LS-DOSÔ 6.3 UPDATE FOR TRSDOSâ 6.2.X

The *LS-DOS* 6.3 release is an upgrade for the TRSDOS 6.2 operating system. Several important changes have been made to extend and enhance the operating system and its utilities. The date ranging has been expanded to accept dates through the year 1999. Files will now carry a modification time as well as a date. The *DATECONV/CMD* program is provided to translate version 6.2 or earlier disks to the 6.3 style dating. The user password has been eliminated from the system. The owner password still remains. The library command *ID* was added to display a customer service number. Several enhancements have been made to *BASIC*. The new *DISKCOPY/CMD* program will duplicate 5" double density floppy disks.

Because the *LS-DOS 6.3* update is a series of enhancements to *TRSDOS 6.2*, the primary documentation remains the 6.2 manual and Technical Reference manual. If you have a version of TRSDOS earlier than 6.2, you can obtain the manuals from Radio Shack under the catalog numbers 26-0316 *(TRSDOS Version* 6 [6.2 DOS manual and disk]), 26-2110 *(Model 4/4D* [6.2] *Technical Reference Manual),* or 26-1117 (6.2 DOS manual only). This documentation should be treated as an addendum to the TRSDOS 6.2 information.

LS-DOS 6.3 installation instructions

Before performing the upgrade, it is recommended that you make several backup copies of the 6.3 master disk. The simplest way to do this is to boot your system using the 6.3 diskette, insert a blank diskette to receive the copy in drive 1, and type the command:

```
DISKCOPY :0 :1
```

When the copy finishes, you can insert another destination diskette and make another copy.

IMPORTANT NOTE

It is important that you *DO NOT* switch between version 6.3 and any earlier version system disk in drive 0 without rebooting the computer.

To update your existing TRSDOS 6.2 disks, boot up with a fresh copy of your new 6.3 system disk. The 6.3 system disk should remain in drive 0 throughout the update procedure. There are two separate methods for updating 6.2 floppy disks, depending on whether they are system disks or data disks.

Updating DATA disks

Place the DATA disk in drive 1 and issue the command:

```
DATECONV :1
```

This completes the conversion of the data disk.

Updating SYSTEM disks

Updating system disks may take two or three steps. Place the 6.2 disk in drive 1, and issue the command:

```
BACKUP :0 :1 (I,S,OLD)
```

If you wish to use the new BASIC enhancements, type the command:

```
BACKUP BASIC:0 :1 (I)
```

Once the backups are completed, give the command:

```
DATECONV :1
```

This completes the updating of a system disk. At this time, you can add any of the new utility files you wish.

IMPORTANT NOTE: If you have sysgened a configuration file, you will have to re-create it the next time you boot the disk.

Updating HARD disks

For most hard disk installations, the following instructions can be used. Boot up the hard disk as you normally would. Typically your system will have the hard disk as drive 0, and the first floppy drive as drive 4. Place the 6.3 disk in drive 4, and type the following commands:

```
BACKUP :4 :0 (I,S)
DATECONV :0
```

Remove the 6.3 disk from drive 4, and insert your normal hard disk boot diskette. Type the following commands:

```
BACKUP :0 :4 (I,S,OLD)
DATECONV :4
MEMORY (A=X'3B',B=0)
MEMORY (A=X'85',B=99)
MEMORY (A="Y",B=255)
SYSGEN (D=4)
```

Now, reboot the system and use *DATECONV* on your other 6.2 hard drive partitions.

IMPORTANT NOTE: If your hard drive configuration has system files sysgened in memory, you will have to re-create the configuration file after putting the 6.3 operating system on your hard drive and on your booting floppy disk.

Reformatting hard drives

If you ever have to reformat a hard drive using the *TRSFORM6/CMD* program that comes with TRSDOS 6.2, you should use the *DATECONV* utility after formatting is complete. This will make sure the hard drive is marked as being version 6.3. Alternatively, you can apply the following patch to *TRSFORM6/CMD* that will permanently alter it to format version 6.3 partitions.

```
.Patch TRSFORM6/CMD.UTILITY for 6.3 compatibility
D04,C0=88
F04,C0=80
D05,01=63
F05,01=62
D09,2B=33
F09,2B=32
D0D,AA=63
F0D,AA=62
```

VERSION 6.3 UPDATE INFORMATION

This section describes the changes to existing commands, and explains the new commands and utility programs. You can continue to use your 6.2 disks with version 6.3 as long as you update them as explained in the installation instructions. The following is a brief guide to version compatibility.

USING 6.2 OR EARLIER DISKS WITH 6.3 - Disks from earlier versions of TRSDOS should be converted with the new *DATECONV/CMD* program before being used with 6.3. Unconverted disks, will work, but the time and date information will not be correct. If a disk that was written on by 6.3 is used with an earlier version DOS, it may appear to have an unknown user password. If the file was previously password protected, access may not be possible without knowing the owner password. To prevent this, use the COPY library command with the (CLONE=NO) parameter when moving from version 6.3 to earlier versions.

USING 6.3 DISKS WITH EARLIER VERSIONS - This normally should not be done if the disk is to be used again with 6.3, as the year and time information will not be correct if the disk is written on.

USING LDOS DISKS WITH 6.3 - Files can be moved from LDOS 5.1 disks to 6.3 disks with either the COPY library command or with a backup by class. When moving files from 6.3 to LDOS 5.1 disks, with the COPY command, the parameter (CLONE=NO) should be used. The *DATECONV* program will not work on LDOS 5.1 disks.

Boot up changes

Since version 6.3 now stores a modification time as well as a date, you will be prompted for the time when booting the computer. You can enter the hours, minute, and seconds, or just the hours and minutes. Pressing only [ENTER] for this prompt will start the time at 00:00:00. If you wish to suppress the time prompt, give the command SYSTEM (TIME=OFF) once the DOS Ready prompt appears. If you wish to reactivate the prompt after you have turned it off, use the command SYSTEM (TIME=ON). These commands do not have to be sysgened.

Library command changes

ATTRIB - The user password field has been removed from all files, so the USER parameter is no longer accepted. The system now assumes a blank user password for all files. If a protection level has not been assigned to a file, full access will automatically be granted regardless of any owner password. The owner password will still be required for full access on password protected files that have a protection level other than FULL. To have a file that allows no access whatsoever without the use of the owner password, change the protection level to NO (or NONE). Although the documentation for the *ATTRIB* library command lists EXEC as the highest protection level, the use of NO is valid and provides the highest level of protection.

COPY - When copying from a 6.2 or earlier version disk to a 6.3 disk, the old user password (if any) will be removed and the 6.3 style date/time information field will be established, with the time being set to 12:00am.

DATE - The acceptable range of dates is from January 1, 1980 to December 31, 1999.

DIR - The directory display now shows a mod time immediately following the mod date for all 6.3 diskettes. The time will be in 12-hour format, with AM or PM indicated. Disks from earlier versions of TRSDOS will display a blank time field.

ID - A new library command that displays a customer service number. This number is installed when a master disk is manufactured, and is linked to the serial number that displays on boot up and to other embedded numbers. You must have this number when calling or writing for customer service on LS-DOS 6.3.

TIME - The time may now be entered with the seconds being optional. The full syntax for the TIME command is:

TIME [*HH:MM*[*:SS*]] [(CLOCK=*ON*|*OFF*)]

Utility Program changes

BACKUP - When doing a backup by class from a 6.2 or earlier version disk to a 6.3 disk, the user password (if any) will be removed and the 6.3 style date/time information will be established. It will *NOT* be permissible to backup SYSTEM files from a 6.2 or earlier disk to a 6.3 disk during a backup by class.

DATECONV/CMD - Converts pre-6.3 version data (non-system) disks to use the new time/date format. The syntax is:

DATECONV :*d*

There are no parameters for this program. If conversion of a SYSTEM disk is attempted, an appropriate error message will be displayed. To convert this type of disk, you must first use *BACKUP* to move 6.3 system files onto it before using *DATECONV*. For example,

BACKUP :*s* :*d* (S,I,OLD)

Data disks from versions earlier than 6.3 will be converted in all cases and marked as a 6.3 type disk.

Conversion of LDOS 5.1 formatted disks is not allowed. Although these disks can be read and written to, the time and year *will* not be updated on modified files. Files should be copied or backed-up onto 6.3 formatted disks.

***DATECONV* does not normally need to be used on a 6.3 disk, since the *COPY* and *BACKUP* commands automatically adjust the time/date storage when moving from LDOS or TRSDOS 6.x to 6.3. For this reason you should *NEVER* move files to a 6.3 disk when booted up under any earlier version. If you do use *DATECONV* on a 6.3 disk, only those files that have an old style blank user password will be changed. This is to protect the time stamp on proper 6.3 files.**

DISKCOPY/CMD - Performs a single-pass format and copy of a 5" double density floppy disk. It will duplicate both single and double sided disks. Attempting to copy other types of disks will abort the program with an appropriate error message. The syntax is:

DISKCOPY :*s* :*d*

The *:s* and *:d* are the *source* and *destination* drive numbers, respectively, and cannot be the same drive. After starting *DISKCOPY, you will* be prompted to insert the source and destination disks.

After the copy is complete, you will be prompted to do another copy. If you wish to copy another disk, press the [Y] key. You can change both the source and destination disks at this point if you desire. Pressing [N] will exit back to `DOS Ready`.

If an error is encountered during the copy, the program will display the error and return to `DOS Ready`.

TED - ASCII Text Editor

The Text EDitor (TED) is a full screen "quick" text editor with typical word-processing type features (four-directional cursor movement; bi-directional scrolling; text directional delete; large text buffer; etc); however, TED was not designed to be a full featured word processor. TED was designed for you to be able to rapidly enter a full-screen text editing environment while accomplishing many of your text file editing tasks.

Summary of editing commands

The following are the command keys and their functions as used in TED. Once you become familiar with the operation of TED, this section may be all you need to refer to from to time to time to jog your memory.

Action	Key Entry
Move the cursor one position left	[¬] or [CTRL] [H]
Move the cursor one position right	[®] or [CTRL] [I]
Move the cursor one position down	[¯] or [CTRL] [J]
Move the cursor one position up	[-] or [CTRL] [K]
Move the cursor to the beginning of the line	[SHIFT] [¬]
Move the cursor to the end of the line	[SHIFT] [®]
Move the cursor to the end of the text	[SHIFT] [¯]
Move the cursor to the beginning of the text	[SHIFT] [-]
Toggle overstrike/insert modes	[CTRL] [A]
Specify a BLOCK	[CTRL] [B]
Specify DELETE	[CTRL] [D]
FILE the text buffer to disk	[CTRL] [F]
Go find the next search string match	[CTRL] [G]
LOAD a text file into the buffer	[CTRL] [L]
Command confirmation or advance to next line	[ENTER] or [CTRL] [M]
Go to the NEXT video page	[CTRL] [N]
REPLACE searched string with new string	[CTRL] [R]

SEARCH for a string	**[CTRL] [S]**
Go UP to the previous video page	**[CTRL] [U]**
EXIT the text editor	**[CLEAR] [SHIFT] [=]**

Invoking TED

TED is invoked via the command:

```
TED
```

TED will display a welcome message on the bottom line of the display. This line will also be used for the display of status, prompting, and error messages. TED displays three different types of messages during its operation. Error messages are indicated by a trailing exclamation point, `!`. Queries which need a response are indicated by a trailing question mark, `?`. All other messages are informative messages. Thus, `Marker!` is an error, `String?` is a query, and `Block` is informative.

Text entry modes

TED will accept only displayable ASCII characters in the range 20H through 7FH for text entry. Any other character will be interpreted as a command entry.

TED operates in two text entry modes: overstrike and insert. The initial mode established when TED is first invoked is the "overstrike" mode. While TED is in "overstrike" mode, it will use an underscore as the cursor character. When you toggle to "insert" mode, the cursor is changed to a full graphics block. You toggle from one mode to the other via the [CTRL] [A] command.

When TED is in "overstrike" mode, any acceptable character that is typed is written over the character which appears under the blinking cursor. You can overstrike a newline character (i.e. [ENTER], which is displayed as small graphics block). You can also overstrike either a "begin" block marker or an "end" block marker. You can be in overstrike mode when you come to the end of the text (or starting from an empty text buffer, for that matter) and still be able to enter text in this mode.

When you switch to "insert" mode, anytime you enter acceptable text, the existing text will be pushed down one position starting from the character under the cursor to make room for the inserted character. The display will be constantly updated as text is inserted.

The text entry mode is only changed via the [CTRL] [A] command. Going into "delete" mode does not change the mode of text entry.

Loading a text file

The [CTRL] [L] command is used to load a text file into TED. When you enter [CTRL] [L], you will be prompted for the name of the file. If the extension is omitted from your entry, "/TXT" will be automatically provided.

The LOAD command will not automatically clear any text remaining in the text buffer prior to the LOAD. The new text is not inserted at the cursor position but rather is appended to the end of the current text. If you wish to load the new file over the old text, simply invoke the command sequence, [SHIFT] [-] followed by [CTRL] [D] then [SHIFT] [¯]. This will delete the entire text buffer.

If the file is too large to fit into the available text buffer, the error message `No room!` will be displayed and no text will be loaded. If any disk read error is encountered while reading the text file into the text buffer, the message `I/O error!` will be displayed. The text which was loaded up to the point of encountered the error will be retained in the text buffer.

Entering text

Entering text is easy, you just type away. If you already have text in the buffer and wish to enter new text at the end of the document, just move the cursor to the bottom (via the [SHIFT] [¯] key), then type in your text. If you wish to enter text at some other point, just position the cursor, toggle to the *insert* mode, then type away. TED will stay in *insert* mode until explicitly toggled back to *overstrike* mode.

As you are entering text, any word which is too long to fit at the end of a video line will be split at the 80th column and continued onto the next line. These long words are not automatically moved onto the next line, as is the case with the typical word processor.

Cursor positioning manipulations

The ARROW keys are the primary tools to move the cursor. They will move the cursor in the direction indicated by the arrow. The shifted keys will be interpreted as cursor movement commands unless TED is in the DELETE or BLOCK modes. The [SHIFT] [¬] command will move the cursor to the first column of the current line. The [SHIFT] [®] command will move the cursor to the last character of the current line. You can position the cursor to the first character of the text buffer by using [SHIFT] [-]. Finally, [SHIFT] [¯] positions to the end of text.

The page up, [CTRL] [U], command will redraw the video screen so that the first displayed line is twenty one lines previous to the current first displayed line. The page next command, [CTRL] [N] , will redraw the video screen so that the first line displayed is the last line of the current displayed text. If the video display has fewer than 22 lines of text displayed, the page next request will be ignored.

Text deletion

TED provides five forms of text deletion in addition to the block deletion discussed later. To delete the single character which appears under the cursor, invoke the delete command via [CTRL] [D]. This action will get rid of the character, and all text which follows that character will be pulled back one position. The command also puts you into DELETE mode which is made apparent by the display of the word Delete in the status line. The DELETE mode is active for only the next keyboard entry. There are only four subcommands associated with the DELETE mode: delete to beginning of line (bol), delete to end of line (eol), delete to top, and delete to bottom. TED will always prompt before performing one of these operations.

Deletion desired	Command sequence
delete to bol	[CTRL] [D] then [SHIFT] [¬]
delete to eol	[CTRL] [D] then [SHIFT] [®]
delete to top	[CTRL] [D] then [SHIFT] [-]
delete to bottom	[CTRL] [D] then [SHIFT] [¯]

After typing [CTRL] [D], the character now under the cursor is the character which was to the right of the deleted character. Since in the case of delete to bol and delete to top, you are deleting text which is in front of the cursor, you really don't want to delete the character which is under the cursor after the FDI. Well, you don't have to worry about that because those two subcommands properly back up one position before continuing the deletion.

Block operations

The BLOCK command, [CTRL] [B], has five subcommands: Begin, End, Copy, Delete, and Move. These subcommands are specified by entering the first letter of the subcommand word ([B], [E], [C], [D], or [M]). The letter may be in either upper or lower case. Note that these subcommands are NOT control key combinations but normal alphabetic single-key entries. When you invoke the BLOCK command, the word `Block` will be displayed in the status line.

Any time you need to deal with a block; say to copy it, move it, or delete it, you have to first mark it. The beginning and ending positions of a block are marked by first positioning the cursor over the first character of the block and then entering the two command sequence, [CTRL] [B] followed by [B]. This is followed up by positioning the cursor over the character immediately following the last character of the block and then entering the sequence, [CTRL] [B] followed by [E]. The beginning positioning will be indicated on the display by a "begin" marker which is inserted by TED into the text. The marker is displayed as a graphic left bracket. The ending position will be indicated on the display by an "end" marker which is also inserted by TED into the text. The marker is displayed as a graphic right bracket. These markers occupy ordinary text positions; thus they may be deleted or overstriked. Any remaining in the text buffer at the time a FILE command is performed will be written to the disk file just as if they were ordinary text characters.

Although you can mark as many blocks as your heart desires, TED provides no way to differentiate between marked blocks in other than the BLOCK-DELETE function. For copying and moving blocks, the first block marked in the text is the one chosen for copying or moving. On the other hand, a BLOCK-DELETE request requires that the cursor be positioned within the interior of the marked block which is to be deleted.

To COPY the first marked block in the text to some other position, simply mark the beginning and end of the block as discussed above, move the cursor to the position in the text where you want the marked block copied into, then invoke the block copy command via the sequence, [CTRL] [B] followed by [C]. Note that the block which will be copied is the first marked block found in the text buffer, A few things could go wrong with your request. If TED can find no properly marked block, it will display the error message `Marker!` and terminate the block mode. Another error which could occur is when the position you wish the block copied into happens to be in the interior of the block itself! Such a block copy can not occur. You will be informed of this by a display of the error message `Cursor!`. The successful block copy operation only copies the marked text; the markers are not copied as well. In fact, the marked text remains in its original position relative to the text which surrounds it. The cursor position relative to the text will be unchanged after the block is copied; however, the screen may be redrawn and the physical location of the cursor on the screen may be different.

A block of text may be MOVED from one position to another by a command sequence similar to the block copy. In this case, simply mark the beginning and end of the block as discussed above, move the cursor to the position in the text where you want the marked block moved to, then invoke the block MOVE command via the sequence, [CTRL] [B] followed by [M]. Again note that the block

which will be moved is the first marked block found in the text buffer. This operation is essentially one of copying and automatic deleting without the double check prompt. As in the case of the BLOCK-COPY, the same errors are possible with similar diagnostic messages when things are not as they should be. With the BLOCK-MOVE command, the new cursor position will be the new position of the moved block. The screen may be redrawn and the physical cursor position altered to accommodate this request.

This last block operation is deletion. Similar to the above functions, you first must mark the block's beginning and ending positions. You must then position the cursor to the interior of the marked block and invoke the command with the sequence, [CTRL] [B] followed by [D]. As a safety check, TED will prompt you before deleting the block. It is necessary to depress [ENTER] to affirm your intentions. Any other character entry (including a [Y]) will causeTED to ignore the block delete request.

The same errors as for copy and move can occur; however, the messages may not be for the same reasons. When a block delete is required, TED will first look for an ending block marker starting from the cursor position. If none is found, the error displayed will be `Marker!.` This doesn't mean necessarily that a properly marked block is missing. On the other hand, if an ending marker is found past the cursor position, TED next scans forward for a beginning block marker. A `Marker!` error will also be posted if none is found. If a marker is found but is also past the cursor position, a `Cursor!` error will be posted.

Filing away your text to a disk file

The [CTRL][F] command is used to save the contents of the text buffer area into a disk file. When you depress [CTRL] [F], you will be prompted for the name of the file. If the file specification you wish to use has an extension of "/TXT", you do not have to enter the extension. If the extension is omitted from your entry, "/TXT" will be automatically provided.

The FILE command will save the entire text buffer, excluding the terminating NULL but including any block markers, into the disk file identified by your input. If any disk write error is encountered while saving the text buffer into the disk file, the message `I/O error!` will be displayed. In any case, the text buffer is left undisturbed.

Text search

TED provides the SEARCH command to scan the text buffer for a specified string of characters. You specify the search by invoking the command with [CTRL] [S]. TED then prompts you for the search string with the query message `String?.`

You can enter up to 23 characters to be used for the search string. Terminate your search string with an [ENTER] (the [ENTER] character code is not included as one of the 23 characters). TED will then look for the string starting with the first character immediately following the cursor. The matching is case sensitive which means that characters entered in upper case must be found in upper case and characters entered in lower case must be found in lower case. If the search string cannot be found, the message `Can't!` will be displayed. At this point, the cursor location remains unchanged. If, on the other hand, a matching string of text is found in the text buffer, it will be displayed. The display window will be redrawn starting with the line which contains that string. The cursor will be positioned to the first character of the matching string.

If you press [ENTER] only in response to the `String?` query, then the search will proceed with the last entered search string, provided one was available. Using this procedure, you can advance the cursor to each occurrence of the search strings in question.

Another way to find each occurrence of a string is with the GO command, [CTRL] [G]. Each depression of [CTRL] [G] is identical to the sequence, [CTRL] [S] followed by [ENTER].

Text search and replace

TED also provides the capability of replacing a text string matching a search string with a different string -the replacement string. When the REPLACE command is invoked via [CTRL][R], the query message `String? will` be displayed. Although the message is the same as for SEARCH, this query is asking you for the replacement character string. You can enter up to 23 characters to be used for the replacement string. Terminate your string with an [ENTER] (the [ENTER] character code is not included as one of the 23 characters). TED will then look for the currently pending SEARCH string starting with the first character IMMEDIATELY under the cursor. If the SEARCH string cannot be found, the message `Can't!` will be displayed. At this point, the cursor location remains unchanged. If, on the other hand, a matching string of text is found in the text buffer, it will be replaced with the REPLACE string. The display window will be redrawn starting with the line which contained the string which was replaced. The cursor will be repositioned to the first character immediately following the replacement string.

If you wish to replace the next occurrence of text which matches up with the SEARCH string with that same REPLACEMENT string, all you need do is enter [CTRL] [R] and [ENTER].

The GO command, [CTRL] [G], still functions to find the next occurrence of the SEARCH string. Knowing this, if the next occurrence of the search string is beyond the text currently displayed on the screen and you wish to confirm its replacement, simply GO to the next occurrence then REPLACE, as necessary.

Printing text

TED provides no facility for printing your text file. The most reasonable way of accommodating that function is with the LIST command provided as part of your DOS.

Exiting from TED

It is easy to exit TED and return to DOS Ready. The [CLEAR] [SHIFT] [=] command tells TED you wish to exit. If the text buffer is empty, TED will immediately terminate. However, if there is any text in the buffer, you are provided an opportunity to retract your request. TED will display the prompt message,

```
Press ENTER to exit
```

Text recovery

There may be times when you exit the TED application inadvertently without saving the edited text to a disk file. TED permits you to re-enter with the asterisk parameter which provides a chance to recover you text. Instead of automatically clearing the text buffer as TED does, TED * will display whatever is in the text buffer memory area. Thus, if you have not altered any of the information in that memory area, you can always go back and recapture it.

Caution: Since all of the text pointers normally established by TED will not be initialized when invoking TED via the * parameter, it will be necessary to scroll through the text until reaching its last character prior to doing any other operation. This may also be performed using NEXT PAGE.

Changes and Additions to BASIC

The changes and additions to BASIC involve a new parameter for the renumber command, the addition of several "immediate action" keys, single letter abbreviations, high speed load. and save, and an interface to DOS SVCs.

The RENUM statement now allows for a fourth line number to be used. If used, it will be the last line of the program to be renumbered. For example:

RENUM 1000,500,10,600

This will renumber the lines between 500 and 600 only, making new line numbers starting at 1000 with an increment of 10. All other restrictions of the current renumber command still apply. For moving blocks of lines in a program, see the C and M commands describing below.

There are 6 "immediate" keys that will function when pressed as the first character of a line. The up and down arrows list the previous and next line of a program, respectively. The left arrow lists the first program line, and the right arrow lists the last program line. The period lists the current line, and the comma enters the edit mode for the current line.

Several single letter commands have been added to BASIC. Four of these are abbreviations for existing commands.

A - AUTO
D - DELETE
E - EDIT
L - LIST

These commands work exactly as their full length counterparts, except that no space is necessary between the letter and the line numbers. For example, "L100-300" is the same as "LIST 100-300". The following four commands are contained in the BASIC/OV2 file. This file must be present when using one these commands, or a `File not found` error will occur. Like other BASIC editing commands, the use of these will clear all variable values and close any open files.

C - Copy a single line
F - Find all references to a line, variable or keyword
M - Move a block of lines
S - Search and display a reference to a line, variable, or keyword.

The Copy command syntax is: C *Num1,Num2*

***Num1* is an existing line number to be copied. *Num2* is the line number to create, and must NOT already exist. No renumbering will be done after the copy. If the line numbers are incorrect, an Illegal function call error will occur.**

The Find command syntax is: **F *Object***

Object **is either a line number, variable, or keyword. The space after the F is mandatory when finding keywords. The resulting display will be all line numbers containing the referenced object. When finding variables, only the first 10 characters of the variable name will be significant. Also, type declarations !,%,#,$,(must be used. For example, the command "F A" would not find A$ or A().**

The Move command syntax is: **M *Num1,Num2,Num3***

Num1 **and *Num2* are existing line numbers and define the block of lines to be moved. *Num2* must be greater or equal to *Numl. Nurn3* is an existing line number and is the line to insert the moved block after. The moved block of lines will be renumbered by one, and all references to these lines (if any) will be corrected. If there is not enough room in memory to move the lines, an `out of memory` error will occur. If this happens, do multiple moves of smaller pieces. If the line numbers are nonexistent or if there is not enough room between Num3 and its following line to fit the block, an Illegal function call error will occur. For example, if you had program lines 100 and 110, a block of lines moved after line 100 could be no more than 9 lines long.**

The Search command syntax is: **S *Object* or S**

Object **is a line number, variable or keyword. The first fine containing the object will be displayed. The space after the S is mandatory when searching for keywords. The S with no object following will search for the next occurrence of the previous object. Like the Find command, variables are limited to 10 significant characters, and any explicit type definitions must be used.**

High speed Load and Save

When using the normal SAVE or LOAD program commands, execution should be two to three times faster then before. Programs saved with the ASCII parameter will not enjoy this speed increase, either when saving or when loading.

New user interface to SVC's

A user interface to the DOS SVC functions is now provided via the USR statement in BASIC. To use this interface, establish an INTEGER array with the first 6 elements containing the following information that may be needed by the SVC:

Element 0: **SVC number (Always needed!)**
Element 1: **Value for register pair HL**
Element 2: **Value for register pair DE**
Element 3: **Value for register pair BC**
Element 4: **Value for register pair IY**
Element 5: **Value for register pair IX**

The interface is accomplished by using a normally out of range USR argument, USR11. To execute the SVC, use the syntax USR11(VARPTR(ARRAY(0))). No DEF USR statement is required. After the SVC executes, the register pairs will be unloaded back into the array. The AF register pair will be placed in array position 0. If the array is not an integer type, or if the SVC number is either zero or greater than 127, an `Illegal function call` error will occur.

The return condition of the SVC can be tested by checking the bits in ARRAY(0) Doing an "AND 64" will produce a non-zero value if the Z flag is set. Doing an "AND 1" will produce a non-zero value if the Carry Flag is set. For further explanation of DOS SVC usage and returned values, refer to the TRSDOS 6.2 *Technical Reference Manual.* The following is a short example of using the SVC interface.

```
100 DEFINT J,K:DIM J(5)' Important- integer array only for SVC interface
200 CLS:PRINT TAB(25)"SVC Demonstration Menu"
210 PRINT:PRINT TAB(25)"1) Set Scroll Protect"
220 PRINT TAB(25)"2) Toggle Caps Lock"
230 PRINT TAB(25)"3) Show DOS Version"
240 PRINT TAB(25)"4) Check Drive Ready"
250 PRINT:PRINT TAB(25);:INPUT "Make a selection ";A$
260 IF A$ < "1" OR A$ > "4" THEN 200
270 A=VAL(A$):ON A GOSUB 1000,2000,3000,4000
280 GOTO 200
1000 CLS:PRINT"Set number of scroll protect lines 07"
1010 PRINT:PRINT"0 will cancel scroll protect ";:INPUT A
1020 IF A < 0 OR A > 7 THEN 1000
1030 J(3)=&H700 + A 'Register B=7, RegisterC=line count
1040 J(0)=15:X=USR11(VARPTR(J(0))) 'Execute @VDCTL SVC
1050 IF A=0 THEN RETURN
1060 FOR K=1 TO 100:PRINT K;"Testing scroll protect to 100":NEXT:RETURN
2000 CLS:J(0)=101:X=USR11(VARPTR(J(0))) ' @FLAGS SVC to get keyboard flag
2010 K=PEEK(J(4)+10)' Get KFLAG value
2020 POKE (J(4)+10), K XOR 32 'Toggle Caps lock bit
2030 INPUT"Type some characters to check Caps lock, press ENTER to end ";A$
2040 RETURN
3000 CLS:J(0)=101:X=USR11(VARPTR(J(0)))' Get flag table base, @FLAG SVC
3010 K=PEEK(J(4)+27)' Peek out version number, base+27
3020 K=K-(&H60)' Earliest version was 6.0, K will be 0 to 3
3030 PRINT "This is DOS version 6 .";K
3040 PRINT:INPUT"Press ENTER to return"; A$:RETURN
4000 CLS:INPUT"Enter drive # to check 0 to 7 ",K
4010 IF K < 0 OR K > 7 THEN RETURN
4020 J(3)=K' Set drive number for SVC
4030 J(0)=33:X=USR11(VARPTR(J(O)))' @CKDRV SVC
4040 PRINT:PRINT"Drive";K;"has ";
4050 IF (J(0) AND 64)=0 THEN PRINT"no disk mounted.":GOTO 4080
4060 PRINT"a disk mounted."
4070 IF (J(0) AND 1)=1 THEN PRINT"The drive is write protected."
4080 PRINT:INPUT"Press ENTER to return ";A$:RETURN
```

This example program shows several methods of accessing information not normally available from BASIC. To make effective use of the SVC interface, you will need to have the *Model 4 Technical Reference Manual.*

The @CKDRV routine (lines 4000-4080) shows how the return status of an SVC can be checked. The Z flag can be tested by doing an "AND 64" against the array(0) position. If the result is zero, the Z flag was NOT set. The CF (carry flag) can be tested in the same manner by doing an "AND 1". Again, a zero result means the flag was not set.

The Caps lock and DOS Version subroutines show how information can be referenced in the system flag area with the @FLAGS SVC. There are several other flags and system storage areas than can also be referenced off of the flag table base.

Certain SVCs, such as @CMNDI, @CMNDR, or setting the high memory pointer must NOT be done. Generally speaking, anything that can be legally be done from BASIC with @CMNDR can be done with the SYSTEM *"command"* statement, and should not be attempted through the SVC interface.

New BASIC Cross Reference Utility

BREF/CMD is a cross reference utility for BASIC programs. It is executed at the `DOS Ready` prompt, not from inside of BASIC. The BASIC program must NOT have been saved in ASCII. The syntax is:

BREF filespec [(parm,parm,...)]

Parameters are:

VAR=*ON*|*OFF*	**- Default is on**
LINE=*ON*|*OFF*	**- Default is off**
P=*ON*|*OFF*	**- Default is off**
W=*n*	**- Width for output to printer, Default is 80**

The *filespec is* the name of the BASIC program. The VAR parameter allows variables to be cross referenced. The LINE parameter includes a cross reference of line numbers. The P parameter allows the listing to go to a printer rather than the video. The W parameter can be used to specify the number of columns for the printer (generally either 80 or 132, although any value between 32 to 255 will be accepted).

Variable names will be displayed up to 14 characters. If the variable is longer than this, the remaining characters will be truncated for display. BASIC stores certain keywords in ASCII, and this makes them indistinguishable from a variable. For example the "AS" used in field statements. This will cause these words to be displayed in a variable cross reference list.

If there are more references than can be displayed on a single line, they will wrap to the next line. This line will have an asterisk in column one to denote the overflow.

Examples

BREF PROG/BAS - Sends only variable references to the video.

BREF PROG/BAS (LINE,VAR=OFF) - Sends only line number references to the video.

BREF PROG/BAS (LINE,P,W=132) - Sends variables and line numbers to a 132 column printer.

Possible errors

There are three error messages that may be generated by BREF:

```
Not a BASIC program
```

The general cause of this error is trying to use a program saved in ASCII, a program saved in the protected mode, or a non-BASIC program file.

```
Out of memory - can't cross reference
```

This will occur if the program is too large to fit into your available memory. The cure is to change your configuration to free up some memory.

```
Line nnnn, Error in original program
```

This message may occur if there is a syntax error in the original BASIC program. The line number should correspond to that line in the program.

TECHNICAL INFORMATION CHANGES

Changes to directory structure

GAT - The disk type byte (GAT+CDH) now assigns bit 3 as the date type bit. If this bit is set, the disk is assumed to use the new time/date fields.

DIR+2 - This byte continues to hold the day in bits 3-7 as documented for earlier versions. However, bits 0-2 should no longer be considered the primary year field. For compatibility with earlier versions, these bits should remain untouched by the user as certain system routines will modify this area.

DIR+18,19 - These two bytes are no longer assigned as the user password. Instead, they contain the new style time/year information as follows:

DIR+18 - bits 3-7 contain the hour.
bits 0-2 contain the most significant bits of the minute.
DIR+19 - bits 5-7 contain the least significant bits of the minute.
bits 4-0 contain the year offset from 1980.

Changes to SVCs

@CKDRV - This SVC now adjusts the appropriate bit in the Y flag to reflect GAT+CDH, bit 3 (the old or new style date bit) for the drive being accessed.

@FLAGS - The 'Y' flag byte has been assigned to mark the dating style of a disk. Bits 0 to 7 correspond to drives 0 to 7. If the bit is set, the drive is assumed to use the new style time/date information. The setting and resetting of this bit is handled by the @CKDRV SVC. The operating system version (flag table base+27) has been set to 63H. The release number (flag table base-47) has been set to 00H.

@HEXD - NEW SVC, number 95. (Similar to @HEXDEC SVC, number 97)

Converts a binary number in HL to decimal ASCII.
Entry conditions:
A = 95 (X'5F')
B = expected number of output digits
HL = number to convert
DE = pointer to buffer to hold converted number

Exit conditions: Success always.
DE = pointer to end of buffer + 1
AF, BC, HL are affected by this SVC

This SVC is the same as @HEXDEC except that a 5 character buffer is not required. Instead, the B register should contain the maximum number of output characters expected, and DE should point to a buffer of at least that size. If the conversion exceeds this number of characters, the overflow will be placed in memory below the buffer pointed to by DE.

@INIT - If a file is created with a password, the protection level will now be set to NO. This means that no future access will be given without the use of the password.

@VDPRT - NEW SVC, number 107. This SVC performs a screen print. If the printer is not ready, the SVC will abort in approximately 20 seconds.

Entry conditions: A=107 (X'6B')
Exit conditions: Success always.
BC, DE, HL are changed. No status is returned.

SOFTWARE LICENSE AGREEMENT

The LS-DOS 6.3 software update provided by Logical Systems, Incorporated (LSI) is intended only for legitimate owners of a TRSDOS 6.x operating system. By accepting this software license, the buyer expressly warrants and represents that he is a legitimate owner of a TRSDOS 6.x operating system.

The LS-DOS 6.3 software update and documentation is copyrighted with all rights reserved by Logical Systems Incorporated. The buyer is granted a license to use this software on one computer system, and may make copies for personal use and archival purposes only. Copying this software for sale or other distribution is expressly prohibited.

LIMITED WARRANTY

LSI warrants the diskette media on which the LS-DOS 6.3 is recorded to be free from defects under normal use for a period of 30 days after the date of original purchase. If during this 30 day period a defect in the diskette media should occur, you may return the diskette to LSI for replacement without charge.

Except as provided for above, LSI makes no warranties, either expressed or implied, with respect to this software or its fitness for any purpose. LSI software is provided "as-is". Should the software prove defective, the buyer assumes the entire cost of all necessary servicing, repair, or correction and any incidental or consequential damages. LSI assumes no liability for direct, indirect, incidental or consequential damages resulting from any use of or any defects in the software.

CUSTOMER SERVICE

Each LS-DOS 6.3 contains its own individual serial number and customer service I.D. number. Your customer service I.D. number is very important and must be included if you write to LSI for service or your letter will be ignored. You must also have this number available when calling our customer

service department. In either case you must first have filled out and returned your registration card. DO NOT DELAY ..... under no circumstance will customer service be provided to unregistered owners.

For customer service contact:

LSI
CUSTOMER SERVICE DEPT.
P.O. Box 55235
Grand Junction, CO 81505

NOTE: Customer service phone hours are 9 a.m. to 12 noon, Mountain time.

Phone: (303) 242-2929
9 a.m. to 12 noon

WARNING

Tampering with the serial number or customer service number or the other security and control systems in the LS-DOS 6.3 product will cause the system to fail possibly causing damage to or loss of data. Remember each system is licensed for use on ONE machine, LSI has taken certain steps to see to it this license is not violated and assumes no responsibility whatsoever regarding damage that may result from violation of this license.