record number, *<where to pick up data for record>*
Only channel 0.
Record number must be in range 0 - 32767.
<WHERE> may be a string, string variable or array, or a string expression. It may also be an address from which the data is to be picked up.
In the latter case, the record size determines the number of bytes transferred; if the data is later to be downloaded into a string variable, then a carriage return should be inserted as a terminator by the user.
In the former cases, where data is transferred from BASIC strings, string size and record size are compared. Where the latter is larger, the string is simply transferred along with the terminating carriage return; where the string will not fit into the record, then as much as possible is transferred, the last position being finished with a carriage return.
The largest record number written to the file is updated if necessary and is available to BASIC at location (Last).
The updated file information is only written to disk on the close command.
N.B. 'PUT' only works in program mode.

7. EXT GET 0, record number, *<where to place data from record>*
Channel 0 only
0 - 32767 for record number
<WHERE>: String variable, string array, or address where data is to be deposited from. If the record is larger than the string variable then only the size of string bytes are transferred and where the record is smaller than the string, a terminating &0D should be present to prevent the appearance of non relevant information. Where records are written to disk from a specified address, it is the user's responsibility to insert the &0D if 'garbage' suppression is desired.

The command only works in program mode and it leaves the file open.

Several zeroised records exist after the largest record written to the disk file by the user; no error is signalled in attempting to access these.

8. EXT RINFO *<filename>*
Adds on some extra file information to that given by the EXT INFO Command.
Works in both immediate and program mode but it assumes that the file is closed. A disk system error will result from using the command on an already open file.

9. EXT SEARCH 0,J, *<looking for?>*, *<where to put it?>*
Channel 0 only
The 'J'th match will be searched for, looking sequentially through the file.
<looking for?>: string, string variable or array. So called 'Wild Card' format may be used, i.e. '*' will be considered as matching
The string must be less than 30 characters in length
<where to put it>: same parameters as *<looking for?>*.

RAM LOCATIONS:
Channel: 0 (&CFFA), two bytes; irrelevant
Size: 0 (&CFFE), two bytes: record size; used by GET and PUT commands. Size is written from HDOUT+8. The latter is written to disk file on EXT CLOSE.
Last: 0 (&CFFC), two bytes: last record; written from HDOUT+10. Latter goes to disk on CLOSE.

DISKED — Lynx Disk Editor

The version supplied on the utilities disk runs under LYNX BASIC. This version can read and write both LYNX DOS disks and CP/M disks. Functions supported are:

(1) Read a disk sector (512 bytes) into RAM
(2) Read several consecutive sectors of a track
(3) Display and modify RAM
(4) Write a sector from RAM to disk

The editor can be used to display and modify any part of RAM above &6000. For convenience a block of 512 bytes is designated as the sector buffer, and the default parameters of most RAM commands refer to this buffer. The user can change the start address of the buffer.

LOADING

Type EXT MLOAD "DISKED"

The editor is loaded at &6B00 and ends at about &7600. The buffer address at startup is &8000.
When running under 128K LYNX BASIC, the program will work with FAST mode either on or off. The use of FAST ON is recommended.

NOTE:- Before loading "DISKED" put the LYNX into text mode. This will reduce frustration by speeding up the display writing.

SCREEN EDITING

The editor displays a block of RAM on the screen (128 bytes on the 96K LYNX, 256 bytes on the 128K LYNX). Bytes are displayed on the screen in similar format to the LYNX MONITOR, as hex digits on the left and ASCII characters on the right. There is a cursor whose position is shown by displaying the character in inverse video. If any command moves the cursor off the screen, a new block of RAM is displayed so that the cursor is again on the screen. Except for the D command the difference between the old and new screen addresses is always a multiple of the screen size. E.g. if the screen displays &9F00 to &9FFF with the cursor at &9FFF, then stepping the cursor forward will cause &A000 to &A0FF to be displayed.

CURSOR COMMANDS

The following commands move the cursor and will be accepted at any time.

<RIGHT ARROW> Forward one byte

<LEFT ARROW> Backward one byte

<DOWN ARROW> Forward one screen line (8 bytes on the 96K, 16 on the 128K)

<UP ARROW> Backward one screen line

*<CTRL>*A Move to top left of screen

*<CTRL>*Z Move to bottom right of screen

*<CTRL>*T Toggle cursor between hex and ASCII portions of screen (for use with E)

COMMAND LINES

Command lines typed at the keyboard are echoed at the bottom of the screen. The *<DELETE>* key may be used. The command is executed when *<RETURN>* is typed.

Commands consist of a single character followed by up to 3 parameters. Certain parameters may be omitted and then defaults will be supplied. In these notes, optional parameters are enclosed by { }. Ram addresses and byte counts are typed as hex numbers using digits only. The following characters may also be used.

/ current start of buffer
. current cursor position

The current values of / and . are shown on the screen.

Two hex numbers on the command line must be separated by at least one space. Elsewhere spaces are optional.

EDIT COMMANDS

Besides the cursor commands the following are available:

. *<address>*
Moves the cursor to the given address, displaying a new block of RAM if necessary.
Default: start of buffer

D *<address>*
Moves the cursor to the given address, displaying a new block of RAM if necessary.
Default: start of buffer

E
Enters values into RAM starting at the current cursor position. The value of a byte may be entered either as two hex digits or as an ASCII character, depending on which portion of the screen contains the cursor (see *<CTRL>*T. Non-printing characters cannot be entered in the ASCII mode. The cursor steps forward automatically as each byte is typed in.

<RETURN>
Used after E to end entry mode and return to command mode.

RAM COMMANDS

/ *<address>*
Set the start of the 512-byte sector buffer to the given address.
Default address: current cursor position.

R source. {dest. *<byte-count>*}
Reads a block of RAM from source to destination.
Default destination: start of buffer.
Default byte count: the last byte of the destination is the last byte of the buffer.

W dest. {source *<byte-count>*}
Writes a block of RAM to destination from source.
Default source: start of buffer.
Default byte count: the last byte of the source is the last byte of the buffer.

(R and W perform the same function, but have different default parameters. The copying is "intelligent", i.e. it functions correctly when the source and destination blocks overlap).

F data (dest. <*byte-count*>)
Fills the destination block with the given data byte.
Default destination: start of buffer.
Default byte count: the last byte filled is the last byte of the buffer.

V source (dest. <*byte-count*>)
Verifies that the source and destination blocks are identical. If a difference is found, moves the cursor to the first differing byte of the source.
Default destination: start of buffer.
Default byte count: the last byte checked is the last byte of the buffer.

L <*string*> (addr. <*byte-count*>)
Locates the first occurence of the given string, starting the search at the given address. The byte count specifies the size of the block in which the search is to take place. If the search is successful the cursor moved to the start of the located string. The string can be specified in two ways:

1. As HEX bytes enclosed by < >, for example L<FE 12 00 06>

2. As ASCII characters enclosed by " ", for example L"BASIC"8000 1000

The string can contain from 1 to 32 bytes.
Default string: the string last used.
Default address: current cursor position.
Default byte count: search continues to the end of RAM.

DISK COMMANDS

The sector to be read or written can be specified in two ways:

1. <*drive*> <:*track*> <;*sector*>
Here the track and the sector within that track are entered in decimal. Note that track numbers start at 0 but sector numbers start at 1. The editor adapts this convention so as to be consistent with the sector header format stored on the disk.

The default drive, track and sector are shown on the screen. They are automatically changed by each G, P or T command.

2. <*drive*> <*S disk sector*>
<*drive*> <*H disk half sector*>
<*drive*> <*Q disk quarter sector*>
Here the sector (or half or quarter sector) is input as a hex number. The numbering starts at 0 and runs through the whole disk. Note that the H and Q formats still define an entire sector to be read or written. The sector is the one that contains the given part sector.

For example the following all define the same sector.

:1;7 S10 H20 H21 Q40 Q41 Q42 Q43

Semi Colon. Brackets Not needed

G — G (sector <*dest.*>) G:(drive):(track number):(sector number)

Gets a sector (512 bytes) from disk and stores it in RAM starting at the destination address. The given drive, track and sector become the new defaults. If the H or Q format is used, the cursor is moved to the start of the part sector; otherwise to the start of the sector.
Default sector: as shown on screen
Default destination: start of buffer

P — P (sector <*source*>) P:(drive):(track number):(sector number)
Puts a sector (512 bytes) onto disk, copying it from RAM starting at the source address.
The given drive, track and sector become the new defaults.
Default sector: as shown on screen.
Default source: start of buffer.

M — M (sector <*dest.*>) M:(sector number)
Multiple get - gets a group of sectors from disk, starting at the given sector and ending at sector 10 of the same track. Otherwise like G.

Q CP/M BLOCK
Shows which quarter sector starts the given CP/M block. Each CP/M record is 128 bytes long (i.e. a quarter sector). Eight records form a 1K block. The blocks are numbered consecutively starting at 0. Block 0 is placed in track 2, sectors 1 and 2.

The block number should be typed in hex, and the program will display in hex the number of the first quarter sector in that block. E.g. Q54 will display &02F0, showing that CP/M block 54 hex comprises quarter sectors &02F0 to &02F7.

EXIT COMMAND

X
On exit from the editor the disk operating system is reinitialized. This is done because the DOS has no record of any changes made to the disk controller registered by the G, P and M commands.

AN EXAMPLE OF USING "DISKED" (Specially useful to CP/M users).

Note: Before loading "DISKED" put the LYNX into text mode. This will reduce frustration by speeding up the display writing.

When in "DISKED" type

>G :1;3 /RETURN/ NB COLON 1st SEMICOLON 2nd.

This results in "DISKED" loading sector 3 of track 1 from disk. With a CP/M system disk loaded, the first 256 bytes of the BIOS will be displayed on the screen. (The BIOS is loaded at E700 when CP/M is first BOOTED up.) In fact, 512 bytes are loaded from a 200K drive, addresses 8100 to 81FF contain the remaining 256 bytes.

ADDRESS	HEX DATA	ASCII DATA
8000	C3 03 E7 C3 68 E7 C3 49 EC C3 37 EC C3 46 E7 C3	XXXXXXXXXXXXXXXX
8010	62 EC C3 5B EC C3 52 EC C3 22 E8 C3 40 E8 C3 2E	XXXXXXXXXXXXXXXX
8020	E8 C3 33 E8 C3 38 E8 C3 70 E8 C3 79 E8 C3 34 EC	XXXXXXXXXXXXXXXX
8030	C3 3D E8 87 EC 54 00 14 01 C0 FF C0 FF 40 04 18	XXXXXXXXXXXXXXXX
8040	F6 54 F6 90 F6 04 F5 XX XX XX XX XX XX XX XX XX	XXXXXXXXXXXXXXXX
8050	XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX	XXXXXXXXXXXXXXXX
8060	XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX	XXXXXXXXXXXXXXXX
8070	XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX	XXXXXXXXXXXXXXXX
8080	XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX	XXXXXXXXXXXXXXXX
8090	XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX	XXXXXXXXXXXXXXXX
80A0	XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX	XXXXXXXXXXXXXXXX
80B0	XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX	XXXXXXXXXXXXXXXX
80C0	XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX	XXXXXXXXXXXXXXXX
80D0	XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX	XXXXXXXXXXXXXXXX
80E0	XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX	XXXXXXXXXXXXXXXX
80F0	XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX	XXXXXXXXXXXXXXXX

The X's simply demonstrate idleness. They represent data that is not discussed here. It should be left intact as read from the disk. Interested parties can investigate the data themselves.

The data at address 8035 (E735) is the CP/M IOBYTE. This was 94 on or e distribution version of the LYNX BIOS. It was changed to 54 to permanently invoke the serial printer as the standard list device, e.g.,

LST:=CRT:

Similarly, if drives B: and C: are to be 800K devices, then change the DRVTYP byte at 8037 (E737) to 14.

"DISKED" makes this sort of modification simplicity itself.
To make such changes enter E at the prompt thus:-

>E /RETURN/

Now simply move about the data, using the cursor control keys, typing appropriate changes. Press /RETURN/ to return to the command level when you have finished.

Write the data back to disk by entering P at the prompt.

Finally enter X to return to the normal operating system.

CAUTION It's easy to destroy data on disks using powerful utilities such as this. Always check the data before writing it to the disk.

TAPEDISK — Tape-to-Disk Copy Program

The purpose of the TAPEDISK program is to enable easy copying of files from cassette tape to disk. Files are read from tape into a fixed buffer regardless of the start address specified on the tape. The start address and execution address (if any) are however copied to disk along with the file. The file type BASIC, machine code, data store, or other) is also copied to disk. Thus when the file is loaded back from disk it behaves just like the tape file.

LOADING

Type EXT MLOAD "TAPEDISK"
The program loads at &6B00 and ends about &7040.

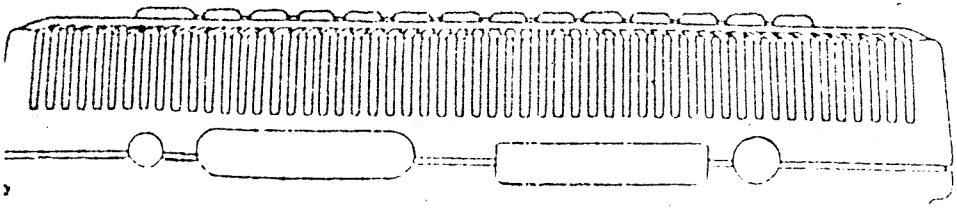
COMMANDS

Commands are just a single letter followed by *<RETURN>*. The program will then prompt for the information required.

F: File copy
Prompt is for name of file as held on tape. The file is then copied to disk on the current drive. Quotes round the filename are optional. If *<RETURN>* is typed without any filename, all files on the tape will be copied. Normally the disk file is given the same name as the tape file. If the name of the tape file contains characters that are illegal in a disk file name (namely any characters other than letters, digits and ".") then those characters will be converted to ".".

T: Tape speed
Prompt is for the new tape speed (0 to 5).

D: Disk drive
Prompt is for the new drive letter (A to D). Quotes round the drive letter are optional.

C: Comment
Prompt is for new comment to be added to the disk files. Quotes round the comment are optional.

X: Exit
End.