FOI TUDO

MULTIDOS

ERRATA SHEET

PAGE 9
Add after (To suppress an automatic "AUTO" function, hold down the "ENTER" key during power-up.)
To input the date and keep the "AUTO" function during power-up, terminate the date entry with <BREAK>.

PAGE 18
Add the following at the end of KEYBRD:
To defeat KEYBRD "ON" attributes during power up, hold down any combination of the following:

Lower case	left-arrow
Graphics driver	"CLEAR" key
Repeating keyboard	right-arrow
Blinking cursor	down arrow
Blinking character	up-arrow
MULTIDOS keyboard	<SHIFT>

PAGE 32 & 33
Under SPOOL/CMD, items 2, 3, 4, 5, and 6 have been deleted. (Use "FORMS" for pagenation). Change the paragraph below item 7 to read:
The SPOOLER will now commence operation and control of any printer output.

To suspend output press shift <BREAK>. You will be asked if the buffer is to be saved. A "Y" response will save the buffer contents, and an "N" response will reset the pointers and send a carriage return to the printer.
The next query to appear will be:

'SPOOL? '

Enter "Y" if you want to continue SPOOLING. An "N" response will unlink the SPOOLER from the printer DCB and restore TOPMEM if possible.

Delete the next two paragraphs. (They belong under FORMS.)

PAGE 37
Second paragraph - delete "and provide." at the end.

PAGE 46
Last line should end with 'GOTO 15'.

PAGE 65
Acceptable syntax for PUT should be:

```
PUT[#]buf[,rec]
```

PAGE 66
Correct syntax for GET should be:

```
GET[#]buf[,rec]
```

IMPORTANT NOTICE

Cosmopolitan Electronics Corporation distributes all software on an "AS IS" basis without warranty. Cosmopolitan Electronics Corporation shall not be liable or responsible to the purchaser with respect to liability, loss, or damage caused or alleged to be caused directly or indirectly by the use of this software, which includes but is not limited to any interruption of service, loss of business, or anticipatory profits, or consequential damage resulting from use of this software.

Throughout this manual there will be references made to trademarked products. The (tm) symbol will be used once here to serve throughout the manual.

DBLDOS (tm) is a registered trademark of Percom Data Co.
LDOS (tm) is a registered trademark of Logical Systems Inc.
TRSDOS (tm) is a registered trademark of Radio Shack.
VTOS (tm) is a registered trademark.

Cosmopolitan Electronics Corporation
P.O. Box 234
Flynouth, Mi. 48170

What is MULTIDOS

This operating system is the underlying software required to interface user programs with the hardware. It was written with the intent of user oriented software without being superfluous.

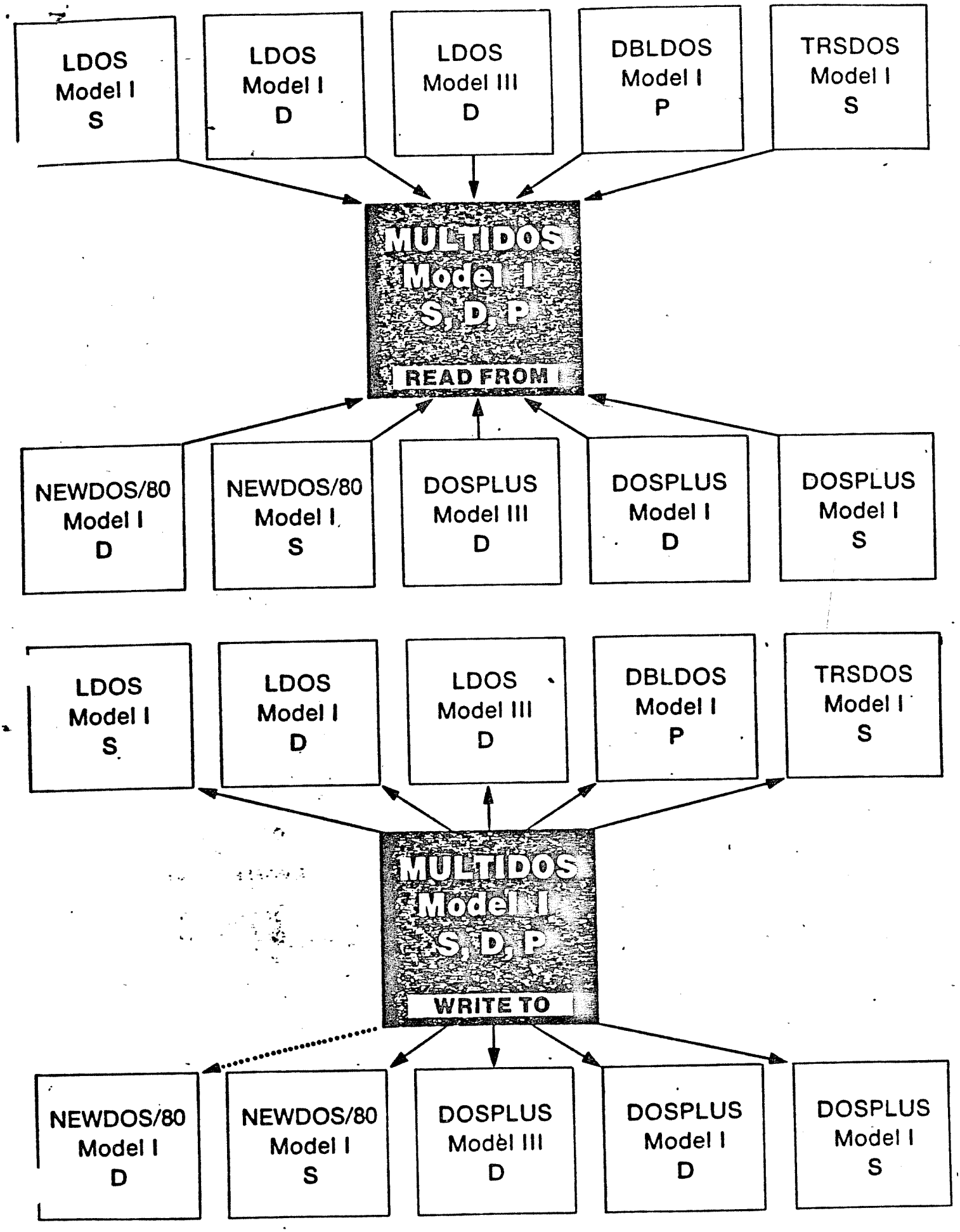
The word format was used here to describe that particular systems' logic in communicating with its diskette.

Since we are no experts on other operating systems, we can only tell you what MULTIDOS is capable of doing. To best describe MULTIDOS' interface capabilities the diagram shown on the opposite page indicates what can be read/written to.

				MULTIDOS MODEL I		MULTIDOS MODEL III	
MOD.	SYSTEM	DENSITY	FORMAT	READ	WRITE	READ	WRITE
I	TRSDOS	single	A	OK	NOTE 1	CONVERT	CONVERT
I	NEWDOS	single	A	OK	NOTE 1	CONVERT	CONVERT
I	VTOS	single	A	OK	NOTE 1	CONVERT	CONVERT
I	ULTRADOS	single	A	OK	NOTE 1	CONVERT	CONVERT
I	NEWDOS/80-1	single	A	OK	NOTE 1	CONVERT	CONVERT
I	DOSPLUS	single	A	OK	OK	CONVERT	CONVERT
I	NEWDOS/80-2	single	A	OK	OK	CONVERT	CONVERT
I	LDOS	single	B	OK	OK	OK	OK
I	MULTIDOS"S"	single	B	OK	OK	OK	OK
I	DBLDOS	double	C	OK	OK	OK	OK
I	VTOS	double	C	OK	OK	OK	OK
I	NEWDOS/80-1	double	C	OK	OK	OK	OK
I	MULTIDOS"P"	double	C	OK	OK	OK	OK
I	DOSPLUS	double	D	OK	OK	OK	OK
I	LDOS	double	G	OK	OK	OK	OK
I	MULTIDOS"D"	double	D	OK	OK	OK	OK
I	NEWDOS/80-2	double	E	OK	NOTE 2	OK	NOTE 2
III	TRSDOS	double	F	NO	NO	TRANSFER	NO
III	DOSPLUS	double	G	OK	OK	OK	OK
III	LDOS	double	G	OK	OK	OK	OK
III	MULTIDOS	double	G	OK	OK	OK	OK
III	NEWDOS/80-2	double	H	NO	NO	OK	NOTE 2

NOTE 1: These systems will have their address marks changed.

NOTE 2: EXTREME CAUTION: MULTIDOS does not know the track count. Be sure sufficient space is avaiable via NEWDOS/80, MULTIDOS does not report correct free space (except CAT).


LDOS Model I S
LDOS Model I D
LDOS Model III D
DBLDOS Model I P
TRSDOS Model I S
MULTIDOS Model I S, D, P
READ FROM
NEWDOS/80 Model I D
NEWDOS/80 Model I S
DOSPLUS Model III D
DOSPLUS Model I D
DOSPLUS Model I S
LDOS Model I S
LDOS Model I D
LDOS Model III D
DBLDOS Model I P
TRSDOS Model I S
MULTIDOS Model I S, D, P
WRITE TO
NEWDOS/80 Model I D
NEWDOS/80 Model I S
DOSPLUS Model III D
DOSPLUS Model I D
DOSPLUS Model I S
S = Single Density
D = Double Density
P - P Density (Modified Double Density)

CONTENTS

SUPERBASIC OVERLAYS

SUPERBASIC FILE MANIPULATION

SUPERBASIC FILE ACCESS

TECHNICAL

Cosmopolitan Electronics Corporation
P.O. Box 234
Plymouth, Mi. 48170

Foreword

I would like to thank several people who have encouraged me to develope MULTIDOS. Gordon Monnier and Dave Welsh for assistance in debugging this fine operating system. Kim Watt gave unlimited technical assistance in high speed and double density operation.

In addition, I would like to acknowledge an excellent word processor, "LAZY WRITER", for a superb job of formatting the printing of this documentation.

Vernon B. Hester

CONTENTS

SUPERBASIC OVERLAY UTILITIES

SUPERBASIC FILE MANIPULATION

SUPERBASIC FILE ACCESS

MULTIDOS Uniqueness

Here is a list of items you have available with MULTIDOS that have yet to appear in other operating systems.

FUNCTION	MULTIDOS	TRSDOS	NEWDOS 80	DOS PLUS	LDOS
Repeat Last DOS Command	YES	NO	use "R"	NO	NO
Multiple DOS Command	YES	NO	NO	NO	TYPE AHEAD
APPEND checks LRL	YES	NO	NO	NO	YES
ATTRIB to Lockout	YES	NO	YES	NO	NO
Blind AUTO	YES	NO	NO	NO	NO
Invinceable AUTO	YES	NO	SYS opt.	NO	NO
BOOT @ HI SPEED	YES	NO	NO	NO	NO
Nested DO'S	YES	NO	Replaces	NO	NO
Software powerup	YES	NO	NO	NO	NO
HI SPEED DEBUG	YES	NO	NO	NO	NO
Executable DEBUG	YES	NO	NO	NO	NO
Alphabetized Directory	YES	NO	NO	NO	NO
DUMP to 3000H	YES	NO	YES	NO	NO
FORMS	YES	NO	NO	YES	NO
Totals FREE	YES	NO	NO	NO	NO
HASH Codes	YES	NO	SUPERZAP	NO	NO
Keyboard attributes	YES	NO	SYS opt.	NO	SYS opt.
LOAD to 5200H	YES	NO	YES	NO	NO
TOPMEM	YES	NO	YES	NO	YES
Auto-multiple density	YES	NO	PDRIVE	SIN/DBL	SIN/DBL
BACKUP GRANULES FROM BAD SOURCE	YES	NO	NO	NO	NO
COPY 1 drive	YES	NO	YES	YES(limited)	YES
COPY only if sufficient space	YES	NO	NO	NO	NO
LOCK out GRANULES vs TRACKS for FORMAT	YES	NO	NO	NO	NO
Graphics driver	YES	NO	NO	NO	NO
Full function single step BASIC	YES	NO	NO	NO	NO
Zero arrays	YES	NO	NO	NO	NO
Delete arrays	YES	NO	YES	NO	NO
Sort	YES	NO	YES	NO	NO
Used variables displayed	YES	NO	NO	YES	NO
Transfer to Level II	YES	NO	NO	NO	NO
Folded CMD"uuuuu"	YES	NO	NO	NO	YES
Find	YES	NO	YES	NO	NO
Intelligent GLOBAL EDITOR	YES	NO	YES	NO	NO
String packer	YES	NO	NO	NO	NO
Line splitter	YES	NO	NO	NO	NO
Line merger	YES	NO	NO	NO	NO
Renumber to line 0	YES	YES	NO	YES	YES
Renumber Packed Strings	YES	NO	NO	NO	YES
Lists GRAPHICS	YES	NO	NO	NO	NO
Automatic HI SPEED	YES	NO	NO	NO	NO
Full time Blinking cursor	YES	NO	NO	NO	NO
Won't hang if no printer	YES	NO	NO	NO	NO
Full time Repeating Keys	YES	NO	NO	YES	NO
Menu driven Copy by file	YES	NO	NO	NO	NO

FIRST THINGS FIRST

The first thing you must do with your MULTIDOS diskette, is make a backup. A blank diskette without the write protect notch covered is required for this procedure.

LEAVE THE WRITE PROTECT TAB ON THE MULTIDOS DISKETTE.

1. Turn on your expansion interface.
2. Turn on your peripherals (disk drives, printer, etc.)
3. Turn on your CPU/keyboard.
4. Insert your MULTIDOS diskette into drive 0.
   (open door, insert, close door)
5. Press the reset button.

After you press the reset button, the display will indicate that this is a MULTIDOS diskette and will prompt you to input today's date. MULTIDOS will date your files with the date you have just entered. If you don't care about dating your files press "BREAK" or "ENTER".

Type BACKUP, then press the "ENTER" key.

Answer 'Which drive contains the source diskette?'
with 0

Answer 'Which drive for the destination diskette?'
with 1 if you have 2 drives otherwise 0.

The next prompt will be:
'Press "ENTER" when the source diskette is in drive 0'.
Press the "ENTER" key.

At this point MULTIDOS will examine the source diskette for track count and density. The track count will be 35. The density will be the one you ordered.

Answer 'Track count for the destination diskette (35 to 96)?'
with the maximum track capacity of the destination drive (usually 35, 40, or 80).

If you only have a 35 track drive you cannot make a 40 track backup!

MULTIDOS will format the blank diskette, then proceed to duplicate the contents of your MULTIDOS diskette.

If you are only using one drive, BACKUP will tell you when to insert the blank (destination) diskette, and when to re-insert your MULTIDOS (source) diskette. During a one drive BACKUP, you will have to exchange the diskettes several times.

When BACKUP has completely duplicated your MULTIDOS diskette,
'Completed
Press "ENTER" when a MULTIDOS system diskette is in drive 0'
will be displayed. Press the "ENTER" key.

It is highly recommended that your original MULTIDOS diskette be stored in a safe place and all future backups made from the copy you just produced.

What Is MULTIDOS

One of the purposes of MULTIDOS, is a link to all the Model I DOS's on the market.

MULTIDOS will permit you to read/write/copy to/from any Model I single/double density operating system configured as purchased. (NEWDOS80/2.0 double density data diskettes can only be read). MULTIDOS is capable of reading/writing/copying to/from several Model III operating systems diskettes. (NEWDOS80/2.0 and TRSDOS are excluded).

In addition, MULTIDOS has an excellent extended disk basic, SUPERBASIC, which has a super fast string sort and leaves the user with more than 40K (48K RAM) of free memory. What was sacrificed in obtaining the 40K - NOTHING! In fact, SUPERBASIC has several innovative and useful functions which the programmer/user will find invaluable. One of which is an intelligent global editor. This is the only one available which will allow the user to change a "T" variable without changing the "T" in a key word as "PRINT", and the letter "T" embedded within quotes. However, it will allow the user to change "PRINT" or any character within quotes independently if desired. Another innovation is the abil.ty to chain basic programs maintaining all scalar and array variables.

SUPERBASIC maintains all of the syntax developed for the Model I, with one exception - octal/hex constants. SUPERBASIC uses the "H" vs the "O" as the optional character whereas TRSDOS basic uses the "O" vs the "H" as the optional character.

EXAMPLE:

TRSDOS	SUPERBASIC
&H4000 = 16384d	&H4000 or &4000 = 16384d
&4000 = 2048d	
&O76 or &76 = 62d	&O76 = 62d, &76 = 118d

Also there are major difference in the "CMD" functions. SUPERBASIC 3.0, which was developed on July 7, 1979, started using several "unused" letters as special functions. Subsequently NEWDOS80/1, TRSDOS Model III, VTOS, LDOS, and DOSPLUS have added their own.

Here are SUPERBASIC's "CMD" functions.

INTRODUCED	CMD	ADDITIONAL Input	FUNCTION
7/06/79	C	No	Deletes spaces & linefeeds from program.
ORIGINAL	D	No	Loads DEBUG
ORIGINAL	E	No	Displays last DOS error.
2/01/81	K	Yes	Zeroes target arrays.
2/01/81	L	Yes	Deletes target arrays.
2/01/81	M	Yes	Moves program line.
2/01/81	N	Yes	Duplicates program line.
7/06/79	O	No	Opens an additional file buffer.
3/14/81	Q	Yes	Sorts string arrays.
ORIGINAL	R	No	Turns on interrupts.
ORIGINAL	S	No	Returns to DOS.
ORIGINAL	T	No	Turns off interrupts.
9/01/81	V	No	Lists currently assigned variables.
7/06/79	X	No	Transfers basic program to Level II.

NOTATION

Symbol	Meaning
<ENTER>	Press the "ENTER" key.
<BREAK>	Press the "BREAK" key.
<SPACE>	Press the "SPACE-BAR".
<COMMA>	Press the "," key.
filespec	A valid MULTIDOS file.
drivespec	A particular disk drive number (0,1,2, or 3)
[]	Brackets enclose optional parameters. The "[" and "]" are not typed in.
a space	At least one mandatory space.
punctuation	Must be entered as shown.
single quote '	Encloses MULTIDOS responses which are directed to the display.
...	The triple periods indicate you may optionally repeat the last item in brackets.

COMMANDS

MULTIDOS is similar to other DOS's. Whenever the prompt,

Multidos ready

is displayed, you may enter a operator command. In the simple form of as little as one word followed by <ENTER>.

EXAMPLE:

FREE<ENTER>

The above command instructs MULTIDOS to display the amount of free granules remaining on all drives.

All of the MULTIDOS commands terminate with <ENTER>.

The maximum length of a command is 63 characters followed by <ENTER>.

When specifying a diskette file the term "filespec" will be used in this reference manual. The "filespec" will be as follows:

FILENAME[/EXT][.PASSWORD][:D]

FILENAME is the name of the diskette file. Valid filenames consist of at least one and not more than eight characters. The first character must be alphabetic (A-Z), and the remaining characters can be alphabetic or numeric (0-9).

/EXT is an optional extension to the FILENAME and consists of the slash character (/) and at least one, but not more than three characters. The first of these characters must be alphabetic The remaining characters can be alphabetic or numeric. All file extensions, except "/CMD", must be entered for proper identification of the file. More on this later.

.PASSWORD is an optional password consisting of the period symbol and one to eight characters. The first character must be alphabetic and the remaining can be alphabetic or numeric.

:D is an optional drivespec consisting of a colon followed by a number in the range of 0 to 3.

Please note that a filespec does NOT have any spaces on it. Spaces are treated as separators by MULTIDOS. If spaces are in a filespec, only the contents preceding the space will be interpreted as the filespec.

The PASSWORD is NOT part of a filespec's uniqueness. The following are interpreted as the same filespec:

WONDER.BOY WONDER.WOMAN WONDER.FULL

However, the following are unique filespecs:

WONDER:1 WONDER:2 WONDER:3

For extensions, and general conformances to our micro's users, the following extentions are typical.

/ASC	A basic program stored in ASCII form.
/ASM	An assembly language source file.
/BAS	A basic program stored in compressed format.
/CIM	A file created via "DUMP".
/CMD	A machine language executable file.
/REL	A relocatable object file.
/SYS	An operating system's overlay.
/TXT	An ASCII text file. (Typically sequential files)

MULTIDOS will insert the default extension /CMD, if you enter a filespec without an extension.

EXAMPLE:

VFU<ENTER>

MULTIDOS will add the extension /CMD to VFU, then load and execute the file VFU/CMD.

If you do not assign a drivespec, MULTIDOS will search all drives starting with 0 for the filespec entered. If you are initializing a filespec without a drivespec, MULTIDOS will search for the first drive without a write protect tab.

More on COMMANDS

Multiple DOS commands are allowed when separated by a comma and no spaces.

EXAMPLE:

FREE,CLOCK,BASIC<ENTER>

1. Will display the free granules and free file spaces on all mounted diskettes.
2. Displays the real time clock.
3. Loads and executes SUPERBASIC

MULTIDOS will repeat the last DOS command if <ENTER> is the only response to the "Multidos ready" prompt.

EXAMPLE:

DIR :1<ENTER>

Followed by <ENTER> will again display the directory of the diskette in DRIVE 1.

NOTE: The diskettes may be changed between the <ENTER>'s.

To "ERASE" the last DOS command press the "BREAK" key.

SPECIAL UNIVERSAL COMMANDS

I - MIGHTY MULTI.

This special overlay will permit you to copy files, display a directory, kill files, and list files from within any program in which the interrupt service is active.

This command will be activated if the following conditions are met.

1. Interrupt service is active.
2. Drive motors have stopped.
3. Simultaneous depression of the ":" and ";" keys.

This command will exit on one of the following:

1. The "BREAK" key is pressed.
2. The "I" character is the first input character.

However, upon exit an extraneous key input may be present.

The syntax for the various functions is accomplished with a single letter and no connectors. (eg. TO)

Ia - COPY - Copies files, "C"

C fileone:d filetwo:d

Please note the connector 'TO' is not part of this command syntax, and the first blank is optional. Although the drive numbers are optional, the filespecs are mandatory. (C fileone:1 :2 is invalid)

Ib – DIRECTORY – Displays a diskette directory, "D"

D:n or Dn

n = 0, 1, 2, or 3, which represents the drive number.

Ic – KILL – Deletes file, "K".

K filespec:d

Id – LIST – List a file to the Display, "L".

L filespec:d

The listing can be temporarily suspended if the SPACE-BAR is held down.

Il – JKL Dump the video contents to the printer.

JKL = The simultaneous depression of the J, K, and L keys.

This function sends the current contents of the display to the device defined as the printer. The function does not print blanks occuring after the last printable character of a line. The system will not "HANG" if the printer is not ready.

Graphic blocks are converted to the "." character. If you do not want this to happen, use the "HJK" command.

To prematurely terminate this function, <BREAK>.

Ill – HJK Dump display contents to printer, including graphics.

HJK = The simultaneous depression of the H, J, & K keys.

This command is similar to the JKL command, except that graphics characters will be sent to the printer instead of being converted to "." characters

The following commands are **MULTIDOS** library commands and are normally entered after the "Multidos ready" prompt has been received.

A mandatory space is required after all library commands in which there are additional parameters or arguments.

EXAMPLES:

(1) DATE 09/01/81<ENTER>

This establishes September 1, 1981 as the current RAM date.

(2) DATE<ENTER>

This displays the current RAM date in the format mm/dd/yy.

Whenever the term "switch" is part of a **MULTIDOS** library command line, it represents an ON or OFF condition. If the "switch" is left out, ON will be invoked.

The acknowledged responses are:

(Y)	= yes, on	(N)	= no, off
(YES)	= yes, on	(NO)	= no, off
(ON)	= yes, on	(OFF)	= no, off

Please note the required "(" and ")" surrounding the switch parameter.

EXAMPLE:

(1) BREAK (N)<ENTER>
(2) BREAK (Y)<ENTER> or BREAK<ENTER>

Example one will disable the "BREAK" key, and example two will enable the "BREAK" key.

APPEND Appends a file to the end of another file.

APPEND filespec1 to filespec2<ENTER>

APPEND allows you to add filespec1 to the end of filespec2. filespec1 is not affected by this command.

NOTE: The disk which contains filespec2, must not be write protected and must have at least enough free space to lengthen filespec2 by the length of filespec1, and both filespecs must have the same logical record length.

Be very careful with basic programs. Both should be ASCII files and the line numbers in filespec1 should be higher than those of filespec2.

EXAMPLE:

APPEND CLASS2/TXT to STUDENTS/TXT<ENTER>

The file STUDENTS/TXT will have the contents of file CLASS2/TXT added to the end. The file CLASS2/TXT will remain unchanged.

ATTRIB Assigns filespec attributes.

ATTRIB filespec (param[,param...])<ENTER>

This command sets file protection attributes.

param	Can be any of the following:
I	The file is invisible to the normal directory command.
V	The file is visible to the normal directory command.
A=pw	pw = The new access password.
U=pw	pw = The new update password.
P=level	level = The protection level.

LEVEL	VALUE	ACCESS PASSWORD PRIVILEGE
KILL	1	TOTAL ACCESS
RENAME	2	RENAME, WRITE, READ, EXECUTE
WRITE	4	WRITE, READ, EXECUTE
READ	5	READ, EXECUTE
EXEC	6	EXECUTE ONLY
NONE	7	LOCKED OUT

If a file has a protection level of WRITE (4), it can be accessed by any of the levels equal to or higher than WRITE, i.e. READ, EXECUTE, with the use of the access password. The file cannot be RENAMEd or KILLed without the update password.

The access attribution does not require an update nor protection level attribute. Files with access attribution require the access password for any operation with the file.

All files have an access and update password. The default password is 8 blanks. The "DIR" command will display the protection level for all files which have a non-blank update password, with its protection level indicated by "P=X" to the right of the filename, whereas X is the protection level value.

EXAMPLES:

(1) ATTRIB FILEA/BAS (A=,U=SAM,P=READ)<ENTER>

FILEA/BAS can be executed, loaded, and read without passwords. To write to the file, RENAME, or KILL the file, the password "SAM" must be used.

(2) ATTRIB FILEB/BAS (V)<ENTER>

FILEB/BAS is visible and will be displayed when the "DIR" command is issued.

AUTO Automatic command after reset.

AUTO[DOSCMD][linefeed DOSCMD]<ENTER>

AUTO provides for automatic operation of one or more MULTIDOS commands or executable command files upon power-up. The commands or executable command files are executed in sequence immediately after power-up.

DOSCMD is any valid MULTIDOS library command or a filespec for an executable command file.

To provide for multiple auto commands, insert a linefeed (down arrow) between each MULTIDOS command or filespec you want executed upon power-up.

A maximum of 32 characters after the "AUTO " including blanks, linefeeds, and the terminating <ENTER> are allowed. If your entry exceeds 32 characters, the 32nd character will be replaced with an <ENTER> and any additional characters will be disregarded.

Since the "AUTO" command writes to the directory, the diskette must not be write protected when the "AUTO" command is set up or removed.

To delete an automatic power-up sequence, type:

AUTO<ENTER>

To suppress an automatic "AUTO" function, hold down the "ENTER" key during power-up.

EXAMPLES:

(1) AUTO DIR<ENTER>

On future power-ups the directory will automatically be displayed after 'DIR' appears on the display.

(2) AUTO
CLOCK<LINEFEED>
BASIC RUN"TEST"<ENTER>

On future power-ups the real time clock will be displayed, BASIC will be loaded, the program "TEST" will load and run. You will see 'CLOCK' displayed, then 'BASIC'RUN"TEST" '

To inhibit the suppression of the "AUTO" function, key in an "!" (exclamation mark - shifted "1" on your keyboard) as the first character after the mandatory space following "AUTO".

EXAMPLE:

AUTO IBASIC<ENTER>

On subsequent power ups, BASIC will be loaded after 'BASIC' is displayed, whether the <ENTER> key is held down or not.

To inhibit displaying the AUTO DOSCMD(s), place a "#" (pound sign, shifted "3" on your keyboard) in front of the first DOSCMD.

EXAMPLE:

AUTO #BASIC<ENTER>

On subsequent power ups BASIC will be loaded but not displayed on the screen.

AUTO I#BASIC

This is the proper combination of the extended "I" and "#" AUTO functions.

BOOT Resets the computer.

This command will cause a hardware and software reset.

BREAK Disables/enables the break key.

BREAK switch<ENTER>

BUILD Creates a "DO" file.

BUILD filespec<ENTER>

This command will create or add to a file for "DO" execution. The maximum input line is 63 characters. This creates a file to store keystrokes for a specified series of commands and/or program responses.

To terminate a current "BUILD" and close the file, press the "BREAK" key.

MULTIDOS has 3 special "BUILD" characters which are recognized only if it is the first character in a "BUILD" line. (#, $, %)

The "#" character is used to pause execution of a "DO" file. If a message is desired, type it in immediately after the "#" character.

EXAMPLES:

#<ENTER> Pauses "DO" and waits until a "SHIFT" key is pressed.

#INSERT DISK #77<ENTER> = Pauses "DO" after displaying message 'INSERT DISK #77'.

The "$" character is used to supress video output during "DO" without a 'pause'. If text follows the "$" character it will pause the "DO" file after being displayed.

The video display is turned on whenever the current "DO" file is completed regardless if it is a nested "DO" within a "DO" which turned off the video display.

The "%" character is used to display text only. It is used to display multi-line messages, which are entered one line at a time. If no text follows the "%", DOS will try to execute "%". Since % is not a valid DOS command the "UNKNOWN" message will be displayed and the "DO" file continued.

NOTE: "BUILD" and "DO" automatically insert the extension "!DO" if none is specified.

CLEAR Zeroes RAM.

CLEAR<ENTER>

CLEAR zeroes RAM from 5200 to TOPMEM. The limit of free memory, called "TOPMEM", is the address pointed to by the contents of 4049H & 404AH.

EXAMPLE:

(1) CLEAR<ENTER>

If the contents of 4049H & 404AH are FFH and FDH respectively, the above command will cause all bytes from 5200H through FDFFH to be set to 00.

CLOCK Displays the real time clock.

CLOCK switch<ENTER>

The ON mode forces the real-time clock to be displayed at the top right of the video display.

As long as interrupts are enabled, the clock will be updated for each second, minute, and hour. After 23:59:59 the clock will be reset to 00:00:00 and the date will not be incremented. Also note that certain functions such as disk I/O and certain programs such as SCRIPSIT cause the interrupts to be turned off. This affects the accuracy of the clock.

EXAMPLES:

(1) CLOCK (OFF)<ENTER>

The display of the clock will be discontinued.

(2) CLOCK (ON)<ENTER>

The clock will be displayed.

CONFIG Default power-up drive attributes.

CONFIG[(param[,param])]<ENTER>

param	meaning
drn	drivespec
STEP=nn	nn is drn track to track access speed
DENSITY=q	q is the default density for drn upon power-up.

Sets individual drive stepping speeds and initial density.

EXAMPLE:
CONFIG :2(STEP=6,DENSITY=P)<ENTER>

Will write this info onto drive zero diskette (system disk), and set RAM configuration, for drive 2, on subsequent reboots/power-ups.

Syntax is forgiving, only the first letter is considered.
i.e. CONFIG 2(S=6,D=P) is equivalent to the above example.

The stepping speeds are the track to track access speeds. The optional speeds available are 6, 12, 20, and 40mS. NOTE: The 6mS speed is only attainable with a double density hardware modification. The density's available are: S = single density, P = DBLDOS (tm) density, and D = double density.

CONFIG<ENTER> echo's system disk's configuration.

CONFIG (?)<ENTER> echo's RAM's current configuration.

CONFIG (X)<ENTER> writes RAM's configuration onto system diskette.

DATE Sets date.

DATE mm/dd/yy<ENTER>
DATE<ENTER>

This command allows you to set the date. If you use the form DATE<ENTER>, MULTIDOS will display the date currently stored in RAM. The date is set to 00/00/00 at power-on. However, at a non-powerup reboot, the system will retain the date previously set in RAM.

DEAD Software power on.

All memory from 4000H upward will be set to 00,
and the system will reset.

DEBUG Real time debugger.

DEBUG[switch]<ENTER>

DEBUG is a real-time debugging package for use with machine language programs. DEBUG lets you examine and alter the contents of RAM and the Z-80 registers,jump to a specified address and begin execution with optional breakpoints, single step or perform CALL's, etc.

All values are required to be in hexadecimal form without the "H" suffix.

Once the debugging facility is enabled, it does not load and execute until one of the following conditions occurs:

1. If the interrupts are enabled and the <SHIFT><BREAK> keys are pressed simultaneously.
2. After an executable program is loaded and before its first instruction is executed.

NOTE: Since DEBUG loads into the overlay are of RAM (4D00H to 51FFH), you cannot use it with DOS0, DOS1, DOS2, DOS3, DOS4, DOS7, BASIC/CMD, BACKUP/CMD, FORMAT/CMD, nor RS/CMD.

To return to the point where you entered DEBUG (provided you have not altered the contents of the PC register), type G<ENTER>.

To begin execution at the address in the PC register type G<ENTER>.

DEBUG offers two display formats:
(1) Register display with indirect RAM plus any 64-byte "page" of RAM.
(2) Full screen, 256-byte "page" of RAM.

In the register display format, DEBUG displays all the Z-80 registers, organized for interpretation either as two 8-bit registers or as 16-bit register pairs. Since most programs use several sets of register pairs as indirect pointers or indexing registers, 16 bytes of indirect data are presented with each register pair. Each of the flag registers is shown with an ASCII representation of its flag bits. For these registers, the hex contents of the flag register is displayed, along with a bit-by-bit alphabetic code which makes it easier to interpret the flag status. For example, bit 0 (rightmost bit) is the carry flag, so the alphabetic code shows an C in that position whenever this bit is "set".
Table of codes for all the flag bits:

Bit status	If set
7 Sign	S
6 Zero	Z
5 Unused	space
4 Half-carry	H
3 Unused	space
2 Parity/overflow	P
1 Add/Subtract	N
0 Carry	C

DEBUG Commands

Commands are executed as soon as you press the specified command key or are executed only when you press <SPACE> or <ENTER>, as indicated below.

Command	Entry Required	Operation Performed
C	none	Single-steps next instruction, with CALLS executed in full.
Dqqqq	<SPACE>	Sets memory display starting address to qqqq in register display mode. In full screen mode, sets starting address to qq00 so qqqq is contained in display.
E	none	Produces continuous "C" commands until a key is pressed.
G[jjjj[,kkkk]]	<ENTER>	Place jjjj in PC register and executes with optional breakpoint at kkkk.
I	none	Single-steps next instruction.
M[cccc]	<SPACE>	Sets the current modification address to cccc. Modification information will be displayed in the lower left of the screen. If cccc is omitted, the last modification address will be used for cccc. If cccc is currently in the display, it will be overlayed by a transparent cusor. (see next page).
N	none	Provides continuous "I" commands until a key is pressed.
Rrp aaaa	<SPACE>	Loads register pair rp with the value aaaa.
S	none	Sets display to full screen memory mode.
U	none	Dynamic display update mode. Lets you observe the execution of a foreground task. Press <SPACE> to exit this mode.
X	none	Sets display to register mode.
up arrow	none	Decrements memory display by one page.
down arrow	none	Increments memory display by one page.

NOTE: If you make an error while typing an address, just type the correct address immediately after the incorrect address. DEBUG will only look at the last four digits entered.

EXAMPLE:
DFAE944<SPACE>

Will display the page of memory containing address E944.

The "M" command detail

Any time you wish to alter the contents of a memory location, type Maaaa then <SPACE>. This sets the memory modification address to aaaa and puts a memory modification prompt in the lower left corner of the display. To modify the contents of aaaa, type the new, two-digit contents and press <SPACE>. The display will then be updated, and DEBUG will increment the modification address by one.

To increment the modification address and leave the current address unchanged, type <SPACE> or <COMMA>. To decrement the modification address and leave the current address unchanged, press the back-arrow. Pressing any non-hex key will exit the modify memory mode.

To disable DEBUG, type:
DEBUG switch<ENTER> switch = (N), or (NO), or (OFF)

DEVICE Current I/O devices.

This command lists: KI=keyboard, DO=video display, PR=line printer, and their routine entry points. These are the defined I/O devices.

DIR A specified diskette directory.

DIR[[:]d[(opt[,opt...])]<ENTER>

opt = A, I, K, P, or S

This command in MULTIDOS displays the file directory for a single drive, including the drive number, diskette title, diskette date the number of tracks/"lumps", number of free granules remaining, the K byte granule equivalent, and names of all visible and non-system files on the diskette in alphabetical order. The "P" option will direct the output to the printer as well as the display. If the printer is not ready the system will direct the output to the video display only. The "K" option will display "KILLED" files provided the directory entry location was not overwritten. The "I" option will display the files with the "I" attribute as well as visible files. The "S" option will display the system files as well as visible files. You may enter as many options as you wish. Use of the "A" option will cause the directory display for each file to be expanded to include the level of password protection, the date of the last update to the file, the starting track and sector, end of file sector and byte within the file, number of segments, the logical record length, and size in granules for each file. The protection level will not be indicated for files with the update password blank. Protection levels are designated as follows:

7 = NO ACCESS	3 = (unassigned)
6 = EXECUTE	2 = RENAME
5 = READ	1 = KILL
4 = WRITE	0 = (unassigned) = FULL

EXAMPLE:

DIR 3(A,P)<ENTER>

The following type of output will go to both the display and the printer:

Drive 0, Multidos - 09/01/81, 40/40 tracks, 31 grans, 38.75 K

Filename		Date	Tr,Sec	Eof	Seg	Lrl	Grans
EA/CMD	P=6	09/01/81	04,5	34/184	1	256	7
GR/CMD	P=6	09/01/81	23,5	0/244	1	256	1
RS/CMD	P=6	09/01/81	24,0	4/252	1	256	1
SPOOL/CMD	P=6	09/01/81	23,0	4/254	1	256	1
VFU/CMD	P=6	09/01/81	22,0	9/250	1	256	2

A maximum of twelve lines of files will be displayed at one time. To display the next file, <SPACE>, to display a maximum of twelve additional files, <ENTER>.

DO Substitute disk file for keyboard input.

DO filespec<ENTER>

You can nest DO's until you run out of memory. Probably you would crash before you run out of memory.

This command executes filespec which was previously created with "BUILD". The extension "/IDO" will be appended to filespec if no extension was entered.

DUMP Transfer RAM contents to disk file.

DUMP filespec (START=X'ssss',END=X'eeee'[,TRA=X'tttt'])<ENTER>
ssss, eeee, tttt are 4 digit hexadecimal addresses

DUMP will transfer the contents of memory starting at ssss and ending at eeee to disk. The dump command inserts the extension "CIM" if none was specified in filespec.

ssss must be equal to or higher than 3000H.

eeee must be greater than ssss.

tttt is optional. If entered, it specifies the entry point for execution of the file. If no entry point is specified the system will default to 402DH, MULTIDOS command mode.

If filespec already exists it will be replaced by the contents of the specified area in RAM.

EXAMPLES:

(1) DUMP V (START=X'4000',END=X'4400')<ENTER>

The contents of memory locations 4000H through 4400H will be transfered to a disk file named V/CIM.

(2) DUMP D/CMD (START=X'FD00',END=X'FFFF',TRA=X'FE00')<ENTER>

The contents of memory locations FD00H through FFFFH will be transferred to a disk file named D/CMD. If MULTIDOS attempts to execute this file it will start execution at FE00H.

FORMS Sets printout parameters.

FORMS[(param)[,param1][,param2)]<ENTER>

Whereas param:

I	Intializes line counter and character counter to zero.
W=XXX	XXX=1 to 255 which is the maximum width of print line in characters, 255 inhibits 'FORMS' character counter.
P=XXX	XXX=1 to 255 which is the page length in print lines.
T=XXX	XXX=1 to 255 which is the printed TEXT in print lines.
S=XXX	XXX=1 to 127 which is the blank spaces between printed TEXT.

Only 2 of P,T, & S are required for setup. Priority is P, T, then S.

EXAMPLE:

FORMS (I,W=80,T=60,P=66)

1. The line and character counters are set to zero.
2. The print width is 80 characters.
3. The printed text length is 60 lines.
4. The page length is 66 lines.

NOTE: If "P" and "T" are the same, pagenation is inhibited.

Powers up with T=66 and P=66 (no pagenation)

FORMS<ENTER>

Will display the current FORMS parameters.

FREE Displays the number of available file spaces, granules and K byte granule equivalent of disk space on all mounted diskettes.

FREE<ENTER>

This command displays the drive number, the diskette's name and date, the number of available files remaining, and the number of free granules and the equivalent K bytes (1024 bytes) on each mounted diskette.

HASH Returns HASH code of filespec.

HASH BASIC/CMD<ENTER>

Returns:
Hash code = F0

KEYBRD Sets keyboard attributes.

1. LOWER CASE	KEYBRD (L=Y) or (L=N)
2. GRAPHICS DRIVER	KEYBRD (G=Y) or (G=N)
3. REPEATING KEYBOARD	KEYBRD (R=Y) or (R=N)
4. "CLEAR" Key	KEYBRD (C=Y) or (C=N)
5. BLINKING CURSOR on	KEYBRD (B=Y) or (B=N)
6. CURSOR CHARACTER	KEYBRD (W=nnn) where nnn=decimal.

Syntax is forgiving - only the first letter is considered. KEYBRD (LOWER=YEP) will flag DOS to check for extra RAM chip and route drivers to display lower case if found.

KEYBRD<ENTER> echo's system disks current attributes.

EXAMPLE:
KEYBRD (L=Y,C=N,R=Y,W=176)

On future reboots/power-ups:
1. The lower case driver is activated.
2. The clear key is defeated.
3. The repeating keyboard is on.
4. The cursor character is 176 decimal.

The graphics driver and blinking cursor remain as configured prior to this command.

LIBRARY COMMANDS

KILL Delete a filespec.

KILL filespec<ENTER>

This command will reset the directory in use bits and deallocate diskette space previously allocated to this filespec. The diskette which contains the filespec must not have it's write protect notch covered. If a drivespec is not included in the filespec, the system will search for the first drive containing the filespec and delete it.

LIB Displays the **MULTIDOS** operating system library commands.

LIB<ENTER>

The library commands load between hexadecimal 5200H and 68FFH except 'BOOT', 'BREAK', 'CLOCK', 'DEBUG', 'LIB', and 'VERIFY', which are contained in DOS1/SYS. (4D00H to 51FFH)

LINK Simultaneous output of printer and display.

The four possible inputs are:

LINK PRDO<ENTER>
LINK DOPR<ENTER>
LINK PRPR<ENTER>
LINK DODO<ENTER>

LINK PRDO will send to the printer whatever goes to the display.
LINK DOPR will send to the display whatever goes to the printer.

To un-LINK either, enter: LINK DODO or LINK PRPR

LIST Display diskette file.

LIST filespec<ENTER>

This command lists a file on the display. If the file contains control codes, the display is suppressed. The <BREAK> key will stop the listing of a file.

LIBRARY COMMANDS

LOAD Place an object file from diskette into RAM.

LOAD filespec<ENTER>

This command loads the specified filespec into RAM and returns control to **MULTIDOS**. With **MULTIDOS**, you can load down to memory location 5200H and retain the operating system. However, if your program loads below 6900H, you cannot use some **MULTIDOS** library command without overlaying your program.
(Permitted are 'BREAK', 'CLOCK', 'DEBUG', 'LIB', & 'VERIFY'.)

PRINT Printout a file onto the printer.

PRINT filespec<ENTER>

This command will print out the filespec onto the line printer.
NOTE: If the file to be printed is not saved in ASCII format, your printout and printer behavior is unpredictable, as many of the characters may be interpreted as linefeeds, tabs, form feeds, etc.

PROT

PROT[[:]d] (param[,param...])<ENTER>

param	meaning
LOCK	assign MASTER PASSWORD to all user files
UNLOCK	remove all passwords from user files
PW	change the MASTER PASSWORD

This command will change the diskette MASTER PASSWORD or lock or unlock all visible and non-system files on the diskette. If drivespec is not included, drive 0 will be used The drivespec referenced must not have it's write protect notch covered.

EXAMPLE:
PROT (LOCK)<ENTER>

This will assign the MASTER PASSWORD to all user files
(non-system and visible files).

RENAME Change filename.

RENAME filespec1 to filespec2<ENTER>

This command will change the name of filespec1 to filespec2.
filespec2 will contain the protection level, password, and directory attributes of filespec1. An error message will appear if the name of filespec2 already exists on the diskette. filespec2 should not have a drivespec specified.

LIBRARY COMMANDS

ROUTE Redirects printer to display or display to printer.

EXAMPLE:

ROUTE PRDO<ENTER>
Will send to the display anything directed to the printer.

ROUTE DOPR<ENTER>
Will send to the printer anything directed to the display.

To UN-ROUTE you MUST specify the correct device. After ROUTE PRDO only ROUTE PRPR is acceptable.

TIME Sets time.

(1) TIME hh:mm:ss<ENTER>
(2) TIME<ENTER>

If version (2) is used, MULTIDOS will display, on a one time basis, the time currently stored in RAM. A non-powerup REBOOT will retain the time previously set in RAM. The hexadecimal locations 4041H - 4043H store the current time, in ss, mm, hh.

TOPMEM Sets upper MULTIDOS system memory.

(1) TOPMEM ddddd<ENTER>
(2) TOPMEM Hnnnn<ENTER>
(3) TOPMEM<ENTER>

ddddd = A decimal number from 28671 to 65535.
nnnn = A hexadecimal number from 6FFF to FFFF.(NOTE: Precede by "H").

The above command sets the upper limit of user free memory available to the operating system. It is useful if you have some high memory drivers which you want to protect. MULTIDOS programs, such as SUPERBASIC, EA/CMD, and others check the value of TOPMEM and operate at that limit. The value is placed in RAM at locations 4049H & 404AH. The default value is the top of system RAM. When version (3) is entered, MULTIDOS will display, in decimal, the highest usable byte currently available for MULTIDOS, MULTIDOS utilities, and SUPERBASIC usage.

Note: MULTIDOS and SUPERBASIC accept and use TOPMEM. They do not use the top 64 bytes of system RAM.

VERIFY Reread at written sector.

VERIFY switch<ENTER>

This command will cause all disk writes to be reread for parity. All directory writes and logical writes are verified.

BACKUP/CMD Duplicate a diskette.

This utility will duplicate all files from one diskette to another requiring the use of one or more drives. The source diskette is the diskette which contains the files, and the destination diskette is the diskette which the files are to be duplicated on. The source diskette drive and destination diskette drive may be the same drive number. If the source and destination drive are the same, BACKUP will prompt you when to mount the source or destination diskette (swapping). To prevent re-writing on the source diskette when swapping is required, it is recommended that a write protect tab by placed over the write protect notch of the source diskette.

The BACKUP utility is menu driven and will take you through the easy procedure to duplicate a diskette.

BACKUP<ENTER> or CMD"BACKUP"<ENTER> from SUPERBASIC

the screen will clear and

'MULTIDOS Disk Duplicator Program - Version 2.0

Q1 Which drive contains the source diskette? '

will appear. The program is awaiting a numerical response of 0, 1, 2, or 3 followed by <ENTER>. Respond with the drive number which will contain the source diskette. If the response was 0, 1, 2, or 3 then:

Q2 'Which drive for the destination diskette? '

will appear. Again the program is awaiting a numerical response of 0, 1, 2, or 3 followed by <ENTER>. Respond with the drive number which will contain the destination diskette. If the response was 0, 1, 2, or 3 then:

Q3 'Press "ENTER" when the source diskette is in drive X.'

will appear. Whereas X is the response to the first query (Q1). Mount the source diskette into drive X, if it is not already there, then <ENTER>. MULTIDOS will analyze the source diskette for track count and density, then display:

'The source diskette has YY tracks, in ZZZZZZ density.

Q4 Track count for the destination diskette (35 to 96)? '

Where as "YY" is the number of tracks on the source diskette and "ZZZZZZ" is the density of the source diskette. The destination diskette will be formatted in "ZZZZZZ" density; however, you may specify the track count for the destination diskette. If a null, just "ENTER", is pressed for the track count, BACKUP will use "YY" for the track count.

If the entered track count is insufficient to copy all files, BACKUP will display:

'Insufficient track number to copy all files!'

then revert to the third query (Q3).
If the track count response is acceptable, BACKUP will display:

'Press "ENTER" when the destination diskette is in drive W.'

whereas "W" is the response to the second query (Q2). If the destination drive is the same as the source drive, you MUST swap the diskettes. If the destination diskette was previously formated then:

'Diskette previously formatted.

Q5 Do you want to re-format this diskette? '

will appear. If you want to abort BACKUP, <BREAK>, if you want to re-format this diskette, enter "Y", if you want BACKUP without formatting, <ENTER>.

BACKUP does not check the destination diskette for a density match with the source diskette. If the destination diskette had been formatted with a different density, you MUST respond "Y" to the fifth query (Q5).

The destination diskette will be formatted if no address marks are found or a "Y" response was entered for the fifth query (Q5). The formatting will proceed then BACKUP will verify the destination diskette. BACKUP does not allow any flaws on a destination diskette.

If the source drive and destination drive are the same, BACKUP will ask you to insert the source diskette and destination diskettes as required. The swapping will continue as necessary until all the files are copied onto the destination diskette.

After BACKUP has copied all files

'Completed
Press "ENTER" when a MULTIDOS system diskette is in drive 0'

will be displayed, prompting you to insert a MULTIDOS diskette to return to the state prior to entering BACKUP.

BACKUP will duplicate the following diskettes

1. TRSDOS	6. DOSPLUS, MOD III	11. MULTIDOS, S-DEN
2. NEWDOS, NEWDOS+	7. DBLDOS	12. MULTIDOS, P-DEN
3. ULTRADOS	8. LDOS	13. MULTIDOS, D-DEN
4. DOSPLUS, single density	9. LDOS, double density	14. See below
5. DOSPLUS, double density	10. LDOS, MOD III	15. See below

For 14 and 15, BACKUP will duplicate a NEWDOS 80 version 1.0 diskette, with a 2 granule directory, (the NEWDOS 80 version 2.0 author decided to ignore the lock out table, WRONG!; therefore , BACKUP cannot duplicate these diskettes. However, MULTIDOS will READ any files from these diskettes, provided the directory is only 2 grans).

It actually takes very few keystrokes to backup a diskette. For example, to duplicate a diskette in drive zero to drive 1, would require the following keystrokes: (after BACKUP is entered)

0<ENTER>, 1<ENTER>, <ENTER>

COPY/CMD Duplicate a single file.

This utility will copy a file from one diskette to another. The diskette which contains the file will be referred to as the source diskette. The diskette in which the file will be placed on will be referred to as the destination diskette. The source and destination diskettes may be:

1. The same diskette.
2. Diskettes to be mounted in the same drive.

or 3. Diskettes in two different drives.

This utility requires ALL drivespecs. If the filespec for the destination diskette is the same as the filespec for the source diskette then the filespec need not be repeated.

(1) COPY CHARGES/TXT:1 TO :2<ENTER>

If the destination diskette's drivespec is the same as the source diskette's drive spec, you will be prompted to mount the source or destination diskette (swapping).

(2) COPY :3 SHIFT/TXT TO MURK/ABC<ENTER>
(3) COPY SHIFT/TXT:3 TO MURK/ABC:3<ENTER>

Both of the above two command entries will duplicate the contents of SHIFT/TXT into MURK/ABC on drive number three.

System diskettes are MULTIDOS system diskettes with at least, DOS0, DOS1, DOS2, DOS3, and DOS4. Alien diskettes are diskettes with a system other than MULTIDOS. Data diskettes are diskettes without a system.

Whenever drive zero is specified and the source or destination diskette is not a system diskette, a "$" MUST precede the source filespec.

(4) COPY $WHENEVER/BAS:0 TO :2
(5) COPY :0 $HELPME/CIM TO SHOWME/CIM
(6) COPY $ THEM/OLD:0 TO THEM/NEW:1
(7) COPY $ MANUAL/TXT:3 TO :0

The "$" designator in COPY will permit you to copy file[illegible] alien/system/data diskette to any other alien/system/data diskette[illegible] multiple swapping will be required to bring in the correct system[illegible] Please follow the prompting provided by COPY. Whenever the prompt:

'Press "ENTER" when a MULTIDOS system diskette is in drive 0.'

appears, remove the source or destination diskette and insert your MULTIDOS system diskette, then press "ENTER".

COPY will carry over the source filespec's date. However, if you want the destination filespec to have the current RAM date, place a "#" immediately in front of the source filespec.

(8) COPY #CHECKING/BAS:1 TO CHECKING/BAS:2
(9) COPY #CHECKING/BAS:0 TO :0
(10) COPY :2 #CHECKING/BAS
(11) COPY :2 #CHECKING/BAS TO CHECKING/BAS
(12) COPY :0 $ #CHECKING/BAS

TECHNICAL NOTES
COPY will execute in the following sequence.

1. Check for "$".
2. Check for "#".
3. Position the source filespec in COPY's DCB-1
4. Insert source drivespec into DCB-1 then check for validity.
5. Position the destination filespec in COPY's DCB-2.
6. Insert destination drivespec into DCB-2 then check for validity.
7. Check for swap.
8. OPEN source filespec.
9. Stores source filespec sector allocation.
10. READ in as much of source filespec into available RAM.
11. Check for swap.
12. Attempts to OPEN the destination filespec, if found stores current sector allocation.
13. Calculates and stores the amount of free sectors on the destination drive. (converts granules)
14. If the destination filespec existed, adds and stores the total number of sectors available.
15. Compares the total sectors available on the destination drive with the source filespec's sector allocation.
16. Abort COPY if insufficient space is on destination diskette.
17. If the destination filespec didn't exist INIT the destination filespec.
18. Check for "SYSTEM" swap.
19. Allocate all sectors (via granule allocation) on the destination diskette.
20. WRITE out to the destination filespec as much of the source filespec read into RAM in step 10.
21. Check for swap if all of source filespec was read into RAM and written out to the destination filespec.

22. Check for "SYSTEM" swap.
23. Check for swap.
24. Close destination filespec.
25. Check for "SYSTEM" SWAP.
26. Display error if any.
27. Return to MULTIDOS or SUPERBASIC.

If an error occurs during steps 4, 6, 8, 10, 16, 17, 20, or 24, then COPY will jump to step 25.

For those users who have operated other COPY utilities, please appreciate that COPY assumes all nonconflicting drivespec diskettes are mounted prior to executing file duplication.

EXAMPLE:

(13) COPY FUNLOVER/TXT:2 TO :3

Does NOT prompt the user to mount any diskettes.

FORMAT/CMD Prepare a diskette for data storage.

This utility will prepare a diskette, without the MULTIDOS operating system, which will leave the maximum disk space for your files. These diskettes are referred to as "DATA" diskettes.

FORMAT<ENTER> or CMD"FORMAT"<ENTER> from SUPERBASIC

Operation is as follows:

'MULTIDOS Formatter Program - Version X.X

(1) Which drive contains the diskette to be formatted? '

Reply with 0,1,2, or 3.

(2) 'Single, Double, or "P" density (S,D, or P)? '

Reply with S, D, or P.
Will default to current system configuration. If "D" is specified then the query

(3) 'Density for track zero (S or D)? '

will be displayed. (Defaults to single density)

(4) 'Track count for this diskette (2 TO 96)? '

Reply with the desired track count. The program can only FORMAT up to the drive capacity. If you only have a 35 track drive, you cannot format 40 tracks. If you do not enter a track count, MULTIDOS will default to 40 tracks.

(5) 'Which track for the directory (1 to XX)? '

XX = The reply to question # 4 less one.

The system will default to track 17 decimal (11 hex)

(6) 'Name of diskette to be formatted?'

Reply with the desired diskette name (DATA1, BASICPRO, etc.). The name may be 1-8 characters in length. The system will default to "DATA ".

(7) 'Date for the diskette to be formatted?'

Reply with a date in the format mm/dd/yy. Defaults to RAM date.

(8) 'The master password for this diskette? '

Reply with the password desired for this diskette. The password may be 1-8 characters in length. Defaults to "PASSWORD".

If the diskette to be formatted contains data, MULTIDOS will suspend formatting. Only a "Y" response will cause formatting to continue.

Formatting will proceed. You will be kept advised of the progress via screen messages. MULTIDOS will lock out granules (if the diskette has flaws) and bump the verifying counters to the next granule.

EA/CMD Editor Assembler.

This utility is Radio Shack's tape version which had been modified by Apparat Inc. for disk I/O. Further enhancements to this program were implemented starting January 1980. Since this program is not a MULTIDOS "original", complete documentation will not by supplied. It is required that the user purchase Radio Shack's tape (not disk) Editor Assembler for complete documentation.

The utility is initialized by typing EA<ENTER> from the MULTIDOS command mode or CMD"EA" from SUPERBASIC

The first prompt will be 'Protect memory from (dec)?' At this point the user should:

1. Press "ENTER" if no other machine language program is in high memory or a machine language program is in high memory with "TOPMEM" properly set.

2. Press "BREAK" to recover previous source code. This can be accomplished after an inadvertent reboot or exit from the utility. This function will use the previous "TOPMEM".

3. Enter the lowest decimal number in which a machine language program occupies in high memory.

NOTE: DO NOT ENTER A NUMBER IF EA/CMD IS ENTERED FROM SUPERBASIC. Any machine language program in high memory prior to entering SUPERBASIC must have its memory protected when entering SUPERBASIC.

There are three additional commands with this version.

1. C copy filespecs
   *C filespec to filespec<ENTER>
   Both filespecs are required and drive specs are recommended.

2. K kill filespec
   *K filespec<ENTER>

3. V View directory
   *V drn<ENTER>
   drn = 0, 1, 2, 3 without a colon (:).
   If drn is not specified, 0 will be used. Due to RAM limitations, the directory will scroll if more than 60 files are on a diskette.

GR/CMD

This MULTIDOS utility allows direct keyboard entry of graphics characters. Enter the command "GR" from the DOS ready mode, do NOT enter from SUPERBASIC! To shift to graphics mode, press the <SHIFT> key and the <CLEAR> key at the same time. Graphics characters are now available as shown in table 4-1. These characters may be directly entered into strings in BASIC.

Graphics

(pixels lit)

0 = 1 = 2 = 3 = 4 = 5 = 6 =

7 = 8 = 9 = A = B = C = D =

E = F = G = H = I = J = K =

L = M = N = O = P = Q = R =

S = T = U = V = W = X = Y =

Z = up-arrow =

lowercase (or SHIFTED without lowercase keyboard)

a = b = c = d = e = f = g =

h = i = j = k = l = m = n =

o = p = q = r = s = t = u =

v = w = x = y = z =

Table 4-1

RS/CMD RAM Scanner.

This utility will scan the memory from 0000H to FFFFH and attempt to locate an 8 bit byte or 16 bit word specified by the user. The utility will ask:

'START?'

Enter the starting address, in hex, at this time. Next the utility will ask:

'STOP?'

Enter the ending address, in hex. Next the utility will ask:

'BYTE, OR WORD SEARCH (B/W)?'

For a one byte, search enter "B". For a two byte search, enter "W". Next the utility will ask you to enter the search target. Enter the target at this time. The input must consist of two or four hexadecimal characters, without the "H" suffix, and must not contain any blanks. Acceptable words would be "00AB", "DE09", etc. Acceptable bytes would be "74", "09", etc.

Next, for word scans only, the utility will ask:

'Enter auxiliary mnemonic '

You can optionally inquire about calls, jumps, and loads to a selected word at this time.

Enter one or more of the following characters and the <ENTER> character:

C = CALL/CARRY
J = JUMP
L = LOAD
M = SIGN MINUS
N = NON
P = SIGN POSITIVE
PE = PARITY EVEN
PO = PARITY ODD
Z = ZERO

The above terms may be combined if needed. If no auxiliary mnemonic is desired press <ENTER>. For the "L" (LOAD) command, the following question will be asked:

'IMMEDIATE "I", I OR DIRECT "D"?'

If the response is "D", the utility will ask:

'"F" FROM or "T" TO the register?'

Answer as desired. Finally the utility will ask:

'Accumulator, A or register pair - BC, DE, HL, SP, IX, IY :'

Enter the desired register pair.

The utility will then display the hexadecimal locations where the specified byte or word is found.

EXAMPLES: let START = 0000, STOP = 2FFF (ROM)

(1) Enter search word 3C00<ENTER>
Enter auxiliary mnemonic <ENTER>
04C1 0555 06F2 2080
Function completed - Press "R" to scan again,
"CLEAR" to return.

The word 3C00 was referenced in rom at locations 04C1H, 0555H, 06F2H, and 2080H.

(2) Enter search word 4000<ENTER>
Enter auxiliary mnemonic J<ENTER>
0005 0008
Function completed - Press "R" to scan again,
"CLEAR" to return.

The "JP 4000H" command is found at locations 0005H and 0008H in ROM.

SPOOL/CMD

This MULTIDOS utility is designed to allow the computer to function at almost full speed without delays for the printer to function. The utility provides a variable RAM buffer for parallel printers and provides more effective utilization of the printer control codes.

SPOOL<ENTER>

The following questions will appear. Respond as indicated.

(1) 'How many 256 byte blocks for spooling (1-99)? '

This lets you set the buffer size for the spooler. The size is selected in 1/4 K increments. As an example, to reserve a 2K buffer the reply to the above question would be 8<ENTER>.

(2) 'PAGENATE (Y/N)? '

Reply "N" if you do not want automatic paging or if your program pokes the line printed device control block to simulate paging. Reply "Y" if you want the spooler to control paging.

(3) 'Print lines per page (1-99)? '

If you replied "Y" to question 2, you will be asked to select the number of printed lines per page. This is not the total number of available lines per page, only the number of printed lines.

(4) 'Blank lines per page (1-15)? '

If you replied "Y" to question 2, you will be asked to select the number of blank lines between pages. The sum of the answers to questions 3 & 4 should equal the total page length.

(5) 'Characters per line (1-255)?'

Select the desired line length. The spooler will send a linefeed/carriage return after that number of characters if the system has not already sent one. You should not select a line length longer than the maximum allowed by your printer. A response of 255 will override this feature.

(6) 'SPOOL (Y/N)? '

Answer "Y" if you want the printed output to be sent through the buffer for better system response. If you answer "N", spooling will not take place, but SPOOL/CMD will still control the output page parameters.

(7) 'ENTER memory size (DECIMAL) YOU WANT TO PROTECT. '

Use this to protect a high memory routine which does not protect itself by setting TOPMEM (4049H). Press <ENTER> to use the value MULTIDOS has established in TOPMEM. <BREAK> to abort spooler.

The spooler will now commence operation and control of any printed output. To suspend output, press shift <BREAK>. You will be asked if the buffer is to be saved and will be prompted again to enter the spooler parameters.

The spooler will recognize CHR$(11) as a linefeed without a carriage return. CHR$(12) will generate a formfeed. CHR$(10) & CHR$(13), will be treated as linefeeds with carriage returns.

The spooler will ignore ROM generated linefeeds/carriage returns and will format the printout per your program and spool parameters within the limits of your printer capability.

VFU/CMD

This versatile file utility (VFU) provides for four frequently needed disk operations: multiple file copying, purging of files, printing a disk directory, and menu based execution of all programs on a disk.

INITIALIZATION & EXIT

To initialize the utility from MULTIDOS, use the following instruction:

VFU<ENTER>

To initialize the utility from SUPERBASIC, use the following instruction:
CMD"VFU"<ENTER>

When properly initialized, the utility will clear the screen and prompt with the following message:

File utility, BY V. B. Hester, - Version X.X Press
"C" = COPY, "E" = EXECUTE, "P" = PURGE, OR "H" = HARD COPY.

A winking cursor will indicate that user input is required.

To exit the utility when the winking cursor is displayed, press <CLEAR>. To exit the utility when the normal cursor is displayed, press <BREAK>. If you entered the utility from SUPERBASIC via the 'CMD"VFU" mode, you will be returned to SUPERBASIC. If you entered the utility from MULTIDOS, you will be returned to the DOS ready mode.

VFU COPY COMMAND

To copy files from one drive to another, use the "C" command. The utility will then prompt:

Press "S" for SELECTIVE, or "T" for TOTAL.

To copy the entire diskette, press "T". To select the files to be copied, press "S". Now the utility will ask if invisible and system files are to be considered in the copy operation with the following prompts:

Include "INVISIBLE" files?
Include "SYSTEM" files?

Answer "Y" or "N" as appropriate. Finally, the utility will request the source and destination drives with the following prompt:

Source drive? to destination drive?

NOTE: the source and destination drive cannot be equal.

If the selective option was chosen, the directory will be displayed with the winking cursor next to the first file name. If that file is to be copied, press "Y". A "+" will appear in front of the file name to indicate that it is to be copied and the winking cursor will move to the next file. If you do not want to copy the file, press "N", space bar or right arrow. The cursor will move to the next file. The four arrow keys may be used to move the cursor.

During the file selection process, the left arrow will reposition the cursor to the previous filename, removing the "+" if present. The shifted left arrow will reposition the cursor to the first filename, removing all "+" present.

If the "T" option was selected, the "+" sign will appear in front of all filenames.

After you have decided what files are to be copied, the following prompt will appear:

Press "A" TO ABORT, "ENTER" TO EXECUTE, or "R" TO REPEAT.

Press <ENTER> to start the copying function. If you do not want to copy, press "A". To prematurely terminate the copying function, hold down a shift key. The utility will complete copying the current file and terminate before it starts to copy the next file.

VFU EXECUTE COMMAND

To execute programs from a diskette, use the "E" command. The utility will then prompt:

```
Drive x      press "Y" to execute to "N" to skip.
 Disk            - MULTIDOS - 03/12/81,    6 FREE GRANS.
```

Next an alphabetical directory will be displayed. Use the arrow keys to position the winking cursor in front of the file to be executed and press "Y". If the selected filespec has the "/CMD" extension, VFU will load and then execute that filespec. If the "/CMD" extension is not present, VFU will load BASIC and then load and attempt to run the filespec.

VFU PURGE COMMAND

To purge files from a diskette, use the "P" command. The utility will then prompt:

Press "S" FOR SELECTIVE, or "T" FOR TOTAL.

To purge all files from the diskette, press "T". To select the files to be purged, press "S". Next the utility will ask if invisible and system files are to be considered for purge action with the following prompts:

Include "INVISIBLE" files?
Include "SYSTEM" files?

Answer "Y" or "N" as appropriate.

Finally, the utility will request the drive number with the following prompt:

Drive number?

If the selective option was chosen, the directory from the selected drive will be displayed with the winking cursor next to the first file name. If that file is to be purged, press "Y". A "+" will appear in front of the file name to indicate that it is to be purged and the winking cursor will move to the next file. If you do not want to purge the file, press "N", space bar or right arrow. The cursor will move to the next file.

During the file selection process, the left arrow will reposition the cursor to the previous filename, removing the "+" if present. The shifted left arrow will reposition the cursor to the first filename, removing all "+" present.

If the "T" option was selected, the "+" sign will appear in front of all filenames.

After you have decided what files are to be purged, the following prompt will appear:

Press "A" TO ABORT, "ENTER" TO EXECUTE, or "R" TO REPEAT.

Press <ENTER> to start the purge function. If you do not want to purge, press "A". To prematurely terminate the purge function, hold down a shift key. The utility will complete purging the current file and terminate without purging any additional files.

After the purge process is completed, the revised directory of the diskette will be displayed.

VFU PRINT DIRECTORY COMMAND

To print a diskette directory, use the "H" command. The utility will ask if invisible and system files are to be included in the directory printout and which drive is to be used. Then it will ask for the name or number of the diskette (8 characters maximum). The directory will be displayed on the screen. Finally, the utility will prompt with the following message:

Press "A" TO ABORT, OR "ENTER" TO EXECUTE.

A reply of <ENTER> will cause the directory to be printed. If a 10 character/inch printer is used, the printout will be sized to fit inside the diskette jacket.

BASIC High level language interpreter.

SUPERBASIC (file name BASIC/CMD) is a MULTIDOS command file which contains the code to enhance/control the ROM, and adds disk input/output capabilities.

BASIC[ff[V]][,mm][,command]<ENTER>
This command will load **SUPERBASIC** into your system, from the Multidos ready prompt.

ff = An optional number of file buffers to be opened by basic (0–15).

If you want to use RANDOM disk I/O, and a record length other than 256, then V must be appended to the number of file buffers.

mm = An optional upper memory limit which can reserve space for machine language subroutines.

command = Any valid BASIC command.

NOTE: The default values are 3 files and all memory to TOPMEM available to **SUPERBASIC.**

This version of **SUPERBASIC** is shorter than any other DISK BASIC, leaving more room for your program.

EXAMPLES:

(1) BASIC<ENTER>

SUPERBASIC will be loaded with 3 file buffers open and all of RAM up to TOPMEM available for use.

(2) BASIC 4<ENTER>

SUPERBASIC will be loaded with 4 file buffers open and all of RAM up to TOPMEM available for use.

(3) BASIC 2V,60000,RUN "PROG1/BAS"<ENTER>

SUPERBASIC will be loaded with 2 file buffers open. RAM from 60000 up will not be available to **SUPERBASIC.** The program "PROG1/BAS" will be loaded by **SUPERBASIC** and RUN. With the V appended to the number of file buffers (2V), user defined record lengths are permitted. However, normal sequential disk I/O is permitted if the V is specified or not.

For each file buffer opened, **SUPERBASIC** will reserve 289 bytes.
33 for a DCB, and 256 for the I/O buffer. If the V parameter is specified, an additional 256 bytes are reserved. One will be used for "FIELD"ing and the other for disk I/O.

BASIC * Recover BASIC program.

BASIC *<ENTER>

This command assumes SUPERBASIC was previously loaded into your system, you now have the MULTIDOS prompt, and you want to return to SUPERBASIC with the previous program unchanged with variables intact.

CAUTION: Because SUPERBASIC has expanded "CMD" functions, and makes additional memory space available, you cannot go from SUPERBASIC to MULTIDOS, execute commands such as "DIR" and then use "BASIC *" to return. In such cases you should use the CMD"uuuuu" function from within SUPERBASIC.

If the return to SUPERBASIC was successful, a 'Continue? ' prompt will appear. Enter 'Y' if you want the program to continue (even after re-boots).

BASIC I Down load a BASIC program.

BASIC I<ENTER>

This unique command permits you to transfer a BASIC program which has been loaded via an alien operating system or SUPERBASIC, with at least 1 file buffer, to SUPERBASIC, without saving the program on disk.

With a BASIC program in RAM, insert your MULTIDOS diskette into drive 0, hold down the enter key, and press reset. When the MULTIDOS prompt appears, enter "BASIC I". SUPERBASIC will be initialized with zero file buffers, previous TOPMEM, and the program text retained.

BASIC # Recover a LEVEL II BASIC program.

BASIC #<ENTER>

This command is used to transfer a program from LEVEL II BASIC to SUPERBASIC, providing proper entry to LEVEL II BASIC was first made via CMD"X" from SUPERBASIC. This feature will allow you to work in a LEVEL II environment to develop programs or to transfer a sensitive LEVEL II program from tape to disk.

MULTIDOS uses SUPERBASIC'S CMD"X" function to enter LEVEL II BASIC. You may use CMD"X" with or without a program in RAM. If a program is in RAM, it will be transferred to LEVEL II BASIC with text retained. This function is provided in lieu of "BASIC2", since it can completely incorporate the "BASIC2" function while maintaining the program in RAM.

To exit from LEVEL II BASIC to SUPERBASIC with the program intact, type "SYSTEM", answer the "*?" With "/16480", hold down the enter key and wait for the MULTIDOS prompt. Enter "BASIC #" as shown above.

SUPERBASIC will load with your program's text retained.

BBASIC Enhanced SUPERBASIC

BBASIC has all of the features of SUPERBASIC with the addition of all of BOSS's single step, trace, and variable review functions. Even with the incorporation of all of these functions, BOSS-SUPERBASIC uses less RAM than any other disk basic. Upon entering BBASIC, a new keyboard driver is activated, changing the key with the "@" symbol to a control key. The "@" character is printed by pressing SHIFT-SPACE-BAR.

There are seven additional functions available with BBASIC. These functions are invoked by pressing down the key with the "@" symbol first then, without removing your finger from this key, press one of the keys labeled 1 thru 7. The same functions can be obtained thru program execution by poking an appropriate number into 16667 decimal or 411B hex. (This is ROM's trace on/off byte.) With the incorporation of these new trace functions, the TRON and TROFF functions are disabled.

The particular functions will be discribed with the "@" symbol preceding a number 1 thru 7. This represents pressing the "@" key along with the numbered key following the "@".

@1 TRACE OFF
This will turn off all trace functions.

@2 TRACE ON - VIDEO
This function will trace line number execution in the upper right hand corner of the display. The last four lines executed will be displayed. As a new line is entered, its number will appear prefixed by the "#" sign. If more than four lines have been executed the trace will start at the top and overprint the previously displayed trace functions.

@3 TRACE ON - PRINTER
This function will direct the trace to the printer in the format " #####", requiring 6 characters for each line.

NOTE: The trace information will show the complete program flow, but will only be printed when an entire print line is available. This is because the printer will not output any data until a complete print line is sent. If a character at a time printer is being used, the trace information will be printed as each program line number is executed. The print format for a trace to printer is space, 10k digit, 1k digit, 100's digit, 10's digit, and unit digit. Zero suppression is used and each line requires six character positions. This fits in well with 72 character per line (12 line traces) and 132 character per line (22 line traces) printers. This output can be controlled by any type of user linkup to the printer device control block, since it is raw space/numeric output. No linefeeds, carriage returns, or control characters are sent.

@4 SINGLE STEP OFF
This function will turn off all single step functions.

@5 SINGLE STEP TO THE END OF LINE
This function will execute one program line then wait.

@6 SINGLE STEP INSTRUCTION
This function will execute one BASIC instruction then wait.

@7 SINGLE STEP WITH TIMED WAIT
This function will execute one program line or instruction, pause for a predetermined amount of time, then continue.

SINGLE STEPPING

You can single step individual lines of a BASIC program or individual instructions within a line. In addition, you can vary the delay in which your program steps between lines or individual instructions. There are four single step commands.

SS1 - Single step off
Pressing "@4" will turn off the single step function and allow your program to run as normal. If the trace function was in use, it will continue to function until turned off.

SS2 - Single step to end of line
Pressing "@5" will cause your program to pause at the end of each line until any key is pressed. The trace to video display mode will also be initiated to show you which line number is being executed. This trace mode can be disabled by a "@1", while the single step mode continues.

SS3 - Single step instruction
Pressing "@6" will cause your program to pause when an instruction separator, :, is found and at the end of each line. Press any key to continue to the next instruction. Again, the trace to video display mode will be initialized to show you which line number is being executed. This function can be useful, but be wary of using it if a program contains lines such as:

```
90 FOR X=1 TO 100:A(X)=6+3*X*Z:NEXT X
```

To single step through this loop would require 300 presses of a key. Instead use "single step to end of line."

SS4 - Variable delay step (auto step)
Pressing "@7" will cause your program to delay approximately 0.25 seconds at the end of each line. Again the trace to video will be invoked to show you which line number is being executed. "@5" and "@6" become sub-commands after "@7" is initiated. Pressing "@6" after "@7" is initiated will cause the delay to occur at an instruction separator, in addition to the end of a line. Pressing "@5" will cause the delay to occur at the end of a line only. This delay has nine settings from approximately 4 milliseconds to approximately 0.9 seconds. To speed up execution (decrease delay) press "@up-arrow". To slow down execution (increase delay) press "@down arrow". The amount of delay can be adjusted any time after BBASIC is initialized. The initial setting provides 0.25 seconds delay. The amount of delay is halved each time "@up-arrow" is pressed, or doubled each time "@down-arrow" is pressed. During this delay, key presses are not recognized.

BREAK POINTS

The trace and single step commands previously described can be invoked by your program while it is running by inserting a POKE instruction in your program at the location where you want to invoke the command. The following codes are used:

FUNCTION	POKE 16667,
TRACE off	1
TRACE to display	2
TRACE to printer	3
SINGLE STEP off	4
SINGLE STEP line	5
SINGLE STEP instruction	6
SINGLE STEP delay	7

EXAMPLES OF BREAK POINT USE

If you want normal program execution to line 1540, then single stepping with trace to the screen, insert just prior to line 1540, the instruction "POKE 16667,5".

EXAMPLE 1	EXAMPLE 2
1530 (users text)	1530 (users text)
1535 POKE 16667,5	1540 POKE 16667,5: (users text)
1540 (users text)	

NOTE: If your program logic has GOTO's, GOSUB's, etc., be sure to position the break point where the code will be executed. Multiple POKE's are permitted; POKE16667,7:POKE16667,6:POKE16667,1
This will invoke "variable delay step" between instructions with the trace disabled.
NOTE: You can insert as many break points in your program as you desire.

REVIEWING VARIABLES

BBASIC will suspend program execution to review selected variables and then return to your program with the display restored to that shown before you reviewed the variables. There are two commands used for this function.

@N = select variables for review
@O = review the selected variables

SELECTING VARIABLES

Pressing "@N" will allow you to select the variables you want to review during program execution. This command can be entered at any time before you run the program or during program execution. After invoking "@N", the query 'Enter maximum length? ' will be displayed. Respond from the following choices:

RESPONSE	RESULT
BREAK	exit function & return to BASIC program
1	1 character variable names
2 or 3	maximum of 3 character variable names
4 - 7	maximum of 7 character variable names
8 - 15	maximum of 15 character variable names
16 - 31	maximum of 31 character variable names
ENTER	default to maximum of 7 character names

The maximum number of variables for review is limited by the maximum variable name length selected, as shown below.

NAME LENGTH	NUMBER OF VARIABLES TO REVIEW
1	maximum of 128
2-3	maximum of 64
4-7	maximum of 32
8-15	maximum of 16
16-31	maximum of 8

NOTE: The name length includes all characters. The variable name A$(21,5) is considered to have a length of eight (8). F(R(3,8)) is nine characters in length. After successfully entering a variable length, the message 'Input variables' will be displayed and all previously entered variable choices will be erased. This function will allow you to enter variables using the following syntax:

A	XI	B#	QI(F(G,Q))
K$	AI(F(G,Q))	F(2,3)	T#(F,(B(A,N),G(E)))
A(B)	WEEKDAY	S%	A(B,C)

Any number of parentheses are allowed, provided you close them within the variable length entered. Although illegal variable names such as A$3 or A(3H) are not rejected, they will cause errors later when review of the variables is attempted. When you have finished entering the variables, press "BREAK" to continue with the review. If you enter the maximum number of variables allowed, BBASIC will automatically proceed with the review. At this point BBASIC invokes an "@O" as described below.

Pressing "@O" at any time during program execution will immediately save the contents of the video display and replace it with the message 'C" = change, "D" = delete, "I" = insert' will appear along with the first variable and its value. Variables are displayed in the order entered by the "@N" function. Pressing "CLEAR" will cause the message "End of variables." to appear. If you are really finished with the variable review, press "CLEAR" again and your original video display will be returned. Your program will resume execution at that point. If you instead want to review more variables, press any key other than "CLEAR". Pressing "C" will allow you to select another variable in place of the last variable displayed. The new variable selected and its value will be displayed. Remember, the variable name is limited in length per your original choice when "@N" was selected. Pressing "D" will delete the last displayed variable. Pressing "I" will insert a variable PRIOR to the last displayed variable. Pressing any key other than "CLEAR", "C", "D", or "I" will advance the display to the next variable selected. If you attempt to review a variable whose subscript is out of range or with an illegal name, the message 'ERROR, RE-ENTER' will be displayed and the "C" command will automatically be invoked. You must select a valid variable to exit from this sub-command. If you evaluate an element of an array (subscript < 11) and the array has not yet been dimensioned by your program, this array will be dimensioned for eleven elements (0-10). If your program subsequently attempts to dimension this array via the "DIM" instruction, an error will occur. Dimension all used arrays before you review them with the review variables function.

SUPERBASIC

SUPERBASIC enhancements to LEVEL II BASIC.

SHORTHAND

Single keystroke commands:

```
    .  (period)     =  list current line
    ,  (comma)      =  edit current line
    /  (slash)      =  list "BREAK in" line
  up-arrow          =  list previous line
down-arrow          =  list next line
shift-up-arrow      =  list first program line
shift-down-arrow    =  list last program line
```

These single keystroke commands MUST be the first entry after the BASIC prompt, ">", appears.

Single letter commands:

```
L[.]<ENTER>       = list current line
E[.]<ENTER>       = edit current line
D[.]<ENTER>       = delete current line
P<ENTER>          = list page from current line
Pn<ENTER>         = list page from line "n"
C<ENTER>          = continue program execution
R<ENTER>          = run program
R"filespec"       = run filespec
```

SUPERBASIC allows you to use abbreviated commands. To list the current line, type ".", "L<ENTER>", or "L.<ENTER>". To edit the current line, type ",", "E<ENTER>", or "E.<ENTER>". To delete the current line, type "D<ENTER>" or "D.<ENTER>". The "-" can be used in conjunction with the "L" and "D". To list everything from the first program line up to the current line, type "L-.<ENTER>". To list the entire program, type "L-<ENTER>". The symbol "/" will list the last line for which the "BREAK IN LINE ##" message was issued. The default line ##, if no break has occured, is the last line of the program. The up arrow will display the preceding line while the down arrow will display the next line. The shifted up arrow will display the first program line. The shifted down arrow or the shifted down arrow & Z will display the last line.

EXAMPLES:

In each example the following BASIC program is in memory and the line pointer points to line #20.

```
10 For X= 1 to 20
20 PRINT X, X*X, X*X*X
30 NEXT X
40 END
```

(1) L<ENTER>
Line 20 will be displayed.

(2) L-.<ENTER>
Lines 10 and 20 will be listed.

(3) L.-<ENTER>
Lines 20, 30, & 40 will be listed.

&H and &O Hex and octal constants.

```
&[H]dddd
&Odddd
```

This command allows you to use hexadecimal (base 16) or octal (base 8) constants within your program. The "H" is optional in SUPERBASIC.

EXAMPLE:

```
X = &4000              This assigns 16384 to the variable X.
Y = &6000 - &5200      This assigns 2584 to the variable Y.
POKE &FFFC, 4          This puts a 4 in RAM location -3 (65533).
```

CMD"C" Space compression.

CMD"C"<ENTER>

This command will eliminate spaces and linefeeds within the resident program in RAM, at the rate of approximately 8000 bytes per second. This command will not remove spaces or linefeeds which are in quoted text, data statements, or remark statements. This command allows you to remove unnecessary bytes from your program, resulting in more free memory space and faster program execution.

CMD"D" Load and execute DEBUG.

CMD"D"<ENTER>

This command causes DEBUG to be loaded and executed. To return to the point from which you entered DEBUG, type "G<ENTER>".

CMD"E" Disk I/O error.

CMD"E"<ENTER>

This command will cause a brief explanation of the last DOS error code to be displayed.

CMD"K" Zero array.

CMD"K"vv(0[,0...])

vv = Any valid variable name

This command will zero the array vv(dim[,dim...]) where dim refers to the originally dimensioned values of the array. All elements of the array will be set to zero.

EXAMPLE:

```
(1)  100 DIM A%(6,7,4)
      110 More program lines
      ...
      690 More program lines
      700 CMD "K" A%(0,0,0)
      710 More program lines
```

In the example above, A% was dimensioned as a 280 element array. After line 700 is executed, A% is still a 280 element array and each element has the value of zero.

CMD"L" Delete array.

```
CMD"L"vv(0[,0...])
 vv = Any valid variable name
```

This command will delete the array vv(dim[dim...]) and free the memory space for SUPERBASIC use. After using this command you can redimension the array.

EXAMPLE:

```
(1)  100 DIM A%(6,7,4)
      110 More program lines
      ...
      690 More program lines
      700 CMD "L" A%(0,0,0)
      710 More program lines
```

In the example above, A% was dimensioned as a 280 element array. After line 700 is executed, the array A% no longer exists and the amount of free memory is increased.

CMD"M" Move program line.

(1) CMD"M"line1,line2<ENTER>
(2) CMD"M"line1,.<ENTER>
(3) CMD"M".,line2<ENTER>

This command will relocate line1 to line2 deleting line1 from the program text and inserting it as line2. If line2 existed prior to this command, it will be replaced by the new line2. The "." may be used to refer to the current line pointer.

CMD"N" Duplicate a program line.

(1) CMD"N"line1,line2<ENTER>
(2) CMD"N"line1,.<ENTER>
(3) CMD"N".,line2<ENTER>

This command will copy line1 to line2 while leaving line1 intact. If line2 existed prior to this command, it will be replaced by the new line2. The "." may be used to refer to the current line pointer.

CMD"O" Open an additional file buffer.

CMD"O"

This command allocates an additional file buffer. When SUPERBASIC is initialized it allocates space for three file buffers. If you caused file buffers to be allocated at initialization, only that number has been reserved. CMD"O" allows you to allocate additional buffers from the direct mode in BASIC or from within your program. The RAM location indicating the number of open buffers is 521AH (21018D). DON'T POKE!!!

Don't execute this command from within a subroutine or a FOR-NEXT loop.

EXAMPLES:

(1) From the direct mode, the following command is typed:

CMD"O"<ENTER>

This results in one additional file buffer being allocated, with most variables retained. Strings created via READ, direct string assignments, and DEFFN will not be retained.

(2) Using CMD"O" within a program

```
10 DIM A(20), B$(15)
...
200 CMD "O"
210 OPEN "R",1,"TESTFILE/TXT"
```

Line 200 allocates a file buffer which is opened in line 210

(3)
```
100 If X>15 then 300
110 ON ERROR GOTO 900
120 OPEN "E",X,B$
...
300 PRINT "MAXIMUM # FILE BUFFERS OPEN":END
...
900 IF ERR = 104 THEN CMD"O" ELSE PRINT ERR/2+1: END
910 RESUME 120
```

The above error trap will cause additional file buffers to be allocated each time lines 100-120 are executed.

(4) To open five buffers, without allocating additional ones each time the program is run, the following line could be used:

```
15 If PEEK(&521A)<5 THEN CMD"O": GOTO 5
```

CMD"Q" String sort.

(1) CMD"Q",n1,vv$(0)
(2) CMD"Q",n1,vv$(0,0),n2

vv$ = Any valid array variable name
n1 = An integer or integer variable representing the number of elements to be sorted.
n2 = A positive integer or integer variable representing the column number to sort in a two dimensional array.

This new command will provide for a quick sort of a string array. Typically, a 1000 element array will take less than seven seconds to sort.

The sort is performed in accordance with the sign of n1. If n1 is positive, the sort will be in ascending or alphabetical order. If n1 is negative, the sort will be in descending or reverse alphabetical order.

VERSION (1) is used to sort a single dimensioned array. The array will be sorted up to the n1th element, including the 0th element. Version (2) is used to sort a two dimensional array with n2 indicating which column of the array to use as the sort key.

EXAMPLE:

```
10 CLEAR 600: DIM A$(10)
20 FOR I = 1 to 10
30 READ A$(I)
40 NEXT I
50 DATA "WASHINGTON", "OREGON", "CALIFORNIA",
   "NEVADA", "IDAHO", "UTAH", "ARIZONA",
   "MONTANA", "WYOMING", "COLORADO"
60 CMD"Q",I,A$(0)
70 FOR I=1 TO 10
80 PRINT A$(I),
90 NEXT I
```

The printout will be as follows:

```
ARIZONA      CALIFORNIA   COLORADO     IDAHO
MONTANA      NEVADA       OREGON       UTAH
WASHINGTON   WYOMING
```

NOTE the following key points of the EXAMPLE:

1. The 0th element was not used (it is a null string).

2. The value of I in line 60 is actually 11. CMD"Q" will not cause an error if the value of n1 is greater than the first dimension of the array.

3. If only line 60 were changed to: CMD"Q",-1,A$(0)
the printout would be as follows:

WASHINGTON	UTAH	OREGON	NEVADA
MONTANA	IDAHO	COLORADO	CALIFORNIA
ARIZONA			

What happened to "WYOMING"? It's in the 0th element. A$(0)="WYOMING" and A$(10)="".

4. If only line 60 were changed to: CMD"Q",6,A$(0)
the printout would be as follows:

CALIFORNIA	IDAHO	NEVADA	OREGON
UTAH	WASHINGTON	ARIZONA	MONTANA
WYOMING	COLORADO		

Only the elements A$(0) through A$(6) were sorted, leaving A$(7) through A$(10) as loaded from the data statement.

CMD"R" Enable interrupts.

CMD"R"<ENTER>

This command enables the interrupts.

CMD"S" Return to MULTIDOS.

CMD"S"<ENTER>

This command returns you to the MULTIDOS OPERATING SYSTEM.

CMD"T" Disable interrupts.

CMD"T"<ENTER>

This command disables the interrupts. This command MUST precede any cassette operation.

CMD"V" Scalar variables.

CMD"V"<ENTER>

This command will display all assigned scalar variables and string equivalents in the order they were created.

The screen will clear and up to 12 variables will be displayed. Press any key to display an additional variable or <ENTER> to display up to 12. This function is complete when you receive the "READY" prompt. However, you may embed this command inside a BASIC program, which will require user intervention for program continuance.

CMD"X" Transfer to LEVEL II.

CMD"X"<ENTER>

This function will transfer the resident **SUPERBASIC** program to the LEVEL II BASIC environment while retaining memory protection and a printer driver pointer, if different than MULTIDOS's. In addition, it will seed the "DEBUG AREA" (starting at 4060H) with a semi-recovery program to aid in return to the DISK BASIC environment. Refer to "BASIC #" for direction on reentry to **SUPERBASIC** from LEVEL II BASIC.

This command can be very useful if you want to work on a program which will normally be run in a LEVEL II environment. You can develop the program in **SUPERBASIC**, test it in LEVEL II, and return to **SUPERBASIC**.

CMD"uuuuu" Execute a MULTIDOS function from **SUPERBASIC**.

CMD"uuuuu"<ENTER>

uuuuu = Any valid MULTIDOS command

This command allows you to use any valid MULTIDOS command, including BASIC, from within the **SUPERBASIC** environment. The command can be used in the direct mode or as a statement within your BASIC program. Your program will be protected while the command is being processed. You can even use complex MULTIDOS commands, such as loading and executing machine language programs provided those programs use TOPMEM (4049H) as the upper memory limits and will fit into the remaining RAM. This command requires a minimum of 6330 free bytes to execute.

EXAMPLES:

(1) CMD"DIR"
The directory contents will be displayed.

(2) CMD"SAUCERS/CMD"
The machine language program "SAUCERS/CMD" will be loaded and executed. If the program does not use protected memory, and if it exits via location 402DH then your BASIC program will resume execution at the next line number.

(3) CMD"EA/CMD"
The Editor/assembler will be loaded and executed. Upon completion, your BASIC program will resume execution at the next line number.

DEF FN Define function.

DEFFNxxx(uuu[,uuu...])=www

xxx = Name of the function and is any valid variable name.

uuu = Variables used by the expression.

www = An expression or formula usually involving the variable(s) (uuu) passed on the left side of the equal sign.

This statement lets you create your own implicit function. After a function has been defined, you use the function as any of the other intrinsic functions, e.g., ATN, COS, ASC, etc.

The type of value returned will be the same as the type of variable used to name the function. In addition, the variables used in the DEFFN statement, have no effect on the value of that variable.

EXAMPLE:

```
10 DEF FN Q(K,L) = K/20 + L/10
20 INPUT "Enter quantity of nickels and dimes";N,D
30 PRINT "The amount in dollars is";FN Q (D,N)
```

The function Q (FN Q) is defined using K and L, but the variables in line 20 and 30 are N and D. K and L may be used in the program and have no effect on FN Q, nor does FN Q definition using K and L have any effect on the variables K and L. The space between DEF and FN is optional, DEFFN is acceptable syntax.

DEFUSR Define entry address of USR routine.

DEFUSRn=aaaaa

n = The digit 0-9. If n is omitted 0 is used.

aaaaa = The entry address to a machine language routine. aaaaa may be any numerical expression, including constants, variables, or functions.

EXAMPLE:

```
10 CLEAR 500: DEFINT A-Z
20 N = 8000
30 DEFUSR 5 = N * 4
```

Defines 32000 decimal to the USR 5 call.

INSTR String search.

INSTR([p]string,substring)

p = Position in the string where the search is to begin.
string = The name of the string to be searched.
substring = (1) The name of the substring for which you are searching, or
(2) The actual substring for which you are searching

This function searches through "string" to see if it contains "substring". If it contains "substring", INSTR returns the starting position of "substring" in "string"; otherwise zero is returned.
If "substring" is a null string, INSTR returns zero.

EXAMPLES (let Z$="SUPERBASIC", W$="", X$="SUPER")

Expression	Result
INSTR (Z$,"PER")	3
INSTR (Z$,"TRS")	0
INSTR (2,Z$,W$)	0
INSTR (3,Z$,"U")	0
INSTR (Z$,X$)	1
INSTR (2,Z$,X$)	0
INSTR (4,Z$,"BASIC")	6
INSTR (3,"ABCDABCDABCDABCD","ABC")	5

LINEINPUT Input a string from keyboard.

LINEINPUT["message";]vv$

message = A prompting message
vv$ = A valid variable name

This BASIC statement allows you to input a complete line from the keyboard, including punctuation, and line feeds. A space is permitted between LINE and INPUT (LINE INPUT). This statement nulls the variable, does not print a question mark, allows the entry of only one string variable, and recognizes leading spaces.

LIST Display program text.

LIST<ENTER>

This command has been modified to show graphic characters included in quoted text. You will not get a string of BASIC reserved words. The actual graphics will be displayed. Editing of these packed strings is allowed, as long as the "A" command is not used. (If you accidently enter an "A" command, use the "Q" command to abort it.)

MID$= Replace portion of a string.

MID$(vv$,p[,c])=rr$

vv$ = The variable string to be changed.
p = The starting position within the string for the replacement.
c = An optional parameter indicating the number of characters to be replaced.
rr$ = The replacement string.

This command lets you change part of a string. The length of the target string (vv$) is not changed by the MID$ = statement. The excess characters to the right of (rr$) would be ignored if the replacement string is too long.

EXAMPLES (let C$="12345678", D$="BASIC")

Expression	Resultant C$
MID$(C$,3,4)="ABCDE"	12ABCD78
MID$(C$,1,2)=D$	BA345678
MID$(C$,5)="YZ"	1234YZ78

TIME$ Get current RAM date and time.

This is a BASIC function which returns, in a 17 byte string, the time and date in the form MM/DD/YY HH:MM:SS

USRn Execute a machine code routine.

USR[n](val)

n = A number from 0 - 9. The default value is 0.
val = An integer or integer variable with value from -32768 to +32767.

This command transfers control to a machine language subroutine which was previously defined with the DEFUSRn statement.

When a USR function is encounterred in a statement, SUPERBASIC transfers the program counter (PC register) to the address specified by the corresponding DEFUSR statement. When the specified USR function is complete, via a RET or JP 0A9A instruction, the BASIC program will continue at the next statement following the USR function.

To pass a value to the USR function, execute a CALL 0A7F as the first instruction in the USR routine. This call will transfer the value of "val" to the HL register pair. To receive a value from the USR subroutine, place the value into the HL register pair, then exit via JP 0A9A. The "val" variable will contain this value when SUPERBASIC continues.

The last USR subroutine will have the LSD entry point in 408EH - 16526 dec and the MSD entry point in 408F - 16527 dec. The USR function in SUPERBASIC, places the address here, then returns to ROM to execute the subroutine. You can circumvent the extended USR function by POKING a C9H 201 dec into RAM location 41A9H - 16809 dec. (This is the value for LEVEL II but has a C3H - 295 dec for the extended USR function). Next poke your subroutine's address into 16526 dec and 16527 dec to execute a LEVEL II program with USR calls in SUPERBASIC. Add the instruction: POKE 16809, 201 prior to executing the USR subroutine.

SUPERBASIC overlay utilities.

FIND Find ASCII characters.

Ftar<ENTER>

This command will find all occurences of "tar" in your BASIC program. Spaces are recognized with this command.

EXAMPLE:
F in <ENTER>

This will FIND all occurences of " in " in the resident BASIC program. The display will indicate the line the "tar" is found on, and if more than one occurences are on this line a "/" is printed followed by the number of occurences.

EXAMPLE:
F:<ENTER>

10 20/5 50 70 90/3

The ":", colon, character is in line 10 once, line 20 five times, line 50 once, line 70 once, and in line 90 three times.

GLOBAL EDITING (GE) - Mass editing BASIC program.

-<ENTER>

This SUPERBASIC feature will permit you to perform surgery on your BASIC program. You can change variable names, items in a data list, integers, strings, etc. You can create compressed strings, merge lines, split lines, change reserved words, and more.

The following types of operations are allowed:

- Change all or part of variable names.
- Change all or part of constants, data list items, or strings.
- Change graphic codes as "CHR$(x)" into packed strings (x = 128-191).
- Change space compression codes as "CHR$(y)" into packed strings (y = 192-255).

- Merging of adjacent line numbers into one long line.
- Splitting of one long line into two shorter ones.
- Change reserved words.

GE - General changes:

To use the utility:
(1a) From the MULTIDOS ready mode:
Load SUPERBASIC and your program. e.g. BASIC LOAD"program"<ENTER>

(1b) From SUPERBASIC:
Load your BASIC program. e.g. LOAD"program"<ENTER>

(2) Enter the global editor by issuing the command: "-<ENTER>"

The screen will be cleared and the following message will be displayed:

Line Target

T =

This is the target (T) entry mode. Enter a target as follows:

To change a constant, a variable name, or item in a data list which is not enclosed in quotes, respond with the item when the target is requested. The response must be from 1 to 255 characters long, and must be terminated with the <ENTER>.

To change a target which is enclosed in quotation marks, precede the target with the $ sign. The target must be from 1 to 254 characters and must be terminated with the <ENTER>.

To change only a single character variable, while leaving multi character variables with the same letter unchanged, enclose the target single character within single quotes ('X').

To change only the first character of a multi character variable, enter the target character followed by a single quote (X'). To change the second or greater character of a multi character variable, while leaving the first character unchanged, precede the target single character with a single quote ('X).

GE - Changes to reserved words:

The global editor will allow you to change reserved words, such as "PRINT" to "LPRINT". You must be careful, however, as lowercase is not acceptable. Under SUPERBASIC or LEVEL II BASIC, as each line is entered or edited, it is processed through a buffer. This buffer looks for reserved words and changes them to a one byte code. It also converts all variables to uppercase. Since the global editor does not process changes through this buffer, do not use lowercase for reserved words or variables as a syntax error will occur at run time. If your target is a reserved word or the arithmetic operators +, -, *, /, up arrow, >, =, or <, bracket this target with the "< & ">" characters. i.e. <+>.

After you have entered a target according to the rules above, you will be prompted to enter Line A, which is the first line to search and has a default value of your first program line. You will next be prompted to enter Line B, which is the last line to be searched and has a default value of the last line of your program. Line A and Line B permit you to limit a change to a specific range of program lines.

Next you will be prompted to enter the replacement for the target specified. To re-enter the target if you entered it in error, use the <ENTER> character as the replacement. You will be returned to the target entry mode. To delete all occurences of a target, use the <SHIFT>@ as the replacement.

Finally, you will be asked if you want the above global changes to be made. This is your last chance to correct an error in your entries. Respond "Y" to the query "Use (Y/N)?" if you want the changes made. SUPERBASIC will then search your program for the target and make the changes. The screen will show the line number being searched under the title "Line" and the last line number where a target was found under the title "Target". SUPERBASIC will display a total of the target occurences to indicate completion.

To exit from the GLOBAL EDITOR press <ENTER> after a successful change or press <BREAK> at the target or replacement queries.

EXAMPLES: (> means changes to, T= is the target, and R= is the replacement)

(1) To change all occurences of the variable "B" to "F", T = B and R = F.

B>F, AB>AF, BA>FA, BB>FF, BC>FC, BD>FD, B$>F$, AB$>AF$, A$(AB)>A$(AF)

(2) To change all occurences of a single "A" to "G", T = 'A' and R = G.

A>G, A$>G$, A$(1)>G$(1), A$(A)>G$(G), A$(AA)>G$(AA)

(3) To change all occurences of the first "A" in a multi character variable to "H", T = A' and R = H.

AA>HA, AB>HB, AC>HC, AD>HD, AE>HE, AA$>HA$, AB$>HB$, A$(AA)>A$(HA), A$(AB)>A$(HB)

(4) To change all occurences of the second "A" in a multi character variable to "I", without changing the first occurence, T = 'A and R = I.

AA>AI, BA>BI, AA$>AI$, A$(AA)>A$(AI)

(5) To change all "E" in strings to "Z", T = $E, and R = Z.

"ABCDE">"ABCDZ"

GE – Building compressed strings:

If your program contains many "CHR$(x)+CHR$(x)" statements, the global editor will shorten it by building a compressed string. This can result in faster execution and more free memory. As an example, the line:

```
10 A$=CHR$(191)+CHR$(129)+"X"+CHR$(176)
```

would require 32 bytes of memory. After the global editor built a compressed string, the line would be:

```
10 A$=".."X."
```

where the "." represents a graphic character. The new length would be 14 bytes , or more than a 50% saving of space.

To build a compressed graphics string in place of CHR$(x) type lines, enter the "+" character for the target.

To build a compressed string with space compression codes, enter the "*" character for the target.

The CHR$(x) OR CHR$(y) cannot contain blanks within the parentheses.
CHR$(191) is not acceptable and will not be changed.

NOTE: The changed code will display properly. SUPERBASIC will also list it properly to the screen. However, since it is compressed code or graphics, it will not look the same when it is sent to a printer which does not have graphics capabilities.

EXAMPLES:

(1) "+" Changes 210 B$=CHR$(131)+"X"+CHR$(176)
to 210 B$=".X." Where "."=graphics.

(2) "*" Changes 337 PRINT CHR$(204)+"TEST"
to 337 PRINT " TEST"

(3) "+" Followed by the "*"
Changes 400 A$=CHR$(178)+CHR$(204)+CHR$(190)
to 400 A$=". ."

GE – Merging line numbers:

The global editor will allow you to merge or append a program line to the preceding line in your program. It does not change references to the line. As an example, if your program contained lines 1,2,3,4, & 5, and you merged line 3 to line 2, then an error would result at run time if line 5 originally contained "GOTO3". This is because line 3 no longer exists. Do not append to a line which contains an open quote (10 A$="THIS IS). Close the quote first, then merge (10 A$="THIS IS").

This function will allow you to create lines greater than 255 bytes in length. The lines will execute properly, but can neither be properly listed on the screen nor edited.

To use this merging function, respond with the "/" character and the line number to be merged as the target.

EXAMPLE:

```
10 A$="TEST #"
20 PRINT A$
```

To merge the above two lines target is /20.
The result is one line as follows:

```
10 A$="TEST #":PRINT A$
```

GE – Splitting lines:

The global editor will allow you to split a program line containing two or more BASIC instructions into two lines. The new line number can have any number greater than the line to be split and up to 65529. However, if you assign a new line number greater than the next line after the line to be split, run time errors can occur if you make references to those in-between lines. As an example, if your program contains lines 10, 20, 30, 40, 50, & 60, and you split line 20 into lines 20 and 45, lines 30 and 40 will not be be found by any reference instructions such as "GOTO 30", "GOSUB 40", etc. However, if the program flow is such that BASIC would normally process the next statement in RAM after the new lines 20 and 45, lines 30 and 40 will be processed. BASIC would, in the absence of any branching instructions, process lines 20, 45, 30, 40, and 50 in that order.

The split point must be directly behind a colon (:) in the program line. To use the split function, the target is in the form –ttt where ttt is the target for the split. If the target is a reserved word such as "PRINT", enclose the target with "<" AND ">" signs. If no target is specified, the line will be split at the first colon. Line A now represents the line number to be split and line B represents the new line number.

EXAMPLES:

(1) 10 A$="TEST #":PRINT A$"

To split the above line, T = –, Line A = 10, Line B=15.

The result is:

```
10 A$="TEST #"
15 PRINT A$
```

(2) 30 A$="ABCDE":PRINT A$:B$="FGH":PRINT B$

To split the above line at ":PRINT B$",
T = –<PRINT> B$, Line A = 30, AND Line B = 35.
The result is:

```
30 A$="ABCDE":PRINT A$: B$="FGH"
35 PRINT B$
```

The global editor executes very rapidly. The creation of compressed strings requires 67 iterations each time, so the "*" and "+" functions will operate at a slower speed. During operation, the editor will display the line being searched at the top of the screen. It will also display the line in which the target was last found.

Be cautious about changing variables in a "DEF" statement. The range must be ascending. If the replacement is larger than the target, your program will be increased in size by that difference for each occurence. If you run out of memory ,an "Out of MEM" error message will be displayed. At this time all changes up to the point where memory was insufficient will be made. If this occurs there is still sufficient memory to reverse the changes or to make other changes which would decrease the program size.

REFERENCE Cross reference variables and integers to 9999999.

;[p][xxx]<ENTER>

p = "*" if the reference listing is to be displayed on the screen
or "$" if the reference listing is to go to the printer also.
NOTE: p is optional.

xxx = Reference target which may be:
(1) A one or two character variable name without a type suffix.
(2) An integer number which may be a line number or a value used in the program.
(3) A reserved word if preceded by a # symbol.

Target version (1) will show all line numbers which contain the variable specified. Target version (2) will show all line numbers which refer to the integer specified. The reference may be as an integer in the line or to a line number. Target version (3) will show all line numbers which contain the reserved word.

After a reference target has been made, you may display those lines by pressing the ";" key. Each line referenced will be displayed sequentially and may be edited before the next line is called.

Use of the p option without a xxx target will result in a reference listing of all integer numbers and variables. If the p option is used with a xxx target, a reference listing will be produced starting with the target and proceeding in ascending order. If the reference listing is requested in the form ";*#<ENTER>", a listing of reserved words will be produced. The reserved word listing is produced in the order ROM-BASIC "TOKENIZES" (converts to compressed storage) the reserved words. This is not alphabetical order.

To pause during a reference listing, press shift @. To resume the listing, press any key. To abort the reference listing, <BREAK>.

EXAMPLES:

(1) ;*<ENTER>

All integer and variable references are displayed on the screen.

(2) ;K<ENTER>

All references to the variable "K" are displayed on the screen.

(3) ;$G<ENTER>

All variable starting with "G" and continuing through "ZZ" will have their references directed to the line printer.

(4) ;#PRINT<ENTER>

All line numbers which contain the reserved word "PRINT" will be displayed on the screen.

(5) ;$#<ENTER>

All reserved words will have their references directed to the line printer.

When a reference is displayed or printed for any target, the line number containing the reference is displayed and may be followed by one or more of the following modifiers:

/n where n = the number of references to the target within that line.
/$n the variable contains the string designator "$".
/%n the variable contains the integer designator "%".
/!n the variable contains the single-precision designator "!".
/#n the variable contains the double-precision designator "#".
(the variable is used as an array variable in this line.

RENUMBER Renumber a BASIC program.

(1) :<ENTER>
(2) :o[nn][,iii][,sss][,eee]<ENTER>

This function will allow you to check for missing or invalid line numbers within your program, recover a "NEWED" program, and renumber all or part of your program.

To check for errors use format (1). **SUPERBASIC** will scan your program for reference operators (GOTO's, GOSUB's, ON's, ERL's, etc.). It will look for missing line numbers, and overflow or improper line numbers.

When format (1) is used, the message 'ONLY CHECKING FOR ERRORS' will appear on the screen. If any errors are found, the message 'Error(s)' will appear, followed by the line number causing the error and the error type. If no errors are found the message 'Function completed' will be displayed.

Error types are:

!!!!!/U(nnnnn)	Line number !!!!! is referenced in line number nnnnn but does not exist.
!!!!!/S	Line number !!!!! contains a syntax error.
!!!!!/O	Line number !!!!! contains an improper line number (>65529).

To recover a program immediately after you have typed "NEW", use FORMAT (1). This will force a rescue of the program. If you establish a variable between the "NEW" and the ":", the results are unpredictable, and probably will not be what you wanted. To renumber your program, or parts thereof, use format (2) as shown above where:

nnn = The first line number to be assigned to a renumbered line. nnn must be in the range of 0 To 65529. The default value is 10.
iii = The increment to be used in renumbering. The default value is 10. If used, iii must be preceded by a comma.
sss = The starting point in the original program where renumbering is to occur. sss must be <= nnn and has a default value of 0.
eee = The ending point in the original program where line renumbering is to stop. eee must be >= sss and has a default value of 65529.

SUPERBASIC will check your program for errors before attempting to renumber. If any errors are found they will be displayed and control will be returned to you with your program unchanged.

EXAMPLES:

Sample program

```
10 PRINT "TEST"
20 GOTO 290
30 INPUT A
40 ON A GOTO 10,20,,60,70
50 GOTO 10
60 PRINT A
70 PRINT A*2
80 GOTO 70000
90 END
```

(1) The command ":<ENTER>" would generate the following messages:

```
Only checking for errors

 Error(s)

 290/U(20)  40/S  80/O

 Function completed

 READY
 >
```

(2) After fixing the above errors, the command ":50,1,50,70<ENTER>" would generate the following renumbered program:

```
10 PRINT "TEST"
20 GOTO 90
30 INPUT A
40 ON A GOTO 10,20,51,70
50 GOTO 10
51 PRINT A
70 PRINT A*2
80 GOTO 10
90 END
```

SUPERBASIC file manipulation.

KILL Delete a file from a diskette.

KILLvar$ or KILL"filespec"

var$ = a string defined as a filespec.
filespec = a file specification for an existing file.

This command allows you to delete a file from the directory. If the statement is within the program, you must be sure to close the file first. From the command mode, SUPERBASIC and MULTIDOS close all files before the directory is changed.

LOAD Load a BASIC program to RAM.

LOADvar$[,R] or LOAD"filespec"[,R]

var$ = a string defined as a filespec.
filespec = a valid BASIC program file name.

This command allows you to load a BASIC program from diskette. The ",R" option will load and run the program without closing any previously open files, which were opened via an OPEN statement.

MERGE Combine two programs in RAM.

MERGEvar$ or MERGE"filespec"

var$ = a string defined as a filespec.
filespec = a BASIC program saved using the ASCII option.

This command combines two program segments within SUPERBASIC's memory space. The program lines in var$ will be inserted into the resident program in sequential order. If line numbers in var$ coincide with a line number in the resident program, the resident line will be deleted.

NAME Load an execute a program keeping variable values.

NAMEvar$[,R] or NAME"filespec"[,R]

This command is SUPERBASIC's chaining function. It allows you to run BASIC programs which are too large to fit in memory. This command will cause SUPERBASIC to load and execute the file var$ or filespec beginning with the lowest line number as if it were a new BASIC program. The previous variables will remain intact, except for DEFFN, strings created via READ, or string assignments directly in a BASIC program. If the ",R" option is specified, the files will remain open.

RUN Load a BASIC program and execute.

RUNvar$[,R] or RUN"filespec"[,R]

This command allows you to load a BASIC program from diskette and immediately execute the program at the lowest program line.

SAVE Save BASIC program onto diskette.

SAVEvar$[,A] or SAVE"filespec"[,A]

This command allows you to transfer the current BASIC program onto the diskette in compressed format. The "A" option will cause the program to be stored in ASCII format. Compressed programs load faster than ASCII programs but cannot be intelligently listed from MULTIDOS.

SUPERBASIC – File Access.

OPEN Sets the mode and assigns filebuffer to a filespec.

(1) OPENmode,buf,var$

mode is a string or constant, and is one of the following:

mode	access mode
D	RANDOM I/O to an existing file.
E	SEQUENTIAL OUTPUT to an existing file.
I	SEQUENTIAL INPUT from an existing file.
O	SEQUENTIAL OUTPUT to a file.
R	RANDOM I/O to a file.

buf = equivalent to the file buffer which will be assigned to "var$". buf value MUST be 1 to "ff", whereas "ff" is the number of file buffers opened during SUPERBASIC initialization.

var$ = a string defined as a filespec.

(2) OPENmode,buf,var$,udl

mode is "R", and udl = user defined record length.
(if mode = "D", udl is ignored, but no error is generated.)

EXAMPLES: let N = 2, Q$ = "IOTA", P$ = "CHECKING/TXT".

OPENQ$,N,P$

Opens the file "CHECKING/BAS" for sequential input into file buffer 2 (The file MUST exist).

OPEN"O",3,"BALANCE/TXT"

Opens the file "BALANCE/TXT", sets the pointer to the beginning of the file, and assigns file buffer 3 to the file.

When the "O" mode is used, if the filespec does not exist, it will be created with the pointer (naturally) set to the beginning of the file. Nevertheless, if the file existed or not, the "O" mode will set the pointer to the beginning of the file; whereas the "E" mode will set the pointer to the end of file, "EOF". If the file does not exist in OPEN"E", the "EOF" will be the beginning of the file!

Only one file buffer may be assigned to the "D", "E", "O", and "R" modes, whereas the "I" mode may have two or more. I addition, a file buffer may not be assigned to more than one filespec at a time.

CLOSE Unassign a file buffer by buffer number.

CLOSE[#][buf[,buf...]]

buf = an expression with a value of 1 to 15. If buf is omited, all open file buffers will be closed.

EXAMPLE:

CLOSE # 3	Closes file buffer 3.
CLOSE 8,2,4	Closes file buffers 2, 4, and 8.
CLOSE T	Close file buffer equivalent to the value of "T".

Anything which generates a CLEAR statement directly or as a subroutine will close all files. The following will close all files: NEW, LOAD/RUN (without ",R"), MERGE, EDITing a program line, and CLEAR.

INPUT# OPEN"I" read command.

INPUT#buf,var[,var...]

buf = file buffer 1 to 15.
var = a variable name to contain the data from the file.

LINEINPUT# OPEN"I" read a string to 0DH (13 dec).

LINEINPUT#buf,var$

buf = file buffer 1 to 15.
var$ = the variable name to contain the string.

LINEINPUT# will read all characters from the current position up to and including a 0DH (not preceded with an 0AH), up to the end of file, or 255 characters - whichever comes first.

PRINT# OPEN"O" and OPEN"E" write command.

PRINT#buf[USINGformat$;]item[m item...]

buf = a file buffer 1 to 15.
format = a sequence of field specifiers used with USING.
m = a delimeter placed between "items".
item = an expression to be evaluated and written to the diskette.

The "item" delimeter, "m", can be a semicolon ";" or comma ",". The use of these delimeters will determine the format on the diskette. If the comma is used, the "items" will be zoned in 16 byte areas on the diskette. This consumes diskette space rapidly.

FIELD OPEN"D" and OPEN"R" file buffer organizer.

FIELD[#]buf,len1 AS str1$[,len2 AS str2$...]

buf = a file buffer 1 to 15.
len1 = the length of the first field.
str$1 = the variable name of the first field.
len2 = the length of the second field.
str$2 = the variable name of the second field.
... = subsequent len AS str$ pairs for the balance of the buffer size. The size is determined by the user for created files or by the file itself for existing files.

All of the data items for RANDOM I/O is defined as strings. To convert numeric data to a string, the following functions are used:

MKI$(num) num is an INTEGER number.
MKS$(num) num is a SINGLE PRECISION number.
MKD$(num) num is a DOUBLE PRECISION number.

The length of the string is determined by the precision of the convert function. MKI$ creates a 2 byte string, MKS$ creates a 4 byte string, and MKD$ creates an 8 byte string.

In order to convert the string back to a number, the following functions are used:

CVI(str$)	str$ is a 2 or greater byte string.
CVS(str$)	str$ is a 4 or greater byte string.
CVD(str$)	str$ is a 8 or greater byte string.

Procedure to write to a RANDOM file.

Once a RANDOM buffer has been FIELDed, and all necessary numbers are converted to a string, the data must be placed into the file buffer. The commands LSET and RSET are used to place the "str$" used in the FIELD statement into the assigned file buffer.

LSETstr$ = exp$
RSETstr$ = exp$

str$ = a FIELDed variable name.
exp$ = the assignment for str$.

The command LSET will left justify the "exp$" into the RAM space pointed to by "str$". The RSET command will right justify "exp$" into the RAM space pointed to by "str$".

NOTE: If "str$" has not been assigned to a RANDOM file buffer, then LSET or RSET will STILL reassign "exp$" to this string. I just gave you a little rope here!!!

LSET and RSET will not increase the length of "str$". If "exp$" is longer than "str$", the characters to the RIGHT are truncated.

PUT OPEN"D" and OPEN"R" write statement.

PUTbuf[,rec]

buf = a file buffer 1 to 15.
rec = the record number. If rec is omitted, the current record is used. The current record is one unit greater than the last record written.

The PUT statement will write to a file if the record length is 256 bytes. If the user has defined the record length less than 256
i.e. OPEN"R",1,"POKER/TXT",67 then MULTIDOS will wait until the I/O buffer is full before a write is executed.

To recap a RANDOM write to a file, the following steps are necessary:
(1) OPEN the file -- "R" or "D".
(2) FIELD the buffer.
(3) MKI$, MKS$, MKD$ as necessary.
(4) LSET or RSET all data.
(5) PUT the record.

Procedure to read from a RANDOM file.

In order to read from a RANDOM file, the file must be OPENed first, then FIELDed per your requirements. The GET command will read a record into the file buffer. The "str$" in the FIELD statement will now have the equivalent of the contents in record read.

GET OPEN"D" and OPEN"R" record getter.

GET[#]buf[rec]

buf = file buffer 1 to 15.
rec = the record number. If rec is omitted, the current record is used. The current record is one unit greater than the last record read.

File position indicators.

SUPERBASIC has three file position indicators which can be used with OPEN"D", OPEN"I", and OPEN"R".

EOF End of file detector.

EOF(buf)

This function returns a zero if the end of file has not been read. Otherwise a -1 is returned.

LOC File location indicator.

LOC(buf)

This function returns one unit higher than the current record read for OPEN"D" and OPEN"R". For OPEN"I", LOC returns one unit higher than the last sector read (physical records).

LOF Last record indicator.

LOF(buf)

This function returns the highest record for OPEN"D" and OPEN"R". For OPEN"I", LOF returns the the highest physical record (sector).