MULTIDOS

COSMOPOLITAN ELECTRONICS CORPORATION
P O BOX 89
PLYMOUTH, MI 48170

MULTIDOS

Cosmopolitan Electronics Corporation
5700 Plymouth Road
Ann Arbor, Michigan 48105

MULTIDOS

Foreword

I would like to thank several people who have encouraged me to develop 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. Larry Ashmun told me how to get the MODEL III disk I/O right.

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

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.

TRS-80 (tm) is a registered trademark of Radio Shack, Inc.
NEWDOS (tm) is a registered trademark of Apparat, Inc.
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.

Cosmopolitan Electronics Corporation
5700 Plymouth Road
Ann Arbor, Michigan 48105
(313) 668-6660

CONTENTS

LIBRARY COMMANDS

SYSTEM UTILITIES

SUPERBASIC

SUPERBASIC OVERLAYS

SUPERBASIC FILE MANIPULATION

SUPERBASIC FILE ACCESS

TECHNICAL

BACKING UP MULTIDOS

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.

(Some of these directions do not apply to the MODEL III)

1. Turn on your expansion interface (MODEL I only).
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.

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 <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 (each having the same number of tracks), otherwise answer with 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–MODEL I, and 40–MODEL III. The density will be the one you ordered for the MODEL I, and double for the MODEL III.

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

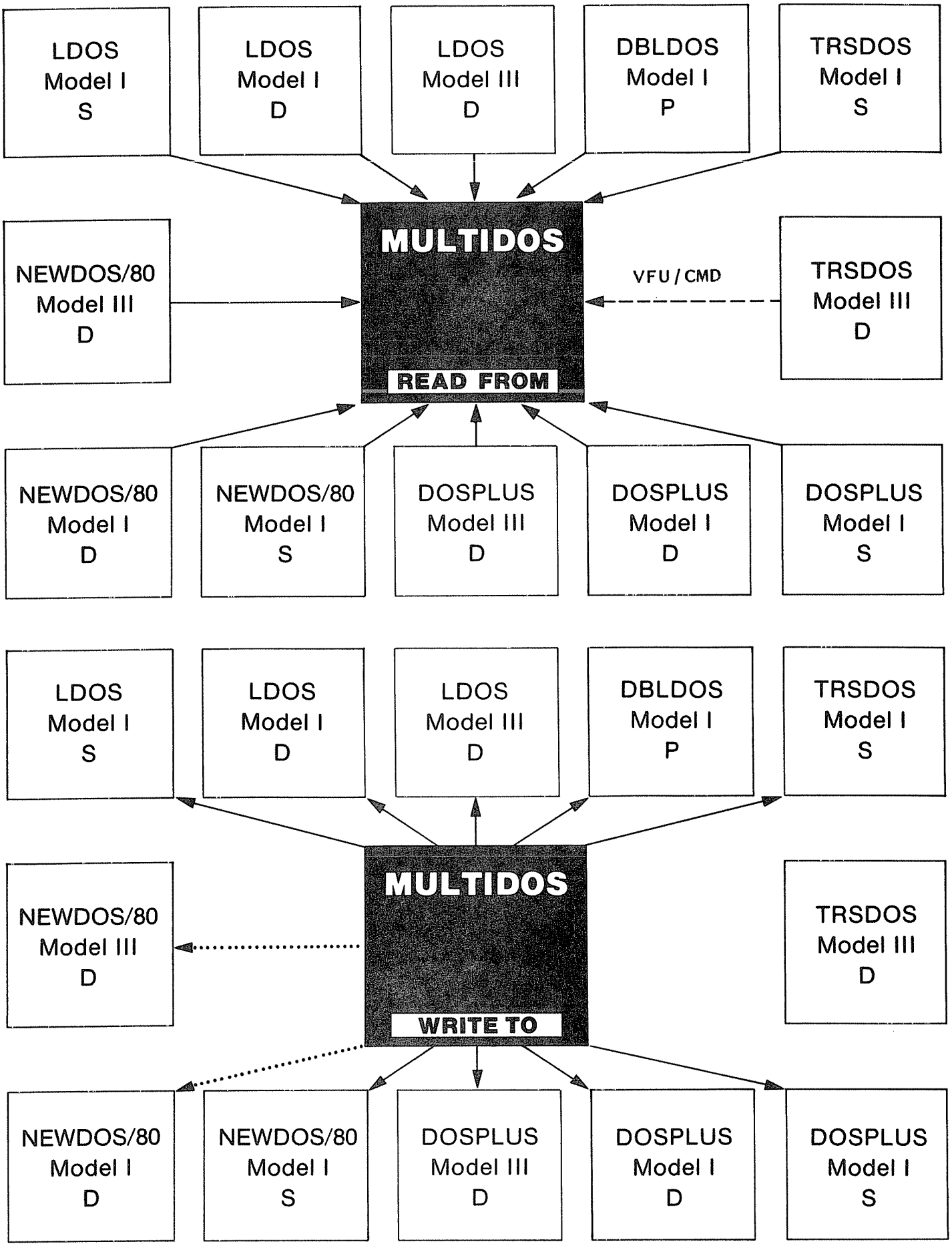
REGISTRATION

Please fill out and mail the registration card when you receive MULTIDOS. As a registered user, this will enable us to inform you, at our option, of any revisions or updates of MULTIDOS. In correspondence with Cosmopolitan Electronics Corporation please state your name, address, registration number, and computer type (Model I or Model III).

ACKNOWLEDGEMENT

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

Vernon B. Hester



S = Single Density **D = Double Density** **P = P Density (Modified Double Density)**

What is MULTIDOS

This operating system is the underlying software required to make the users programs and the hardware work together. MULTIDOS was written with the intent of being user-oriented software without being superfluous.

The word "format", below, is used to describe the particular systems' logic in communicating with its diskette.

Since we are not experts on other operating systems, we can only tell you what MULTIDOS is capable of doing. To best describe MULTIDOS's interface capabilites the diagram shown below 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	CONV	CONV
I	NEWDOS	SINGLE	A	OK	NOTE 1	CONV	CONV
I	VTOS	SINGLE	A	OK	NOTE 1	CONV	CONV
I	ULTRADOS	SINGLE	A	OK	NOTE 1	CONV	CONV
I	NEWDOS80-1	SINGLE	A	OK	NOTE 1	CONV	CONV
I	DOSPLUS	SINGLE	A	OK	OK	CONV	CONV
I	NEWDOS80-2	SINGLE	A	OK	OK	CONV	CONV
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	NEWDOS80-1	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	NEWDOS80-2	DOUBLE	E	OK	NOTE 2	OK	NOTE 2
III	TRSDOS	DOUBLE	F	VFU	NO	VFU	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	NEWDOS80-2	DOUBLE	H	NO	NO	OK	NOTE 2

CONV = Convert. VFU = Versatile File Utility
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 available via NEWDOS/80. MULTIDOS does not report correct free space (except CAT/CMD).

MULTIDOS MANUAL NOTATION

The notation in the list below is used throughout the MULTIDOS manual.

Symbol	Meaning
<ENTER>	Press the <ENTER> key.
<BREAK>	Press the <BREAK> key.
<SPACE>	Press the <SPACE-BAR>.
<COMMA>	Press the "," key.
<LF>	A linefeed character, the down-arrow.
filespec	A valid MULTIDOS file.
drivespec or d	A particular disk drive number (0,1,2, or 3) For two-side 0', 1', 2', or 3' for opposite side.
[]	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

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 (A–Z, or 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. For two-side operation - 0', 1', 2', or 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

The following are filename extensions generally employed by micro computer users.

/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: BACKUP<ENTER>

MULTIDOS will add the extension /CMD to BACKUP, then load and execute the file BACKUP/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 they are separated by a comma and with no spaces.

EXAMPLE: FREE,DIR 1,CLOCK,BASIC<ENTER>

1. Displays the free granules and free file spaces on all mounted diskettes.
2. Displays the directory on drive one.
3. Displays the real time clock.
4. Loads and executes SUPERBASIC.

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

MIGHTY MULTI will be activated if the following conditions are met:

1. Interrupt service is active.
2. Simultaneous depression of the : and ; keys

To exit MIGHTY MULTI:

1. Press <BREAK> or
2. Press "!" as the first input character.

However, upon exit an extraneous character may be present.

Commands for the various functions are single letters.

Ia - Mighty Multi COPY - Copies files, <C>

C fileone[:d] [to] filetwo[:d]

Please note the first blank is optional. Although the drive numbers are optional, both the filenames are mandatory.

Ib – Mighty Multi DIRECTORY – Display diskette directory, <D>

D:n or Dn

n = 0, 1, 2, 3 for the drive number
n = 0', 1', 2', 3' for the opposite side of a double headed drive.

Ic – Mighty Multi KILL – Delete file, <K>

K filespec

Id – Mighty Multi LIST – List a file to the display, <L>

L filespec

The listing can be temporarily suspended if the <SPACE–BAR> is held down.

II – <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 system will not "HANG" if the printer is not ready (MOD I, MOD III must press the <BREAK> key).

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>.

II – <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.

This command may produce erractic results if your printer is not able to print TRS–80 graphic characters.

MULTIDOS SYSTEM DISKETTES

MULTIDOS requires that certain system (/SYS) files be present on a MULTIDOS diskette, in drive 0, for proper operation. More than a minimal system is required for operation of certain commands and functions.

A MULTIDOS minimum system diskette contains the following files:

DOS/SYS - 3 grans - 4300H to 4CFFH.
DOS0/SYS - 1 gran - 4E00H to 50FFH.
DOS1/SYS - 1 gran - 4D00H to 51FFH.
DOS2/SYS - 1 gran - 4E00H to 51FFH.
DOS3/SYS - 1 gran - 4E00H to 50FFH.

Other MULTIDOS system files which are required to execute certain commands and functions are:

DOS4/SYS - 1 gran - 4D18H to 51FFH. Contains DOS error messages.
DOS5/SYS - 1 gran - 4D00H to 51FFH. This is the file for DEBUG.
DOS6/SYS - 4 grans - 5200H to 67FFH. Library executor. Contains code for APPEND, AUTO, BUILD, CONFIG, DEVICE, DIR, DUMP, FREE, KEYBRD, LINK, LIST, PRINT, ROUTE, SKIP.
DOS8/SYS - 4 grans - 5200H to 67FFH. Library executor for MULTIDOS library commands: ATTRIB, CLRDSK, DATE, DDAM (MODEL I), DO, FORMS, HASH, KILL, PATCH, PROT, RENAME, RESTOR, SETCOM (MODEL III), TIME, and TOPMEM.
DOS9/SYS - 5 grans (MODEL I) - 4 grans (MODEL III) - 5200H to 67FFH. Library executor for HELP.

EDIT/SYS - 1 gran - 4D00H to 51FFH. Performs BASIC global editing.
RENUM/SYS - 1 gran - 4D00H TO 51FFH. Performs BASIC renumbering.
CREF/SYS - 1 gran - 4D00H to 51FFH. Performs BASIC cross reference.
ERROR/SYS - 1 gran - 4D00H to 51FFH. Displays BASIC error messages.

MULTIDOS requires a MULTIDOS system diskette to be in drive 0 to load a /CMD file from a MULTIDOS diskette or an "alien" system diskette. MULTIDOS will not execute an "alien" DOS Basic. MULTIDOS may or may not successfully execute an "alien" system DOS library command or utility program.

MULTIDOS requires a system diskette, in drive 0, for disk I/O. Exceptions to this requirement are noted in this manual and specifically involve utilities that require diskette "swapping".

Certain machine language programs that are designed specifically to use an "alien" DOS may not execute properly with MULTIDOS.

Proper "operation" of certain MULTIDOS UTILITIES does not require a system diskette in drive 0. Specific system restraints are covered in the description for the particular utility in the SYSTEM UTILITIES section of this manual. MULTIDOS requires that a system diskette be in drive 0 when any of the utilities are called up.

MULTIDOS requires certain /SYS files present beyond the minimal system for proper operation and execution of certain functions.

LIBRARY COMMANDS

The following commands are MULTIDOS library commands and are normally entered after the "MULTIDOS" 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.

Multiple library commands are permitted provided a comma follows the last character of a given command.

EXAMPLES:

(1) DIR (A),DIR 1(A),DIR 2(A)<ENTER>

This multi-command will display the <A>llocation directory for drives 0, 1, and 2.

(2) RENAME NEW/CMD:2 TO OLD/CMD,DIR 2<ENTER>

This multi-command will rename "NEW/CMD" on drive 2 to "OLD/CMD", then pull a directory on drive 2.

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.

APPEND NEXT/TXT:2 to BEFORE/TXT:1<ENTER>

The file NEXT/TXT on drive 2 will be appended to the file BEFORE/TXT on drive 1.

NOTE: APPEND does not prompt you for diskette mounts. All diskettes must be mounted.

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
default	0	TOTAL ACCESS
KILL	1	TOTAL ACCESS
RENAME	2	RENAME, WRITE, READ, EXECUTE
not used	3	WRITE, READ, EXECUTE
WRITE	4	WRITE, READ, EXECUTE
READ	5	READ, EXECUTE
EXEC	6	EXECUTE ONLY
NONE	7	LOCKED OUT

Protection level with a value of 0 usually occurs when a diskette is PROTected with the master password. This will set both the access and update password to the diskette's master password which must be use when setting the protection level to a different value. The master password on your MULTIDOS diskette is PASSWORD.

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 or protection level attribute. Files with access attribution require the access password for any operation with the file.

All files naturally have an access and update password. This default password is 8 blanks (spaces) if none is specified. With the default password of 8 blanks, you do not need to enter the password. 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, where 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.

MULTIDOS can exectute multiple AUTO comands. To set up multiple AUTO commands, insert a linefeed (down arrow) between each MULTIDOS command or filespec you want executed upon power-up. This will show each AUTO command prior to execution.

EXAMPLE:

AUTO DIR<LF>
FREE<LF>
BASIC<ENTER>

However you may enter a multi-AUTO command by seperating the commands with commas. This will only show the multi-AUTO command once.

EXAMPLE:

AUTO DIR,FREE,BASIC<ENTER>

A maximum of 32 characters after the "AUTO " including blanks, linefeeds, and the terminating <ENTER> are allowed. If your entry exceeds 31 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.

To query an AUTO command, type:

AUTO ?<ENTER>

To execute an AUTO command, without re-booting, type:

AUTO %<ENTER>

NOTE: This command will convert a multiple AUTO command to a multi-AUTO command for this execution.

EXAMPLES:

(1) AUTO DIR<ENTER>

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

(2) AUTO
CLOCK<LF>
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". This command syntax will bypass the DATE entry requirement on power-ups.

EXAMPLE:

AUTO !BASIC<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 !#BASIC

This is the proper combination of the extended "!" and "#" AUTO functions. The "!" designator is only recognized if it is the first character after the mandatory blank following AUTO. The "#" designator is recognized if it immediately precedes the command.

BLINK Disables/enables the blinking cursor.

BLINK[switch]<ENTER>

BOOT Resets the computer.

This command will cause a "system" (4000 hex - 51FF hex) software reset.

BREAK Disables/enables the break key.

BREAK[switch]<ENTER>

BUILD Creates a "DO" file.

BUILD filespec<ENTER>

BUILD creates a disk file to store a series of keystrokes for execution by the DO command. The data in the disk file is interpreted by the DO command as keyboard entries.

This command will create 'filespec', if it does not exist, or add to 'filespec', if it exists, for "DO" execution. The maximum input line is 63 characters.

To BUILD a "DO" file follow these steps:

1. Type BUILD filespec where filespec is the name of the file for "DO" execution.
2. The prompt 'Type in up to 63 keystrokes.' will appear.
3. Enter the text. You may enter up to 63 characters before you need to press <ENTER>.
4. Keys are stored to the file exactly as typed. Exceptions are noted below.
5. Continue steps 2 thru 4 until the "DO" file is "BUILT".
6. To terminate a current "BUILD" and close the file, press the "BREAK" key.

If no extension is specified in the filespec, "/IDO" is used as the extension by default.

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

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 NUMBER 77<ENTER> = Pauses "DO" after displaying the message:

'INSERT DISK NUMBER 77'.

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

EXAMPLE:

$<ENTER> = Suppresses video output and continues "DO" file.

$This is the beginning of darkness!<ENTER>

Will display the message 'This is the beginning of darkness!', wait for a <SHIFT>, then turn off future video updates until a DO" file closes.

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 "REDO" message will be displayed and the "DO" file continued.

EXAMPLE:

%We the people of the United States, in Order to form a more<ENTER>
%perfect Union, establish Justice, insure domestic Tranquility,<ENTER>
%provide for the common defence, promote the general Welfare,<ENTER>
%and secure the Blessings of Liberty to ourselves and our<ENTER>
%Posterity, do ordain and establish this Constitution for the<ENTER>
%United States of America.<ENTER>

NOTE: The "%" character is required on each comment line.

Special BUILD characters are summarized in the following table:

BUILD Character	Action description
#	pause and waits for "SHIFT" key press prior to continuing
#zzzzz	pause after displaying zzzzz and wait for "SHIFT" key press prior to continuing
$	turn off video
$zzzzz	turn off video after displaying message zzzzz and wait for a "SHIFT" key press to continue
%zzzzz	display zzzzz and continue

EXAMPLE of BUILDing a file for DO execution:

```
BUILD BLAST/IDO<ENTER>
CLS<ENTER>
%LOADING BASIC<ENTER>
BASIC 3,61440<ENTER>
CLS<ENTER>
%GET READY FOR A BLAST !<ENTER>
RUN"BLAST/BAS"<ENTER>
<BREAK>
```

The above example shows that when DO BLAST is entered from MULTIDOS command mode that: The screen will clear. The message 'LOADING BASIC' will appear. BASIC will be loaded with 3 file buffers and memory set to 61440. The screen will clear. The message 'GET READY FOR A BLAST !' will appear. The program "BLAST/BAS" will load and execute.

CLEAR Zeroes Random Access Memory (RAM).

CLEAR<ENTER>

CLEAR zeroes RAM from 5200H to TOPMEM. The limit of free memory, called "TOPMEM", is the address pointed to by the contents of 4049H - 404AH MODEL I, and 4411H - 4412H MODEL III.

EXAMPLE:

(1) CLEAR<ENTER> (MODEL I)

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 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.

The RAM storage for the time is in the following locations:

Contents	MODEL I	MODEL III
hours	4043H	4219H
minutes	4042H	4218H
seconds	4041H	4217H

CLRDSK Zeroes unused granules.

CLRDSK [[:]d]

This command will "clean up" unassigned granules on the diskette specified. This is useful when you want to give a diskette to someone but do not want specific deleted files from being recovered. This command is not necessary for BACKUP, since BACKUP only copies assigned granules.

EXAMPLE:

CLRDSK :2<ENTER>

This will zero all unused granules on the diskette mounted in drive 2.

CLS Clear the screen.

This command simply clears the screen. This command is very useful in a "DO" file to clear the screen between large comments.

CONFIG Default power-up drive attributes.

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

param	meaning
drn	drivespec (not in parenthesis)
STEP=nn	nn is drn track to track access speed
DENSITY=q	q is the default density for drn upon power-up.
SIDES=ss	ss in the number of sides on a disk drive.

Sets individual drive stepping speeds and initial density.

The normal side, side zero, is accessed by specifying the drive number without any modifier. Side one is accessed by specifying the drive number immediately followed with an apostrophe. (This is the character obtained by keying in shifted 7 – Analogous to PRIME).

EXAMPLE:

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

Will write this info onto drive zero diskette (system disk), set RAM configuration for drive 2, and remove all SKIP's. The diskette in drive zero must not be write protected.

Syntax is forgiving, only the first two letters are considered.

i.e. CONFIG 2(ST=6,DE=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 (30mS for MODEL III).

NOTE: The 6mS speed is only attainable in the MODEL I with a double density hardware modification. The density's available are: S = single density, P = DBLDOS (tm) density, and D = double density.

CONFIG command has three other parameter specifications. They are:

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

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

CONFIG (X)<ENTER> writes RAM's configuration onto the system diskette and removes all SKIP's.

DATE Sets the date.

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

This command allows you to set the date. If you use the format DATE<ENTER>, MULTIDOS will display the date and time 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. Since you are prompted to input the DATE during power-up, the DATE does not have to be set again until the next power-up. MULTIDOS dates the files to the current RAM date; therefore, it is recommended that you set the DATE during power-ups.

If you do not want the DATE prompt to appear during future power-ups, use the AUTO ! command. There does not have to be anything following the "!" character.

DDAM Directory Data Address Marks (Model I only)

DDAM[[:]d] (type)<ENTER>

DDAM alters the Data Address Marks on a single density diskette's directory sectors.

d = drive number
type = O for 1771 FDC, MODEL I - TRSDOS (tm), NEWDOS/2.1, etc. compatible.
N for 1791/1793 FDC, MODEL III type.

This command does NOT un "read protect" a directory.

The address marks placed on a single density diskette by MULTIDOS cannot be easily read by TRSDOS, NEWDOS (not NEWDOS/80), and ULTRADOS. If you must write to an "alien" single density operating system, use DDAM to convert the address marks after writing. This will enable the "alien" system to easily read itself. Converted or non-converted address marks for the "alien" system can be easily read by MULTIDOS.

The "Deleted Data Mark" is used for MODEL I to/from MODEL III compatibility. If you do not intend to work with a MODEL III and want to use the "alien" system compatible address marks then patch a backup system diskette as follows:

```
Single Density:     PATCH DOS/SYS (REC=10,BYTE=176) 62
Double Density:     PATCH DOS/SYS (REC=20,BYTE=176) 62
"P" Density         PATCH DOS/SYS (REC=8,BYTE=176) 62
```

NOTE: The distributed value for this byte is 230d. The 62 will defeat the DDAM (N) function.

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 stepping or perform CALL's, etc.

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

Once the debugging facility is enabled - via DEBUG (ON) - it does not load and execute until one of the following conditions occurs:

1. If the interrupts are enabled and the <SHIFT><BREAK> keys, <LEFT>-<SHIFT><BREAK> MODEL III, are pressed simultaneously.

2. After an executable program, with protection level of less than EXEC, is loaded and before its first instruction is executed.

DEBUG offers three display formats:

(1) Register display only on the right hand side on the screen.
(2) Register display with indirect RAM plus any 64-byte "page" of RAM.
(3) Full screen, 256-byte "page" of RAM.

When DEBUG is initially loaded, the display will be blank. Press <O> to enable the display.

In the register display formats, 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 in format 2.

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 the carry flag is "set".

Table of codes for all the flag bits:

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

NOTE: Since DEBUG loads into the overlay are of RAM (4D00H to 51FFH), you cannot use it with MULTIDOS overlays or utilities which load below 5200H.

To return to the point where you entered DEBUG, provided you have not altered the contents of the PC or SP registers, type G<ENTER>.

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

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 <SPACE> is pressed.
F	none	Disables all display formats.
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 cursor. (see next page).
N	none	Provides continuous "I" commands until <SPACE> is pressed.
O	none	Enables the display formats.
P	none	Set display to register only mode (1).
Rrp aaaa	<SPACE>	Loads register pair rp with the value aaaa.
Q	none	Sets display to previous format 2 or 3.
S	none	Sets display to full screen memory mode (3).
U	none	Dynamic display update mode. Lets you observe the execution of a foreground task. Press <SPACE> to exit this mode.

Command	Entry Required	Operation Performed
X	none	Sets display to register mode (2).
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 characters 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> where switch = (N), or (NO), or (OFF)

DEVICE List current I/O devices.

This command lists: KI=keyboard, DO=video display, PR=line printer, RI=RS 232-C input (Model III only), RO=RS 232-C output (Model III only), and their routine entry points. These are the defined I/O devices.

DIR Display 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.

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.

You may enter as many options as you wish.

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 3, Multidos - 09/01/81, 40/40 tracks,  31 grans,  38.75 K
Filename                 Date     Tr,Sec   Eof    Seg   Lrl  Grans

GR/CMD          P=6  09/01/82     23,5    0/244   1     256     1
RS/CMD          P=6  09/01/82     24,0    4/252   1     256     1
SPOOL/CMD       P=6  09/01/82     23,0    4/254   1     256     1
VFU/CMD         P=6  09/01/82     22,0    9/250   1     256     2
```

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

By pressing a single key you can display a directory of a mounted diskette. The key must be the first keystroke after <ENTER> or <BREAK>. The keystrokes are:

"0" = DIR 0
"1" = DIR 1
"2" = DIR 2
"3" = DIR 3

DO Substitute disk file for keyboard input.

DO filespec<ENTER>

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

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

The "DO" file buffer is loaded into TOPMEM to TOPMEM – 290 decimal (122H). TOPMEM should be set BEFORE the "DO" file is executed, when you want to load a machine object code into high memory. The use of the multi–AUTO command, or the multiple AUTO command will permit you to accomplish this task during power–ups.

EXAMPLE:

AUTO TOPMEM xxxxx,DO STARTER<ENTER>

or AUTO TOPMEM xxxxx<LF>
DO STARTER<ENTER>

Please note that you may wipe out the "DO" buffer if you attempt to set TOPMEM from within the "DO" file.

DUMP Transfer RAM contents to disk file.

DUMP filespec (START=X'ssss',END=X'eeee'[,TRA=X'tttt'][,CIM][,TITLE])<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 "CMD" if none was specified in filespec.

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, the MULTIDOS command mode.

CIM specifies direct transfer to diskette sectors without 'load marks'.

TITLE causes DUMP to place a title block at the beginning of the file's contents.

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',CIM)<ENTER>

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

(2) DUMP D (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>

Where 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 blocks of printed TEXT.
X	Writes out to the system diskette, the current W, P, and T values, for default parameters on subsequent power-ups. The diskette in drive zero must not be write protected.
N=nn	nn = nulls after linefeed.
L	Send linefeed after carriage return.
C	direct printer output to RS-232-C (Model III only). (Affects a psuedo ROUTE PR RO except the data is formatted in accordance with the FORMS parameters.)

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, pagination is inhibited.
If "W" = 255 then no carriage return is sent after a specific character count. The width command is EXACT in MULTIDOS, no adjustments are necessary. MULTIDOS is distributed with W=255, P=66, and T=66 (no pagination nor character counting).

FORMS will always recognize a CHR$(11) as a linefeed without a carriage return (vertical tab one line). This can be used to send linefeeds to certain printers which do not respond to multiple linefeed characters.

A CHR$(12) will generate a formfeed. A CHR$(11) will generate a line feed.

A CHR$(13) will generate a carriage return.

When the ESC character, CHR$(27), is encountered, FORMS will not count this nor the next character as one of the characters to be counted for page width, nor will the next character be recognized as a stand alone character. i.e. LPRINT CHR$(27)+CHR$(12). This will send BOTH codes to the printer without generating a formfeed, and leave the character counter where it was before this ESC pair was sent to the printer.

The RAM address for these values are as follows:

Print lines per page.	(P)	NOT directly available.
Printed text lines.	(T)	4028H
Line counter.		4029H
Width of text.	(W)	402AH
Character counter.		402BH
Spaces between pages.	(S)	lower 7 bits of 402CH

Byte 402CH high bit is the pagination bit. If this bit is set then pagination is enabled, And the (P) value is the sum of 4028H and the lower 7 bits of 402CH.

FORMS<ENTER>
Will display the current FORMS parameters.

Whenever the FORMS command is entered the current values are echoed on the display. However, if the "I" parameter is specified, the display will not echo the settings. This is useful for initializing the FORMS parameters from within SUPERBASIC without messing up the display.

FREE Displays the number of available file spaces, free granules, and kilobytes 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. FREE will total all the directory space, and the diskette space and display this total.

HASH Returns HASH code of filespec.

HASH BASIC/CMD<ENTER>

Returns: Hash code = F0

The hash code byte is the byte on the <H>ash <I>ndex <T>able (HIT) located at the directory track. Several people use the jargon "HIT" byte.

HELP Provides a brief explanation of LIBrary commands.

HELP[COMMAND]<ENTER>

This command will assist you in using the MULTIDOS LIBrary commands along with the format required to execute each command.

If COMMAND is omitted or incorrect, MULTIDOS will provide a list of the available COMMANDS which are in the HELP file.

KEYBRD Set keyboard attributes.

KEYBRD[([function = q][,function2 = q])]<ENTER>

q = "YES" or "Y", if the function is desired. "NO" or "N", if the function is not desired.

function is defined by:

B = blinking cursor.
C = <CLEAR> key.
E = Epson graphics converter.
G = Keyboard graphics converter.
K = <BREAK> key.
L = lowercase. (MODEL I only)
R = repeat keyboard. (MODEL I only)

and one function which does not use a "switch" but requires a decimal value is :

W=nnn where nnn is the decimal value of the cursor character.

None, one or more functions may be specified. If any KEYBRD attributes are changed they are updated to the system disk immediately. However, they do not go into affect until a future reboot/power-up of the system diskette.

'E' function, the Epson graphics converter, when enabled, adds 32d to the value of each graphics character sent to the device defined as the printer. This offset is required by Epson printers that print TRS-80 graphic characters.

On the MODEL I, if the 'G' function is set to ON, then on reboot/power-up, the keyboard graphics driver is loaded into high memory and the TOPMEM address is reset. (GR/CMD need not be present on disk.) The MODEL III self-configures DOS for keyboard graphics and does not use high memory.

If the 'G' function is set to ON, the <SHIFT> & <CLEAR> keys pressed simultaneously are used to switch the keyboard between graphics entry and normal (ASCII) entry modes. Refer to the keyboard graphics chart at the end of the LIBRARY COMMANDS section of this manual for graphic key descriptions.

Syntax is forgiving - only the first letter is considered. KEYBRD (eQs4X=YEP), the same as KEYBRD (e=Y), will configure DOS such that the Epson graphics converter is set to ON.

KEYBRD<ENTER> echoes system disk's current KEYBRD attributes.

EXAMPLE:

KEYBRD (K=N,C=N,B=Y,W=176)

On future reboots/power-ups:

1. The <BREAK> key is defeated.
2. The <CLEAR> key is defeated.
3. The blinking cursor is on.
4. The cursor character is 176 decimal.

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 its 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 67FFH except 'BLINK', 'BOOT', 'BREAK', 'CLEAR', 'CLOCK', 'CLS', 'DEAD', 'DEBUG', 'LIB', 'LOAD' (MOD III), and 'VERIFY', which are contained in DOS1/SYS (4D00H to 51FFH).

LINK Simultaneous output.

The four possible inputs are:

LINK PR DO<ENTER>
LINK DO PR<ENTER>
LINK RO DO<ENTER>
LINK RO PR<ENTER>

LINK PR DO will send to the printer whatever goes to the display.
LINK DO PR will send to the display whatever goes to the printer.
LINK RO DO will send to the display whatever goes to the RS-232 communications board (MODEL III only).
LINK RO PR will send to the printer whatever goes to the RS-232 communications board (MODEL III only).

To un-LINK all or any LINKed devices:

LINK<ENTER>

LIST Display diskette file.

LIST filespec<ENTER>

This command lists a file on the display. If the file contains control codes, the display of the control codes is suppressed. The <BREAK> key will terminate the listing of a file. The simultaneous pressing of <SHIFT><@> will pauuse the display until any key is pressed.

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 6800H, you cannot use some MULTIDOS library commands without overlaying your program.

PATCH Modify the contents of a diskette file.

PATCH filespec (REC=nn[,BYTE=yy]) b1[;b2][;b3][;b4]<ENTER>

nn = physical record in 'filespec' NOTE: The first record is 0.

yy = relative byte in physical record 'nn'. This input can be in decimal (0 to 255) or hexadecimal (X'00' to X'FF').

b1,b2,b3,b4,etc. = DECIMAL value of replacement bytes.

The physical record will automatically advance if 'b2' up causes the relative byte to exceed 255 decimal.

This command will permit you to make changes to ANY disk file.

In order to PATCH a file you need to know the physical (NOT logical) record to reference, and the relative byte in that sector. PATCH automatically ignores the logical record, and password protection on all files. The first record of a file is 0 (zero).

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
DATE	update all files to current RAM date (used with LOCK or UNLOCK)

This command will change the diskette MASTER PASSWORD or lock/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 its write protect notch covered.

EXAMPLE:

PROT (LOCK)<ENTER>

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

The master PASSWORD on your MULTIDOS diskette is PASSWORD.

If no parameters are entered with PROT, you will be prompted to change the diskette's name (ID) and the date. There is NO checking for a valid name or date; use what you wish! If you only want to change one of the name/date pair, simply <ENTER> for the one not to be changed. To prematurely exit this command, press <BREAK>.

RENAME Change filename.

RENAME [$]filespec1 [to] filespec2<ENTER>

This command will change the name of filespec1 to filespec2. If the optional '$' is specified then the renamed file's date will be changed to the current RAM date.

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, and the diskette with the filespecs must not be write protected.

RESTOR Recover a KILLed filespec.

RESTOR filespec[:][d]<ENTER>

This command will let you attempt to recover a file inadvertently killed. If the file space on diskette has been re-assigned you will be notified of this condition and restoration will not take place.

The drive number is mandatory for RESTOR to work properly. Since MULTIDOS uses the next unused directory entry for a new file on a given diskette, you should recover a file prior to writing a new file to this diskette.

ROUTE Redirects one device to another.

ROUTE will modify the driver address of a device, to allow the system to redirect calls to this device.

The following are the MULTIDOS devices available.

```
KI = keyboard              RI = R-232-C input (MODEL III only)
DO = display               PR = printer
RO = RS-232-C output (MODEL III only)
```

EXAMPLE:

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

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

To un-ROUTE you simply enter:
ROUTE<ENTER>

SETCOM Initializes RS-232-C, and sets parameters.
(Model III only.)

SETCOM[([BAUD=rr][,sel][,WORD=ww][,STOP=][,psw][,tsw][,DTR][,RTS][,wsw])]<ENTER>

```
rr  = BAUD rate. Permitted are: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2000,
      2400, 3600, 4800, 7200, 9600, or 19200.
sel = ODD or EVEN
ww  = Word length. Permitted are: 5, 6, 7, or 8.
bb  = Stop bits. Permitted are: 1 or 2.
psw = Parity Switch. (parity on/off).
      Permitted are: PE for parity enabled, and PD for parity disabled.
tsw = Transmit switch. TE = Transmitter enabled, TD = Transmitter disabled.
```

DTR if entered sets DTR high, otherwise DTR will be set low.
RTS if entered sets RTS high, otherwise RTS will be set low.
wsw = Wait Switch. Permitted are WAIT or NOWAIT. If NOWAIT is selected, MULTIDOS will return from the RS-232-C routine whether the byte was received/transmitted or not.

SETCOM<ENTER> will display the current RS-232-C settings.

NOTE: MULTIDOS does not check for the presence of a RS-232-C serial board.

SKIP Read a 40 track diskette in a 80 track drive.

SKIP[[:]d]<ENTER>

d = drive number to read a 40 track diskette in a 80 track drive.

The permitted drive numbers are 1, 2, and 3. If 0 is used, all skips are removed.

This command will allow you to READ a 40 track diskette in an 80 track drive.

If no drive number, or 0, is specified, then all SKIP's are removed.

NOTE: Whenever the system diskette is reconfigured with CONFIG, all skips are removed. To determine if SKIP is present, use CONFIG (?)<ENTER>.

TIME Sets time.

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

Use syntax (1) to set the current time in RAM.
hh = hour, mm = minute, ss = second.

Use syntax (2) to display, in hh:mm:ss format, on a one time basis, the time currently stored in RAM (MOD III only). A non-powerup REBOOT will retain the time previously set in RAM. The hexadecimal locations 4041H to 4043H - MODEL I, 4217H to 4219H - MODEL III store the current time, in ss, mm, hh.

TOPMEM Sets upper MULTIDOS system memory.

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

ddddd = A decimal number from 28671 to 65535.
nnnn = A hexadecimal number from 6FFF to FFFF.

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, and others check the value of TOPMEM and operate at that limit. The value is placed in

RAM at locations 4049H & 404AH – MODEL I, or 4411H & 4412H – MODEL III. The default value is the top of system RAM. When version (3) is entered, MULTIDOS will display the highest usable byte currently available for MULTIDOS, MULTIDOS utilities, and SUPERBASIC usage.

VERIFY Reread at written sector.

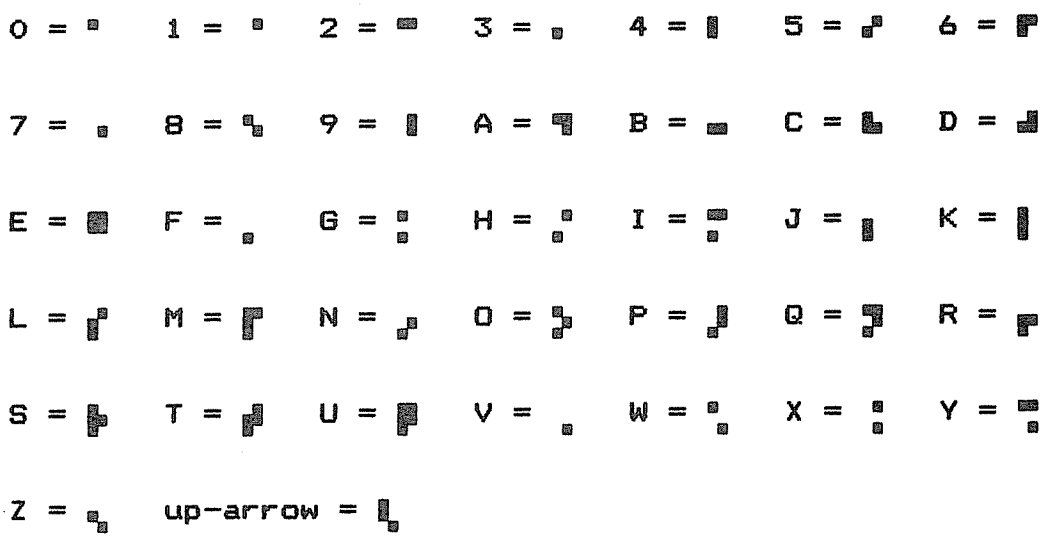
VERIFY switch<ENTER>

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

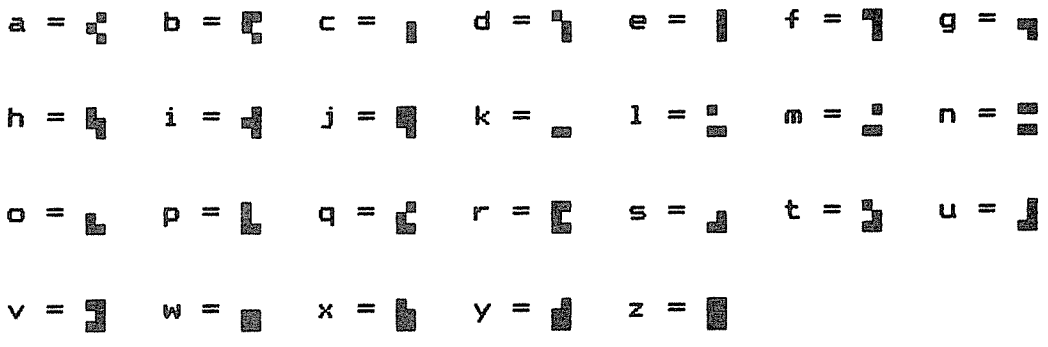
These are the characters obtained when the keyboard graphics driver is on.

Graphics

(pixels lit)



lowercase (or SHIFTED without lowercase keyboard)



BACKUP/CMD Duplicate a diskette.

BACKUP[A] [:][drv] [:][drv]<ENTER>

This utility will duplicate all files from one diskette to another. The 'A' option, if specified, causes an absolute FORMAT and does not check or warn the user if the destination diskette contains data. Source and destination drive numbers may be specified in the entry.

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, place a write-protect tab over the write protect notch of the source diskette. BACKUP does not verify at the file transfer stage unless 'VERIFY' is active.

NOTE: BACKUP will not backup a NEWDOS/80 Model III 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 X.X

Q1 Which drive contains the source diskette? '

will appear. The program is awaiting a numerical response of 0 to 3, or 0' to 3' followed by <ENTER>. If the source drive number was specified in the command entry, this question will automatically be bypassed.

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 to 3, or 0' to 3' followed by <ENTER>. If the destination drive was specified in the command entry, then this question will be bypassed.

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 "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.'

where "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 formatted and the 'A' option was not specified in the BACKUP command then the message:

'Diskette previously formatted.

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

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

The above message and query will not appear if the 'A' option was specified as part of the BACKUP command. If this is the case then a (re)-format of the destination diskette will take place using the source diskette density.

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 zero.' or 'Insert SYSTEM <ENTER>'

will be displayed if either the source or destination diskette was in drive 0. This prompts you to insert a MULTIDOS diskette to return to the state prior to entering BACKUP.

You may change the format pattern bytes which BACKUP/CMD generates. The system diskette should be in drive 0. To do so use the ZAP utility and specify Display File Sectors from the ZAP menu. When the relative file sector prompt appears just press <ENTER> and relative file sector 0 of BACKUP/CMD will be displayed. You may change the format bytes to a different pattern. The format pattern bytes are located at relative bytes 8E hex and 8F hex, of this sector, for double density pattern. The format pattern bytes for single density are located at 9E hex and 9F hex of this sector. For further details on the use of ZAP/CMD see the description towards the end of this section of the manual.

Display of relative sector 0 of BACKUP/CMD using ZAP utility.

```
B HEX00 0F7A 4261 636B 7570 2020 2030 3938 327E .zBackup   0982~
A    10 2A20 2020 204E 4F54 4943 4520 2020 202A *    NOTICE    *
C DR 20 2A20 2028 6329 2020 2031 3938 3220 202A *  (c)   1982  *
K  0 30 2A20 436F 736D 6F70 6F6C 6974 616E 202A * Cosmopolitan *
U    40 2A20 2045 6C65 6374 726F 6E69 6373 202A *  Electronics *
P TR 50 2A20 436F 7270 6F72 6174 696F 6E2E 202A * Corporation. *
/ 03 60 2A20 2020 204E 4F54 4943 4520 2020 202A *    NOTICE    *
C 03 70 2A2A 2A2A 2A2A 2A2A 2A2A 2A2A 0182 004D ************...M
M    80 4442 4C20 4445 4E53 4954 5920 3D20 6DB6 DBL DENSITY = m.
D SE 90 534E 4720 4445 4E53 4954 5920 3D20 E5E5 SNG DENSITY = ..
   05 A0 5072 6573 7320 2245 4E54 4552 2220 7768 Press "ENTER" wh
   05 B0 656E 2003 0A43 6F6D 706C 6574 6564 2E0D en ..Completed..
      C0 7468 6520 736F 7572 6365 2064 6973 6B65 the source diske
FILE  D0 7474 6520 6973 2069 6E20 6472 6976 6520 tte is in drive
0000  E0 002E 0374 6865 2064 6573 7469 6E61 7469 ...the destinati
000H  F0 6F6E 2064 6973 6B65 7474 6520 6973 2069 on diskette is i
```

CAT/CMD Obtain a directory of a TRS-80 (tm) diskette.

CAT[[:][d] [(M[,I][,S])]<ENTER>

M = wait for "alien" diskette mount in drive 0.
I = include invisible files.
S = include system files.

This utility will pull a directory on practically any MODEL I or MODEL III diskette, regardless of the address marks, density, or sector/granule format.

CONVERT/CMD Change address marks on single density diskettes. (MODEL III only)

CONVERT [:][drv]

This Model III utility will change the address marks on an "alien" system formatted single density diskette. In order for MULTIDOS to read this diskette, the address marks must be changed. To accompiish this, simply enter the drive number of the target diskette after CONVERT.

EXAMPLE:

CONVERT :2

Will change the address marks on the diskette in drive 2.

If no drive number or zero is specified you will be prompted to insert the target diskette in drive zero. After the process is completed or an error occurs, you will be prompted to insert a MULTIDOS system diskette.

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 filespecs from an alien/system/data diskette to any other alien/system/data diskette. However, multiple swapping will be required to bring in the correct system overlays. Please follow the prompting provided by COPY. Whenever the prompt:

'Press "ENTER" when a MULTIDOS system diskette is in drive zero.' or 'Insert SYSTEM <ENTER>'

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 ON COPY

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. Store source filespec sector allocation.
10. READ in as much of source filespec into available RAM.
11. Check for swap.
12. Attempt to OPEN the destination filespec, if found store current sector allocation.
13. Calculate and store the amount of free sectors on the destination drive. (convert granules)
14. If the destination filespec exists, add and store the total number of sectors available.
15. Compare 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 doesn'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.

DDT/CMD Disk drive timer.

DDT<ENTER>

DDT/CMD is a disk drive diskette rotation speed timer. The utility will ask for the drive that is to be timed. At that point you may press <BREAK> to exit the utility. Enter the number of the disk drive. A diskette, formatted or un-formatted, must be in the selected disk drive. An instantaneous speed will be displayed as well as a smoothed speed. The diskette rotation speed should read 300 revolutions per minute (RPM) plus or minus 1 percent (297 RPM to 303 RPM). To restart the utility use the left-arrow key.

NOTE: DDT does an OUT 254,0 to set to 1.77 MHz (Model I), or OUT 95,0 to set to 2.02752 MHz (Model III), most high speed modification clocks.

FORMAT/CMD Prepare a diskette for data storage.

FORMAT[[:]d] [A]<ENTER>

This utility will prepare a diskette that will not contain the MULTIDOS operating system, which will leave the maximum disk space for your files. These diskettes are referred to as "DATA" diskettes.

'd' = drive number and is optional.
A = absolute FORMAT - does not check or warn user if the target diskette contains data.

Operation is as follows:

'MULTIDOS Formatter Program - Model I(II) - Version X.X

Which drive contains the diskette to be formatted? '

Reply with 0, 1, 2, or 3 for side0, and 0', 1', 2', or 3' for side1 of two-sided drives. No query if drive number was specified in FORMAT command.

'Name of diskette to be formatted? '

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

'Track count for this diskette (2 to 96)? '

Reply with the desired track count. FORMAT can only format up to the drive capaacity. If you only have 40 track drives, you cannot format 80 tracks. If you do not enter a track number, FORMAT will default to 40 tracks.

'Date for diskette to be formatted? '

Reply with a date in the format mm/dd/yy. FORMAT will default to the current RAM date.

'The master PASSWORD for this diskette.

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

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

Reply with S, D, or P.
Will default to current system configuration.

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

XX = One less than track count.
Default is to track 17 decimal (11 hex).

If the diskette to be formatted contains data, MULTIDOS will suspend formatting. Only a "Y" response will cause formatting to continue. This is not the case if the 'A' option was specified as part of the FORMAT command.

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.

You may change the format pattern bytes and/or the default track count in FORMAT/CMD. The system diskette should be in drive 0. To do so use the ZAP utility and specify Display File Sectors from the ZAP menu. When the relative file sector prompt appears just press <ENTER> and relative file sector 0 of FORMAT/CMD will be displayed. You may change the format bytes to a different pattern. The format pattern bytes are located at relative bytes 8E hex and 8F hex, of this sector, for double density pattern. The format pattern bytes for single density are located at 9E hex and 9F hex of this sector. The default track count is in hexadecimal. The default format track value is located at relative byte AF hex of this sector. For further details on the use of ZAP/CMD see the description towards the end of this section of the manual.

Display of relative sector 0 of FORMAT/CMD using ZAP utility.

```
F HEX00 0F7A 464F 524D 4154 2020 2030 3938 327E .zFORMAT   0982~
O    10 2A20 2020 204E 4F54 4943 4520 2020 202A *     NOTICE     *
R DR 20 2A20 2028 6329 2020 2031 3938 3220 202A *  (c)   1982   *
M  0 30 2A20 436F 736D 6F70 6F6C 6974 616E 202A * Cosmopolitan *
A    40 2A20 2045 6C65 6374 726F 6E69 6373 202A *  Electronics *
T TR 50 2A20 436F 7270 6F72 6174 696F 6E2E 202A * Corporation. *
/ 02 60 2A20 2020 204E 4F54 4943 4520 2020 202A *     NOTICE     *
C 02 70 2A2A 2A2A 2A2A 2A2A 2A2A 2A2A 0182 004D **********. ▫.M
M    80 4442 4C20 4445 4E53 4954 5920 3D20 6DB6 DBL DENSITY = m▫
D SE 90 534E 4720 4445 4E53 4954 5920 3D20 E5E5 SNG DENSITY = ..
   00 A0 4446 4C54 2046 4D54 2054 524B 203D 2028 DFLT FMT TRK = (
   00 B0 3E80 CD7B 4F18 0F3E 00CD 9344 3AEC 370F > .{O..>..▬D:.7.
      C0 38FA 3C32 2C50 3A3E 50CD 9B4F 2298 5121 8.<2,P:>P.▌O"▫Q!
FILE D0 8651 CD67 4416 00CD 2B50 CD33 4FC2 2E4F ▫Q.gD...+P.3O..O
0000 E0 3A3E 5057 3C32 3E50 CD98 4F20 CA3A 314D :>PW<2>P.▫O .:1M
000H F0 CD7B 4F3E 00EE 80E6 883E EE28 023E F632 .{O>.. . ▫>.(.>.2
```

GR/CMD Configure keyboard for graphics

GR<ENTER>

GR/CMD is the MODEL I method of producing graphics from the keyboard. It loads into high memory and resets the TOPMEM address. A pseudo-version of this utility is loaded automatically, at reboot/power-up, if the KEYBRD function 'G' is enabled. (Refer to – LIBRARY COMMANDS; KEYBRD).

To produce keyboard graphics press <SHIFT><CLEAR>. To return to normal keyboard characters press <SHIFT><CLEAR> again. Refer to the keyboard graphics chart at end of the LIBRARY COMMANDS section of this manual for graphic key descriptions.

MEM/CMD Machine memory test

MEM<ENTER>

MEM is a RAM memory test utility. It will test random access memory from 4000H to TOPMEM. If a test byte fails it will be indicated on the video along with the bit number and the utility will exit.

RS/CMD Machine Memory Scanner.

RS<ENTER>

This utility can 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.

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. For example, to reserve a 2K buffer the reply to the above question would be 8<ENTER>.

(2) '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 - MODEL I, 4411h - MODEL III). 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 <RIGHT-SHIFT>-<BREAK>. You will be asked if the buffer is to be saved. A "Y" response will save the buffer contents, and a "N" response will reset the pointers and send a carriage return character 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.

TAPE/CMD Tape to disk transfer utility

TAPE<ENTER>

This utility allows you to transfer 500 baud system programs from tape onto diskette. The utility will prompt:

ENTER "T" to read TAPE, or "D" to return to DOS?

Ready the tape deck and position tape to start of file. Enter <T> to begin loading the system tape.

If the file on tape does not load properly or is not contiguous an error message will appear and the tape load will exit.

If the file loads properly then it is ready to be transferred to diskette.

The START, END and ENTRY points of the program will be displayed.

NOTE: The following prompts refer to the disk file to be created.

You will be asked if the just-loaded program from tape, is to be modified. If so respond <Y>, if not then respond <N>.

If <Y> is entered then you will be prompted:

Are interrups to be enabled or disabled (E or D)?

Enter <E> if the interrupts are to be enabled before the program executes. Enter <D> if the interrupts are to be disabled before the program executes.

Enter new base address in HEX ?

Enter a new base address for the program to load at. This enables the program to be loaded from disk to a non-conflicting area with DOS. The suggested base address should be greater than 5300 hex. An appendage will be added to the disk file such that the program will be transfered to its proper operating location after it is loaded from disk.

A third prompt will appear:

Initialize LEVEL II type DCB & RESTART vectors (Y OR N) ?

Respond <Y> if the program is to operate in a LEVEL II environment. This will allow programs that normally operate in a LEVEL II to be executed properly after loading from disk. This is done by adding an appendage to the disk file which creates the LEVEL II environment before any program execution. Respond <N> if the program is not to operate in a LEVEL II environment.

Whether the program is to be modified or not the following prompt will appear:

Is a new filespec required (Y or N) ?

Enter <Y> if a new filespec is required. If <N> is entered the name of the file loaded from tape will be used and the extention /CMD will be appended to the filename.

Filename please?

will appear if you choose to give the program a new filespec.

Once all prompts are answered the program will be transferred to diskette.

NOTE: Some machine language programs may not execute properly after transfering to diskette.

EXAMPLE:

ENTER "T" to read TAPE, or "D" to return to DOS?

<T> is entered and the system tape loads. After loading the START, END and ENTRY points are displayed.

Is the PROGRAM "SAMPLE" to be modified (Y or N)?

Y is entered.

Are interrupts to be enabled or disabled (E or D)?

D is entered.

Enter new base address in HEX?

7000 is entered.

Initialize LEVEL II type DCB & RESTART vectors (Y or N) ?

Y is entered.

Is a new FILESPEC required (Y or N)

Y is entered.

Filename please?

SPLATT/CMD:1 is entered.

The file is transferred to diskette on drive 1. When the program, SPLATT/CMD, is called up from diskette, it will be loaded beginning at 7000 hex, the interrupts will be disabled, the LEVEL II environment will be initialized, and the program will be transferred to its proper operating location and begin to execute.

VFU/CMD

This versatile file utility (VFU) provides for five frequently needed disk operations: moving of files from one diskette to another, multiple file copying, purging of files, printing a disk directory, and menu based execution of all programs on a disk. VFU will prompt the user if a system diskette is required in drive 0 for proper operation of VFU.

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>

VFU/CMD will allow you to remove your **MULTIDOS** system diskette from drive zero to allow you to copy or move between 2 diskettes, using a 2 drive system, or purge a diskette using a 1 drive system, provided a warning message does not appear.

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

Versatile File Utility – Version X.X
(c) 1982 Cosmopolitan Electronics Corporation.

Press	Action
"C"	Copy
"E"	Execute
"H"	Hard Copy
"M"	Move
"P"	Purge

Choice

A rapidly winking cursor will indicate that user input is required.

To exit the utility, when the winking cursor is displayed, press <CLEAR>. 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 select the files to be copied, press "S". To copy the entire diskette, press "T". Then the utility will ask:

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? Destination drive?

NOTE: The source and destination drive cannot be identical. If this situation occurs you will be re-prompted for the source and destination drives.

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. The shifted left arrow will reposition the cursor to the first filename. To remove a "+" position the cursor over the plus and press <N>.

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

After marking the files 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.

NOTE: VFU's copy command can also be used as a transfer utility to copy files from TRSDOS (tm) Model III double density diskettes. The minimum configuration required is 2 drives. This will allow you to copy TRSDOS V1.3 files to a non-TRSDOS diskette in another drive.

VFU MOVE COMMAND

This command performs the same as VFU's COPY command except the file on the source drive will be purged after the copy is completed.

VFU EXECUTE COMMAND

To execute programs from a diskette, use the "E" command. The utility will then display the drive, diskette name and date, the free space and an alphabetical directory. 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 select the files to be purged, press "S". To purge all files from the diskette, press "T". 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. The shifted left arrow will reposition the cursor to the first filename. To remove a "+" position the cursor over the "+" and press <N>.

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

After marking the files 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. To exit the utility when the normal cursor is displayed, press <BREAK>. 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 width will be sized to fit inside a diskette jacket.

ZAP/CMD Disk sector/memory modify utility

ZAP<ENTER>

ZAP/CMD, or ZAP, is a quick, simple way to read and write diskette sectors, file sectors, memory and fix disk directories.

I - MODES OF ZAPPING

A. DISKETTE SECTORS

This function is the default for ZAP. It will properly step through a single density, double density, or "P" density diskette provided the target diskette has its respective configure byte updated (only possible if a "DIR" was performed). There is no configuration for properly stepping through a TRSDOS (tm) MODEL III diskette (sector numbers 1-18 vs 0-17). 35 is the default track number when a step is executed that would decrease the track number below 0. Use of this function requires user discipline in selecting the track number. ZAP uninhibitly allows track and sector numbers from 0-255.

B. FILE SECTORS

The primary function of ZAP is to access files, regardless of protection, regardless of density, regardless of the operating system. However, the filespec must be in the directory.

ZAP has the ability to display and modify file sectors on the following operating systems:

MODEL I	MODEL III
TRSDOS 2.0 to 2.3	TRSDOS
NEWDOS/21	LDOS
ULTRADOS	DOSPLUS
VTOS all	NEWDOS/80
DOSPLUS all	
LDOS all	
NEWDOS/80 all	

NOTE: As of this writing TRSDOS 2.7, Model I double density, is not supported.
NOTE: TRSDOS MODEL III does not have their system files in the directory.

In addition, ZAP requires the filename to be entered in the case which the file will be in the directory. This may seem strange to some, but there are ways to enter a lower case file name! (Usually upper case). Although ZAP will search all drives for a given filespec, it is suggested that the drive number be appended to the filename. i.e. FILENAME/EXT:D. This will greatly speed up the purpose at hand, because ZAP performs an analysis on each diskette prior to the HIT search for the filespec's hash code.

To display file sectors enter the appropriate filename. You will be prompted for the relative sector in the file to display. Default is the first sector of the file. A request for display of a sector that is out of range displays the last sector.

C. MEMORY

Memory can be addressed in one byte increments.

II - USER INPUT

A. VALUE KEYS

Drive numbers are always entered in decimal form. Track numbers, sector numbers, file sectors and memory addresses are interpreted as decimal, unless "H" - hexadecimal notation - is appended to the number.

B. BREAK KEY

Pressing the "BREAK" key will return the user to the previous query. When the ZAP command prompt is displayed, pressing "BREAK" will exit the ZAP utility.

NOTE: DRIVE, TRACK, and SECTOR are considered the same query.

III. DISPLAY MODES AND MOVEMENT

A. Diskette sectors

Key pressed	Action
up arrow	increase track count by one
down arrow	decrease track count by one
right arrow	increase sector by one
shift right arrow	increase sector by one
left arrow	decrease sector by one
shift left arrow	decrease sector by one
"T"	reselect track/sector
"S"	reselect sector

B. File sectors

Key pressed	Action
up arrow	increase file sector by one
down arrow	decrease file sector by one
right arrow	increase file sector by one
shift right arrow	increase file sector by one
left arrow	decrease file sector by one
shift left arrow	decrease file sector by one

C. Memory

Key pressed	Action
up arrow	increase address by 256/100H
down arrow	decrease address by 256/100H
right arrow	increase address one
shift right arrow	increase address by 16/10H
left arrow	decrease address one
shift left arrow	decrease address by 16/10H

IV. MODIFICATION AND UPDATE

A. Pressing "M" after a displayed 256 bytes of a MEMORY page or a diskette SECTOR, will place the user in the modification mode. This will be noted by the dual blinking cursors – one in the HEX area and the other in the correspondind ASCII area.

B. The four arrow keys will move the cursors in the direction of the arrows.

C. The "@" key will toggle the user between HEX and ASCII modification mode. This is indicated by the presense of 'HEX' or 'ASC' in the upper left hand corner while displaying a sector.

D. MEMORY modification is effective immediately. SECTOR modification are not effective until the user presses "ENTER" to terminate modification, and "ENTER" a second time to update the sector.

FIX DIRECTORY COMMAND

This command is designed to fix the directory of MULTIDOS diskettes. The FIX DIRECTORY command of ZAP will repair the GAT sector for read errors and for erroneous granule allocation bytes for files. It will not repair the GAT sector for erroneous granule allocation bytes for granule lock-out. It will fix the HIT sector for read errors and will fix erroneous HIT bytes using directory file sector information. It will repair the directory file sectors for read errors but will not repair erroneous directory file sector entry information.

Display of GAT (granual allocation table) sector using ZAP utility.

```
HEX 00  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
    10  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFD ................
    20  FCFC FCFF FFFF FFFF FFFF FFFF FFFF FFFF ................
DRV 30  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
 1  40  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
    50  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
    60  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC ................
TRK 70  FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC ................
017 80  FCFC FCFF FFFF FFFF FFFF FFFF FFFF FFFF ................
11H 90  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
    A0  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
SEC B0  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
000 C0  0303 0303 0542 FF00 00B0 8F00 2323 E042 .....B....▀.##.B
00H D0  4D75 6C74 6964 6F73 3039 2D32 352D 3832 Multidos09-25-82
    E0  0D20 2020 2020 2020 2020 2020 2020 2020 .
RPT F0  2020 2020 2020 2020 2020 2020 2020 2020
```

Display of relative file sector 18 of DOS9/SYS (single density) using ZAP.

```
D ASC00 3C53 5041 4345 3E20 0182 0063 6973 2048 <SPACE> . ▪.cis H
0    10 454C 4420 646F 776E 2E0D 5041 5443 4820 ELD down..PATCH
S DR 20 4649 4C45 5350 2028 5245 433D 6E6E 2C7B FILESP (REC=nn,{
9  1 30 4259 5445 3D79 797D 2920 6231 7B3B 6232 BYTE=yy}) b1{;b2
/    40 7D7B 3B62 337D 7B3B 6234 7D7B 3B62 357D }{;b3}{;b4}{;b5}
S TR 50 3C45 4E54 4552 3E0A 0A6E 6E20 3D20 7068 <ENTER>..nn = ph
Y 25 60 7973 6963 616C 2072 6563 6F72 6420 6E75 ysical record nu
S 19 70 6D62 6572 206F 6620 4649 4C45 5350 2E20 mber of FILESP.
     80 2866 6972 7374 2072 6563 6F72 0182 8063 (first recor.▪ c
  SE 90 6420 6973 207A 6572 6F29 0A79 7920 3D20 d is zero).yy =
  03 A0 7265 6C61 7469 7665 2062 7974 6520 696E relative byte in
  03 B0 2072 6563 6F72 6420 226E 6E22 2E0A C530  record "nn"...0
     C0 2074 6F20 3235 3520 6465 6369 6D61 6C20  to 255 decimal
FILE D0 6F72 2058 2730 3027 2074 6F20 5827 4646 or X'00' to X'FF
0018 E0 2720 6865 782E 0A62 312C 6232 2C62 332C ' hex..b1,b2,b3,
012H F0 6574 632E 203D 2064 6563 696D 616C 2076 etc. = decimal v
```

```
M2286H  3F45 7874 7261 2069 676E 6F72 6564 0D00 ?Extra ignored..
M2296H  CD05 1FB7 2012 237E 23B6 1E06 CAA2 1923 ...▄ .#~#▄... ▪.#
M22A6H  5E23 56EB 22DA 40EB D7FE 8820 E3C3 2D22 ^#V."▪@... ▪ ..-"
M22B6H  1100 00C4 0D26 22DF 40CD 3619 C29D 19F9 .....&"▪@.6..▐▪..
M22C6H  22E8 40D5 7E23 F5D5 7E23 B7FA EA22 CDB1 "▪@▪~#..~#▄..▪"▪▪
M22D6H  09E3 E5CD 0B07 E1CD CB09 E1CD C209 E5CD ..............▪▪
M22E6H  0C0A 1829 2323 2323 4E23 4623 E35E 2356 ...)####N#F#.^#V
M22F6H  E569 60CD D20B 3AAF 40FE 04CA B207 EBE1 .i`...:▀@...▪...
M2306H  722B 73E1 D55E 2356 23E3 CD39 0AE1 C190 r+s..^#V#..9...
M2316H  CDC2 0928 09EB 22A2 4069 60C3 1A1D F922 ...(.."▪@i`....▀
M2326H  E840 2ADF 407E FE2C C21E 1DD7 CDB9 22CF .@*.@~.,.....▄".
M2336H  282B 1600 D50E 01CD 6319 CD9F 2422 F340 (+.....c..▐▪$".@
M2346H  2AF3 40C1 7E16 00D6 D438 13FE 0330 0FFE *.@.~....8...0..
M2356H  0117 AABA 57DA 9719 22D8 40D7 18E9 7AB7 ..▐▀W.▀.".@...z▀
M2366H  C2EC 237E 22D8 40D6 CDD8 FE07 D05F 3AAF ..#~".@......_:▀
M2376H  40D6 03B3 CA8F 2921 9A18 1978 56BA D0C5 @..▪▪.▀)!▐..xV▐..
```

Display of memory page 2286H using ZAP utility.

DBLFIX/CMD Fix boot sector on DBLDOS (tm) diskettes (Double density operating systems only) (MOD I only)

DBLFIX [:]drivespec

This utility will modify DBLDOS (tm) data and system diskettes so that they can be read and written to by MULTIDOS. It will not affect the proper operation of the diskette. To use this utility you must not have the write-protect notch covered on the diskette to be fixed.

To use the utility place the diskette to be fixed into a disk drive other than drive 0. Enter the word 'DBLFIX' followed by a space and the drive number of the target diskette.

EXAMPLE: To fix a DBLDOS diskette in drive 1.

DBLFIX :1

The colon is optional. Once the diskette has been modified the message:

'Modification completed'

will appear. If a data diskette has been previously fixed the message:

'Previously modified DBLDOS data diskette'

will appear. If a system diskette has been previously fixed the message:

'Previously modified DBLDOS system diskette'

will appear. If the target diskette is not a DBLDOS diskette the message:

'That is not a DBLDOS diskette'

will appear: If the target diskette is not formatted the message:

'Data record not found during read'

will be displayed. If an improper drivespec is used to specify the target diskette then the message:

'Please specify drive number'

will appear.

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 to LEVEL II Basic.

BASIC[ff[V]][,mm][,command]<ENTER>

This command will load **SUPERBASIC** into your system, from the Multidos ready prompt. A space is mandatory after the word "BASIC" if any of the optional parameters are to be specified.

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

If you want to use RANDOM disk I/O (input/output), and a record length other than 256, then key–in a V immediately following the character(s) that specify the number of file buffer(s).

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 (device control block), 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 ! Down load a BASIC program.

BASIC !<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 !". 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, variable review, and program pushing functions. 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 several 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 on the top row of the keyboard. Some of these 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 described with the "@" symbol preceding a character. This represents pressing the "@" key along with the 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 number.

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. 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 '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	X!	B#	Q!(F(G,Q))
K$	A!(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 for review 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, instead, you 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.

STACKING BASIC PROGRAMS

You may stack one or more programs in high memory while you work on or run antother program. Of course, this ability is limited by the amount of free memory space available. You can retrieve the stacked program(s) at will. There are five major commands for this function.

@- = Save the current BASIC program in high memory
@: = Recall the last saved program from memory
@8 = Append the last saved program to the current program
@9 = Append the next to last saved program to the current program
@Ø = Recall the next to last saved program

Pressing @- will save the resident program in high memory and will automatically adjust memory size to the beginning of the pushed program, thus protecting it from BASIC. Your current program will also be left available in BASIC RAM, if memory permits. Since memory size is adjusted, subsequent pushes can be made as desired. When a program is pushed, a graphic vertical bar will appear in the upper right hand corner of the video display. This bar indicates that a program has been pushed into high memory and a program is in BASIC memory for user execution or modification. If insufficient memory is available to push your program and also maintain it in BASIC RAM (i.e. trying to push a 30K program in a 48K machine), your program will be pushed and a "NEW" invoked. This condition will be indicated by a clear screen with a small graphic block in the upper right of the video display.

Pressing @: will pop or retrieve the last saved program from high memory. The resident program will be lost. Memory size will automatically be adjusted. If no saved program remains, the error message "NOTHING TO POP" will be displayed.

Pressing @Ø will retrieve the next to the last saved program from high memory. The resident program will be lost. If you want to switch the resident program with the last saved program first push the current program via the @- command and then retrieve the next to the last program via the @Ø command.

Pressing @8 will retrieve the last saved program and append it to the resident program.

Pressing @9 will retrieve the next to last saved program and append it to the resident program.

Line number sequence is mandatory for proper execution of the appending commands. The stacked program should have its lowest line number greater than the highest line number in the current program in BASIC ram. These commands append, they do not merge.

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 + Z     =    list last program line
```

The above single keystroke commands MUST be the first entry after the BASIC prompt, '>', appears.

Single letter commands:

```
                C = continue program execution
           D[.] = delete current line
           E[.] = edit current line
           L[.] = list current line
      Mln1,ln2 = move ln1 to ln2
        Mln1,. = move ln1 to current line
        M.,ln2 = move current line to ln2
      Nln1,ln2 = duplicate ln1 as ln2
        Nln1,. = duplicate ln1 as current line
        N.,ln2 = duplicate current line as ln2
             P = list page from current line
            Pn = list page from line "n"
             R = run program
R"filespec" = run filespec
```

The above commands must be followed by <ENTER>.

To list a line just type in the line number followed by <ENTER>. Variables are kept. To delete a line you must use D or DELETE, followed by the line number (optional), followed by <ENTER>.

The 'M' command will relocate ln1 to ln2, deleting ln1, and inserting it as ln2. If ln2 existed prior to the move it will be replaced by the ln2. The "." may be used to refer to the current line.

The 'N' command will duplicate ln1 as ln2 while leaving ln1 intact. If ln2 existed prior to this command, it will be replaced by the new line ln2. The "." may be used to refer to the current line.

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 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 &FFFD, 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. To remove remark statements use CMD"U".

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>". For a complete description of the DEBUG facility see the LIBRARY COMMAND section.

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 dynamically 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"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 (21018 D). DO NOT POKE THIS LOCATION!!!

DO NOT 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 15
```

CMD"P" Pack program lines

```
CMD"P"<ENTER>
```

This command will pack BASIC program lines together and maintain program logic. Upon invoking the command the prompt:

Maximum line length

will appear. Enter the maximum number of characters that are to appear in a program line. Enter any value between 0 and 65535. Entering a 0 will cause the packer to pack lines to a maximum of 65536 characters.

A second prompt will appear:

First

Enter the line number of the first line of the program that is to be packed. The final prompt will then appear:

Last

Enter the last program line that is to be packed. If the value entered for the "Last" is less than or equal to the "First" then you will be re-prompted for the first and last lines.

The packer will then analize the program for logic and check for some minor errors. If any errors are present in the program then they will be displayed on the video. They are:

SN in # = syntax error in line
OV in # = overflow error in line
UL # in # = undefined line number in line

where # is a line number

OV and UL usually occur on branching instructions (such as GOTO's, GOSUB's etc.) and denote either a line number that is too high or a reference is made to an undefined line, respectively.

If any errors occur the program will not be packed. Should no errors occur the program will be packed and spaces will also be removed.

The packer does not renumber the program. It packs lines using the following rules:

Lines which are referenced are not packed to a previous line.
Lines which contain IF statements are not packed to the following line.
Lines which contain REM and ' statements are not packed to the following line.

NOTE: In the BASIC program, that is to be packed, you must have an ending quotation mark for string constants (messages enclosed in quotes) or packing the program may destroy logic.

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 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",-I,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 (MODEL I only).

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 (MODEL I only).

CMD"T"<ENTER>

This command disables the interrupts. This command is invoked automatically when CLOAD, CSAVE, or SYSTEM is entered in the command mode. Other cassette operations such as INPUT#-1, and PRINT#-1 MUST be preceded by this command.

CMD"U" Remove REMark statements

CMD"U"<ENTER>

This command removes REMark statements from a BASIC program in RAM. Both REM and ' type remarks are removed. If a line consists of only a remark statement then the entire line will be removed. If only part of a program line consists of a remark statement then only the part that is the remark text will be deleted from the line. After the remark statements are removed the renumber utility is called to check for missing line numbers. The program is not renumbered.

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 re-entry 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 - MODEL I, 4411H - MODEL III) as the upper memory limits and will fit into the remaining RAM. This command requires a minimum of 6074 free bytes to execute.

EXAMPLES:

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

(2) CMD"BACKUP"
BACKUP/CMD 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. (Model I only, Model III uses ROM code)

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, with the EDIT function, 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 encountered 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 LSB entry point in 408EH [16526 dec] and the MSB 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 POKEing a value, C9H [201 dec], into RAM location 41A9H [16809 dec]. This is the value for LEVEL II. (MULTIDOS Basic uses a C3H [295 dec], at this location, 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 (To 'find' keywords use REFERENCE).

Ftar<ENTER>

This command will find all occurrences of "tar" in your BASIC program. The display will indicate the line the "tar" is found on, and if more than one occurrences are on this line a "/" is printed followed by the number of occurrences. Spaces are recognized with this command.

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.

EXAMPLE:

F in <ENTER>

This will FIND all occurrences of " in " in the resident BASIC program.

GLOBAL EDITING (GE) - Mass editing BASIC program.

-<ENTER> (a minus sign followed by <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).
- Merge adjacent line numbers into one long line.
- Split 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 occurrences 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 occurrences 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 occurrences 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 occurrences 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 occurrences 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 occurrences 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", 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 be 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 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. 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 ";p#<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) :[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:

```
IIIII/U(nnnnn)  Line number IIIII is referenced in
                line number nnnnn but does not exist.
IIIII/S         Line number IIIII contains a syntax error.
IIIII/O         Line number IIIII 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. The ending line is not renumbered.

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.
If the ",R" option is specified, the files will remain open.

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. In 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 delimiter placed between "items".
  item = an expression to be evaluated and written to the diskette.
```

The "item" delimiter, "m", can be a semi-colon ";" or comma ",". The use of these delimiters 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 are 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.

PUT[#]buf[,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 record length other than 256 is used, (only possible using the V option at BASIC initilization), then MULTIDOS will wait until the I/O buffer is full before a write is executed;

i.e OPEN"R",1,"POKER/TXT",67.

To recap a RANDOM write to a file, the following steps are necessary:

(1) OPEN the file — use "R" or "D" (if it isn't already).
(2) FIELD the file buffer.
(3) Use MKI$, MKS$, MKD$ as necessary.
(4) LSET or RSET all data to the file buffer.
(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.

To recap a read from a RANDOM file, the following steps are necessary:

(1) OPEN the file (if it isn't already).
(2) FIELD the buffer.
(3) GET the record
(4) CVI, CVS, CVD as necessary.

NOTE: Using a str$ in an assignment, on the left side of an equals sign, de-allocates the field assignment to the particular str$.

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

MULTIPLE/MULTI-AUTO - a useful application.

A multiple AUTO command is an AUTO command which has several commands separated with a linefeed, <LF>, (down arrow).

A multi-AUTO command is an AUTO command which is a multiple DOS command. i.e. DIR↓LIB,FREE.

Multiple AUTO commands will display each command before it is executed, whereas a multi-AUTO command will display the multiple DOS command only once.

In a multiple/multi-AUTO command, if you want to go into **SUPERBASIC**, BASIC must be the last command.

In a multi-AUTO command a "DO" file will write over the DOS command buffer; therefore, can only be the last command in a multiple DOS command line.

In order to excute a "DO" file, then a COMMAND, then back to a "DO" file during power-up, use the multiple AUTO command.

EXAMPLE:

AUTO DO SPL<LF>
VFU<LF>
DO GEN<ENTER>

Upon power-up/re-boot file SPL/IDO will be executed with all key inputs from the file SPL/IDO. Then VFU/CMD will load and execute, getting key inputs from the user. After VFU/CMD has terminated, the file GEN/IDO will provide all key inputs until it reaches its EOF.

Another space saving feature of the multiple/multi-AUTO function, is its ability to execute several commands, which will save that minimum one granule "DO" file.

Let's say you want to load a serial driver (SERDVR/CMD) then run a BASIC program. You can use the multiple/multi-AUTO command to handle this 'command only' task without consuming an entire granule of disk space.

AUTO SERDVR/CMD<LF>
BASIC RUN"MYPROG/BAS"<ENTER>

Remember, the AUTO command can only contain 31 characters, which include spaces, commas, and linefeed characters. The above example has 32 characters. The last quote mark will be overwritten with the <ENTER> character. This will not produce an error in this example but would produce an error if the serial driver program's name was one byte longer.

Suggestion: Use short filenames.

PAGE 4400 **MULTIDOS** MODEL III (version 1.6) 9/28/1982

These addresses will be useful for assembly language programmers. All addresses will be described; however, not all will be useful for your particular application. Addresses particular to the MODEL I are noted. Same for the MODEL III.

LABEL	ADDRESS	LEN	ACCESSED	USED A	BC	DE	HL	FUNCTION
DOS	4400	3	JUMP	X	X	X	X	LOADS DOS1, DOS INTERPRETER
DATA	4403	2	LOAD/READ					LOADED BY DOS0 FOR DCB
EXEC	4405	3	JUMP	X	X	X	X	EXECUTE COMMAND @(HL)
REPDOS	4408	1	LOAD/READ					LOADED/READ FOR REPEAT
ERROR	4409	4	CALL(JUMP)	X				INTERROGATE ERROR IN A
DEBUG	440D	4	CALL					LOAD/EXECUTE DEBUG
IUS (MODEL I)	4410	3	CALL	X			X	INSERT USER INTERRUPT SERVICE, @ DE POINTER, INTO SLOT A
TOPMEM (MODEL III)	4411	2	READ/LOAD					TOPMEM ADDRESS
SUB Y (MODEL III)	4413	6M	CALL	X			X	LOADS HL WITH (HL)+1
DUS (MODEL I)	4413	3	CALL	X		X	X	DELETE USER INTERRUPT FROM SLOT A
SUB Z (MODEL III)	4414	5	CALL	X			X	LOADS HL WITH (HL)
IDS (MODEL I)	4416	3	CALL	X			X	INSERT DEFAULT SERVICE @ DE, (NO POINTER!)
ICAT (MODEL III)	4419	3	CALL	X				INTERNAL CATALOGER
DDS (MODEL I)	4419	3	CALL	X		X	X	DELETE DEFAULT SERVICE
FILK	441C	3	CALL	S			X	POSITIONS FILE @(HL) TO (DE)
VERB	441F	1	READ					2ND BYTE OF WRITE VECTOR
INIT	4420	3	CALL	E				INIT FILE, DE=DCB
GBYT (MODEL III)	4423	1	READ					GRAPHICS IF NON ZERO
(MODEL I)	4423	1	READ					NULLS AFTER LINE FEED
OPEN	4424	3	CALL	E				OPEN FILE, DE=DCB

LABEL	ADDRESS	LEN	ACCESSED	USED A	BC	DE	HL	FUNCTION
LAST	4427	1	READ/LOAD					CONFIG OF LAST READ DRV
CLOSE	4428	3	CALL	E				CLOSE FILE, DE=ODCB
DOING	442B	1	READ/LOAD					NON ZERO IF "DO" ACTIVE
KILL	442C	3	CALL	E				KILL FILE, DE=ODCB
(MODEL III)	442F	1						UNASSIGNED
(MODEL I)	442F	1	READ					CLOCK BYTE
LOAD	4430	3	CALL	E	B		X	LOAD FILE, DE=DCB
LRUN	4433	3	CALL	E	B			LOAD/RUN FILE, DE=DCB
READ	4436	3	CALL	E				READ FILE, DE=ODCB
WRITE	4439	3	CALL	E				WRITE FILE, DE=ODCB
VERF	443C	3	CALL	E				WRITE FILE/REREAD DE=ODCB
REWIND	443F	3	CALL	E				POSITION AT BEGINNING
POSN	4442	3	CALL	E				POSITION @ BC LOGICAL RECORD
BCKSPC	4445	3	CALL	E				BACKSPACE ONE LOGICAL RECORD
POSEOF	4448	3	CALL	E				POSITION TO END OF FILE
DEXT	444B	2	CALL					JUMP RELATIVE TO DEXT1
FUNC	444D	2	READ					OVERLAY CALL ADDRESS
FORCE	444F	2	READ					DISK I/O TEST ADDRESS
DEVID	4451	3	CALL	X			X	DIVIDE HL BY A, HL = QUOTIENT, A = REMAINDER
PARAM	4454	2	CALL					JUMP RELATIVE TO PARAM1
FIXDEN (MODEL I)	4456	14M	CALL	X				SET CONFIG TO 'A' DRIVE
FIXDT (MODEL I)	4459	11	CALL	X				SET CONFIG TO 'A'

LABEL	ADDRESS	LEN	ACCESSED	USED A	BC	DE	HL	FUNCTION
JKL (MODEL I)	4461 (Available at 01D9H	3 MODEL III)	CALL	X		X	X	DUMP SCREEN <80H
CEOF	4464	3	CALL	X	C		X	RETURNS EOF STATUS
VOD	4467	3	CALL	X		X	X	SENDS TO (401E) @ (HL) UNTIL 03 OR 0D
PRLIN	446A	3	CALL	X		X	X	SENDS TO (4026) @ (HL) UNTIL 03 OR 0D
DTF (MODEL III)	446D	3	CALL	X				RETURNS CONFIG PRESENT
TIME (MODEL I)	446D	3	CALL	X	X	X	X	TIME
STS (MODEL III)	4470	3	CALL	X				RETURNS CONFIG OF 'A' DRIVE
DATE (MODEL I)	4470	3	CALL	X	X	X	X	DATE
DEXT1	4473	3	CALL	X	X		X	ADDS EXTENSION @ (HL) TO (DE) IF NONE
PARAM1	4476	3	CALL	X	X	X	X	(DE) @ TABLE, (HL) @ PARA
DOSCAL	4479	3	CALL	X	X	X	X	EXECUTES COMMAND @ (HL)
ADRQ	447C	5	CALL	X				FORCES PRESENT CONFIG TO 'A'
SUQQ	4481	5	CALL	X				SET CONFIG TO (IX+6)
STQ	4486	13	CALL	X	X			C RETURNS 5 IF S-DEN/P-DEN, OR 6 IF D-DEN
LWAIT	4493	9M	CALL					OUTS (0F0H),A
WAIT (MODEL III)	4495	7	CALL					WAIT 179 'T' STATES
WAIT (MODEL I)	4496	7	CALL					WAIT 179 'T' STATES
FRM1	449C	3	CALL	X	X		X	SPOOL LINK TO FORMS
WAT (MODEL III)	449F	2	READ					ADDRESS OF FORMS/SPOOL

LABEL	ADDRESS	LEN	ACCESSED	USED A	BC	DE	HL	FUNCTION
FPOS	44A1	10	CALL	X			X	HL=EOF A=REL BYTE IN SECTOR
SEEK	44AB	30	CALL	X				FDC STEPS PORT F3 TO PORT F1
MOTON	44C9	3	CALL	X				SELECTS DRIVE 'C'
GRDUM	44CC	3	CALL	X		X	X	TRANSFERS DISPLAY TO (4026)
WAT (MODEL I)	44C7	2	READ					ADDRESS OF FORMS/SPOOL
ZZZZ	44CF	3	CALL	X		X	X	DISPLAYS PRESS"ENTER" WHEN SYSTEM...AND WAITS <ENTER>
IUS (MODEL III)	44D2	3	CALL	X			X	INSERT USER INTERRUPT SERVICE, @ DE POINTER, INTO SLOT A
DTF (MODEL I)	44D2	3	CALL	X				RETURNS CONFIG PRESENT
DUS (MODEL III)	44D5	3	CALL	X		X	X	DELETE USER INTERRUPT FROM SLOT A
STS (MODEL I)	44D5	3	CALL	X				RETURNS CONFIG OF 'A' DRIVE
IDS (MODEL III)	44D8	3	CALL				X	INSERT DEFAULT SERVICE @ DE (NO POINTER!)
DDS (MODEL III)	44DB	3	CALL			X	X	DELETE DEFAULT SERVICE
READV	44DE	3	CALL	E				READ SECTOR FOR PARITY C=DRV E=SEC D=TR
READS	44E1	3	CALL	E				READ SECTOR INTO (HL) C=DRV E=SEC D=TR HL=BUFF
WRITS	44E4	4	CALL	E				WRITE SECTOR FROM (HL) C=DRV E=SEC D=TR HL=BUFF
DIRRD	44E8	3	CALL	E				READ DATA(DIRECTORY)SECTOR C=DRV E=SEC D=TR HL=BUFF
DIRWT	44EB	3	CALL	E				WRITE DATA SECTOR C=DRV E=SEC D=TR HL=BUFF

LABEL	ADDRESS	LEN	ACCESSED	USED A	BC	DE	HL	FUNCTION
RDIRP	44EE	3	CALL	E			X	EXITS WITH HL=43X0
WDIRP	44F1	3	CALL	E			X	EXITS WITH HL=4300
USRF	44F4	3	CALL	E				SENDS TO FDC 'A' FUNCTION (DO NOT USE TO READ TRACK)
GETDT	44F7	3	CALL	X		D		LOAD D WITH DIRECTORY TRACK OF DRIVE C
MODE	44FA	3	CALL	E	?	?	?	CHECKS FOR UNWRITTEN CONTENTS FOR ONE BYTE WRITES. IF UNWRITTEN WRITES IT.
OPESA	44FD	3	CALL	E				SAVE REGISTERS WHEN (DE) @ OPEN DCB, EXITS WITH IX=DE.

This is the total access to the "NUCLEUS" (DOS/SYS) used by all of MULTIDOS overlays and utilities. None of these programs make a CALL, LOAD, JUMP, OR READ above 44FDH. However, there is a major scratchpad area in the 4200H page MOD III, and 4300H page MOD I.

NOTES:

4400H is a dead end jump.
4405H is a dead end jump.
4409H is dead end unless bit 7 is set in A.

None of these routines use the IX, IY, alternate accumulator nor alternate registers BC, DE, & HL.

Obviously, all registers (not alternates) are used by executing most Library commands.

X Where an "X" appears, the register pair is altered.
Where a single register appears, only that register is altered.

S Under FILK, A returns a zero if valid filespec, otherwise non-zero.

E Where "E" appears the accumulator has an error code if non-zero.

CAVEAT

VERB, GBYT, and DOING will create a disaster if altered by user. DO NOT ALTER!

In fact don't poke anything into MULTIDOS. We have noticed that several application programs poke DEBUG off when loaded, especially those made for the SHACK. This will probably cause MULTIDOS to reboot. If you run into this problem, let us know, and we will tell you where to zero out this superfluous code.

DRIVE TABLE

The numbers represent drive zero to drive three respectfully.

4280H to 4283H - MOD III, 4300H to 4303H - MOD I Current head position for drives other than present.

4284H to 4287H - MOD III, 4304H to 4307H - MOD I Current track for directory.

4288H to 428BH - MOD III, 4308H to 430BH - MOD I Current CONFIG for drives.

CONFIG bit pattern:

BIT	IF SET	IF NOT SET	REMARKS
7	DOUBLE DENSITY	SINGLE DENSITY	
6	DIVISION 2	NO DIVISION 2	TS=DE/1.8
5	TWO SIDE	SINGLE SIDE	
4			NOT ASSIGNED
3	DIVISION 1	NO DIVISION 1	TS=DE/1.8+1
2	DOUBLE STEP @ SEEK		READ 40 IN 80
1			SPEED
0			SPEED

NOTE: If bit 6 is set, bit 3 must be set to divide.

In reading a directory, MULTIDOS will automatically flip bits 7, 6, and 3 as required. However, in raw reading a sector (such as ZAP programs do) only bit 7 will be changed as necessary.

EXAMPLES

1. Let's change drive 2 from single density to DBLDOS (tm):

```
LD    A,2
CALL  STS
OR    88H
CALL  FIXDT
```

2. To determine what was just read successfully:

The CONFIG is in ram location 4427 (LAST)

```
LD    A,(LAST)
BIT   6,A
JR    NZ,NEWDOS 80
BIT   3,A
JR    NZ,DBLDOS
BIT   7,A
JR    NZ,DOUBLE DENSITY
(SINGLE!)
```

SYSTEM ROUTINES and pertinent addresses

These routines are useful for machine language programmers. All 4 character addresses are hexadecimal, and are the same for both versions of double density as well as single density.

The registers other then the AF, which are altered are indicated next to the routine's description, preceeded by a colon ":".

For file manipulation routines, the DE register pair contains the address of a 20H byte DCB (device control block). This will be indicated by DE=DCB next to the routines' descriptions.
The following are bytes, words, calls, jumps, or a group of bytes used by MULTIDOS. The addresses will be suffixed with "b", "w", "c", "j", and "g" respectively.

0013 c One byte read DE=DCB

This routine will return the byte read into the A register if the Z flag is set otherwise the A register contains the error code. When the EOF is reached, this routine will return a 1CH without the Z flag set. This is the expected exit from this routine when all data has been read.

001B c One byte write DE=DCB

This is a very useful routine for writing to a file using all current DOS's. In each case, each DOS will maintain the proper NEXT and EOF values. Since the user should know when the last byte has been written, the proper EOF state will be properly set by each DOS.

The A register contains the byte to be written. If the Z flag is set after this routine was called the write was successful, otherwise the A register contains the error code.

400F c ENTER Debugging Package

RST 30H vector. This is set to JP 440D at power-up/re-boot. A JP 440D will load and execute DOS5/SYS, DEBUG. Users who have other monitors, may insert their monitors entry point in (4010) to execute their monitor in lieu of DOS5/SYS. This is unique to MULTIDOS.

402D j DOS EXIT :All

This routine will terminate any calls and load the SP with the DOS stack location. All of MULTIDOS library commands & utilities exit to this address, error or no error. During a CMD"uuuuu" from SUPERBASIC, the location pointed to by this address is temporarily altered to return to SUPERBASIC, and all registers are maintained.

4030 j DOS EXIT :All

This is MULTIDOS's no hang exit. This is unique to MULTIDOS.

4040 - 4046 g (4216 - 421C MOD III)
The bytes are ticker, seconds, minutes, hours, year, day, and month respectively.

4049 w (4411 MOD III) TOPMEM address
This address contains the highest byte of memory MULTIDOS, MULTIDOS utilities and SUPERBASIC will use. Protected memory is one byte higher than this address. During SUPERBASIC's CMD"uuuuu" this address is temporarily altered to protect SUPERBASIC and any BASIC program in memory.

4400 j DOS EXIT

This is the address pointed to by 402D unless a CMD"uuuuu" is in process.

4405 j Execute COMMAND :BC, DE, HL

This routine will execute the COMMAND at the address contained in the HL register pair (text pointed to by HL). The length of this COMMAND is unlimited and should be terminated with a byte which has the value of 00 to 0D. Upon completion of the COMMAND a 402D exit will be envoked.

4409 c Display DOS ERROR

This routine will display the error message which corresponds with the byte in the A register bits 0 thru 5. If bit 6 is not set, a hex to decimal conversion of the error message is displayed via,
'DOS ERROR = XX'.

If bit 7 is not set a jump to 402D is invoked; otherwise, a return, with all registers maintained, occurs.

440D c Enter DEBUG

This routine will return to the instruction following a 440D call if the SP is not altered, other than calls encountered while debugging a user routine.

4410 c (44D2 MOD III) Insert a foreground task. :HL

This routine will insert a routine to be executed, whenever the interrupt service is activated. MULTIDOS has 12 available slots,on the MOD I, and 10 on the MOD III. Slots 0 to 7 (0 to 5 MOD III) are executed every 200 mS, and slots 8 to 11 (6 to 9 MOD III) are executed every 25 mS (33 1/3 mS MOD III). On entry, the A register contains the slot number, and DE contains an address which POINTS to the routine. When a user interrupt routine is entered, the interrupts have been disabled and MUST remain disabled. Do NOT use the video routine, 33H, on the MOD III – it enables the interrupts – brilliant!!. All registers except the IY register pair (IY is saved on the MOD III) are saved and the HL register contains the address of the start of this routine.

EXAMPLE:

```
                                ORG     5200H
5200H                LD         A,4
5202H                LD         DE,POINT
5205H                CALL       4410H
5208H              (return to user program)

                                ORG     0FFEBH
0FFEBH      WORK     LD         A,0
0FFEDH               CPL
0FFEEH               LD         (WORK+1),A
0FFF1H               OR         A
0FFF2H               LD         A,2AH
0FFF4H               LD         (3C3CH),A
0FFF7H               RET        Z
0FFF8H               LD         A,20H
0FFFAH               LD         (3C3CH),A
0FFFDH               RET
0FFFEH      POINT    DEFW       WORK
```

Every 200ms the routine at FFEB is executed.
(This routine flashes an asterisk 2.5 times a second)

4413 c (44D5 MOD III) Delete a foreground task. :HL,DE

This routine will delete a user task from the interrupt chain. The A register contains the slot number for the deletion.

4416 c (44D8 MOD III) Insert default foreground task. :HL

This is a default insert routine. The interrupt chain works on the principle of address pointers. If an insertion is made via a 4410 (44D2 MOD III) call, an address is inserted which contains an address that points to a user routine. Where there are no user insertions, the remaining slots contain an address which has an address that points to a RET instruction. A call to 4416 (44D8 MOD III) will change the address which points to a return instruction to the address in the DE register. The user routine does NOT need to have a pointer.

4419 c (44DB MOD III) Delete default foreground task. :DE,HL

This routine will set the default pointer to a RET instruction.

441C c Extract a filespec. :AF, BC, HL

This routine will transfer the text pointed to by HL to the location pointed to by DE, and terminates it with a 03H.

This routine converts the text pointed to by HL into the filespec syntax to the location pointed to by DE. All lower case letters are converted to uppercase.

EXAMPLE:

HL=>abcdefghij/PQ1.K478ABQWNO:3<CR>
CALL 441C
DE=>ABCDEFGH/PQ1.K478AABQW:3<03>

4420 c OPEN/INITIALIZE. DE=DCB (name)

On entry the HL points to a 256 byte buffer, then a call to 4424 is made. If the 4424 call returns zero (file already exists) this routine will exit. Otherwise, the file is created with the logical record length in B. After the file is created the file is opened. If no error occurs the carry flag is set.

4424 c OPEN. DE=DCB (name)

On entry the HL points to a 256 byte buffer. The logical record length is extracted from the directory entry of the filespec. This is a major difference in MULTIDOS.

4428 c	CLOSE	DE=DCB (open)
442C c	KILL	DE=DCB (open)
4430 c	LOAD	DE=DCB (name)
4433 c	LOAD then EXECUTE	DE=DCB (name)
4436 c	READ	DE=DCB (open)
4439 c	WRITE	DE=DCB (open)
443C c	VERIFY	DE=DCB (open)
443F c	POSITION at start of file	DE=DCB (open)
4442 c	POSITION at BC logical record	DE=DCB (open)
4445 c	BACKSPACE one logical record	DE=DCB (open)
4448 c	POSITION at the end of file	DE=DCB (open)

4467 c VIDLIN :DE,HL

This routine will send to the DCB at 401D, the string at the address contained in the HL until a terminator, 03H or 0DH, is encountered. If the terminator is an 0DH, then it is sent also. Note MULTIDOS does NOT maintain the HL register.

446A c PRTLIN :DE,HL

This routine is the same as 4467, except the DCB is 4025.

446D c GETTIM (3036 MOD III) :BC,DE,HL

This routine tranforms RAM time to HH:MM:SS format in the address contained in the HL.

4470 c GETDAT (3033 MOD III) :BC,DE,HL

This routine tranforms RAM date to MM/DD/YY format in the address contained in the HL.

4473 c (444B or 4473 MOD III) Add default extension :BC,DE,HL

4476 c (4454 or 4476 MOD III) Get parameter values :BC,DE,HL

This routine will convert both decimal and hex numbers. Hex numbers use the "X" convention. This is accomplished with the letter "X" followed by a hexadecimal number enclosed in single quotes. Thus the number 100 could be written as X'64'.

The following are MULTIDOS raw disk I/O calls.
When "COND1" appears, it represents the following condition:

```
reg     C = drive number (Modulo 4)
reg     D = track number
reg     E = sector
```

In the register used column, when you see A=error, if there is an error the Z flag will be reset and the accumulator, A, will contain the error code. However, if you perform a read, or read verify on a directory sector without a DIRECTORY type command, the A register should contain an error code of 6.

CALL	FUNCTION	REGISTERS REQUIRED	REGISTERS USED
44DE	READ VERIFY	COND1	A=error
44E1	READ SECTOR	COND1, and HL=buffer	A=error
44E4	WRITE SECTOR	COND1, and HL=buffer	A=error
44E8	READ DIRECTORY SECTOR	COND1, and HL=buffer	A=error
44EB	WRITE DIRECTORY SECTOR	COND1, and HL=buffer	A=error
44EE	READ DIRECTORY Position	C=drive, bits 5-7 B=position bits 0-3 B=dir sec+2	HL=42XX,A=error
44F1	WRITE DIRECTORY Position	C=drive, bits 5-7 B=position bits 0-3 B=dir sec+2	HL=4200,A=error
44F4	USR FUNCT	COND1, A=FDC code HL=buffer if req'd	A=error
44F7	GET DEFAULT DIRECTORY TRACK	C=drive	D=track E=unchanged

The balance of DOS/SYS, 4500H to 4CFFH contains the code to execute these function, as well as other routines, tables, pointers, and several I/O devices.

MULTIDOS SYSTEM FILES

DOS/SYS - 3 grans - 4300H to 4CFFH.
Resident file loaded during power-up/re-boot.

DOS0/SYS - 1 gran - 4E00H to 50FFH.
CMD"uuuuu" executor, granule allocation, and extended directory entries. This file must be on a system diskette.

DOS1/SYS - 1 gran - 4D00H to 51FFH.
DOS COMMAND interpreter, library commands: BLINK, BOOT, BREAK, CLEAR, CLOCK, CLS, DEAD, LIB, LOAD, and VERIFY. This file must be on a SYSTEM diskette.

DOS2/SYS - 1 gran - 4E00H to 51FFH.
INIT and OPEN. This file must be on a SYSTEM diskette

DOS3/SYS - 1 gran - 4E00H to 50FFH.
CLOSE and KILL. This file must be on a SYSTEM diskette.

DOS4/SYS - 1 gran - 4D10H to 51FFH.
DOS error messages. Contains code for FIND & CMD"U".

DOS5/SYS - 1 gran - 4D00H to 51FFH. DEBUG

DOS6/SYS - 4/5 grans (MODEL I), 4 grans (MODEL III) - 5200H to 67FFH.
Library executor.

DOS7/SYS - 1 gran - 4D00H to 51FFH.
Mighty Multi, CMD"C", CMD"Q" and BASIC's M & N functions.

DOS8/SYS - 4/5 grans (MODEL I), 4 grans (MODEL III) - 5200H to 67FFH.
Library executor: ATTRIB, CLRDSK, DATE, DDAM, DO, FORMS, HASH, KILL, PATCH, PROT, RENAME, RESTOR, SETCOM, TIME, and TOPMEM.

DOS9/SYS - 4/5 grans (MODEL I), 4 grans (MODEL III) - 5200H to 67FFH.
Library executor: HELP

You may delete any of the /SYS files desired via VFU/CMD's purge function. However, the following must remain on a system disk for proper file manipulation: DOS0/SYS, DOS1/SYS, DOS2/SYS, DOS3/SYS and DOS4/SYS.

You may delete DOS4/SYS if necessary. However, DOS errors will not be available.

If you want to execute files on a diskette that do not CLOSE or KILL in their execution then DOS3/SYS may be deleted.

SUPERBASIC overlays.

EDIT/SYS - 1 gran - 4D00H to 51FFH.
Performs BASIC global editing.

RENUM/SYS - 1 gran - 4D00H to 51FFH.
Performs BASIC renumbering.

CREF/SYS - 1 gran - 4D00H to 51FFH.
Performs BASIC cross reference.

ERROR/SYS - 1 gran - 4D00H to 51FFH.
Displays BASIC error messages.

These SUPERBASIC overlays may be deleted if desired.

SUPERBASIC error messages and ERR codes.

ERR	ERR/2+1	Message
100	51	Field organization exceeded the logical record length.
102	52	Internal error, use CMD"E" for specific.
104	53	The buffer number is not available or was used improperly.
106	54	Directory does not contain file.
108	55	Incorrect file mode.
110	56	File previously opened.
112	57	Disk read error, use CMD"E" for specific.
114	58	Disk write error, use CMD"E" for specific.
116	59	File already exists.
118	60	End of file encountered.
120	61	Drive not available.
122	62	Disk space full.
124	63	"EOF" reached before any characters read.
126	64	Bad "PUT" parameter.
128	65	Improper file name.
130	66	Access mode differs from OPEN mode.
132	67	File/Program delimiters are too far apart for buffer.
134	68	Directory space full.
136	69	The write protect notch is covered.
138	70	Incorrect password used to access file.
140	71	Full directory. File cannot be extended.
142	72	The file has not been opened.
144	73	Undefined function.

Dear MULTIDOSSER,

We had promised a newsletter in May of this year but couldn't get one going due to the enormous sales during that period. We have expanded internally and believe we will be able to inform you of pertinent information on a more timely basis.

MULTIDOS has had a major change in its MODEL I version (version 1.4), with the unfortunate inclusion of several errors. The following patches describe the error, the date it was corrected, and how to fix it. If the effective file is dated after the error date [as displayed via DIR(A)], the correction is on your diskette.

***** PATCH ØØ1 ***** Ø5/14/82 ***** MODEL I ***** M *

COPY/CMD inadvertently had a typo in the source code. We had EQUated TOPMEM as 4409H. TOPMEM should be 4Ø49H.

PATCH COPY/CMD (REC=3,BYTE=164) 74.64
PATCH COPY/CMD (REC=3,BYTE=168) 73.64

***** PATCH ØØ2 ***** Ø5/26/82 ***** MODEL I ***** M *

MULTIDOS Version 1.4 introduced a method to dump the screen contents including graphics to the printer with a call to the routine at 44CCH. This routine is available in the MODEL III also. In the MODEL I version of MULTIDOS, this routine branched to the wrong address. The following patch should be made ONLY to Version 1.4 with the file DOS/SYS dated PRIOR to Ø5/26/82.

PATCH DOS/SYS (REC=3,BYTE=2Ø8) 197 [Single & "P" density]
PATCH DOS/SYS (REC=21,BYTE=2Ø8) 197 [Double density]

An example in the use of this routine.

```
1Ø CLEAR
2Ø DEFUSRØ = &44CC
3Ø 'PROGRAM GOES HERE TO PUT YOUR SPECIALTY ON THE SCREEN
5ØØ  X  =  USRØ(Ø)'  THIS  WILL DUMP THE ENTIRE SCREEN INCLUDING
GRAPHICS
```

If your printer does not have graphics, then line 2Ø should be:
(MODEL III) 2Ø DEFUSRØ = &1D9. (MODEL I) 2Ø DEFFUSRØ = &4461

***** PATCH ØØ3 ***** Ø5/27/82 ***** MODEL I ***** M *

VFU/CMD would hang up continuously printing garbage if the "H" command was used.

PATCH VFU/CMD (REC=5,BYTE=214) 231.4

The next two patches are optional for Version 1.4/1.5 MODEL I, and Version 1.3 MODEL III.

***** PATCH ØØ4 ***** Ø6/Ø1/82 ***** MODEL I ***** O *

Optional patch to limit the directory command in MIGHTY MULTI to only display user type files.

PATCH DOS7/SYS (REC=1,BYTE=147) 126.23Ø.248.254.16.Ø

***** PATCH ØØ5 ***** Ø6/Ø1/82 ***** MODEL III ***** O *

Optional patch to limit the directory command in MIGHTY MULTI to only display user type files.

PATCH DOS7/SYS (REC=2,BYTE=16) 126.23Ø.248.254.16.Ø

There are several programs which have to be patched to properly function with MULTIDOS. As we obtain a copy of these programs, we will advise you on an individual basis, and compile the patches in the next newsletter.

***** PATCH ØØ6 ***** Ø7/23/82 ***** MODEL I ***** M *

PROFILE initializes by determining how many drives are on the system. The file INIT determines this by trying to write a file (AØ/TØØ) to each diskette starting with drive #3. If INIT does not find the drive available, it will lock out that drive. INIT determines this condition by the error code returned. TRSDOS will return a 1AH, 26D (Directory space full) error if the drive is not available. The earlier versions of MULTIDOS maintained this error method. However, Versions 1.4 and greater now return the proper error code -- Ø8H (Drive not avalable).

PATCH INIT (REC=Ø,BYTE=1Ø6) 8

***** PATCH ØØ7 ***** Ø8/Ø2/82 ***** MODEL I/III ***** M *

DLDIS, Jake Commander's/Instant Software labeling dissembler. This program use one byte transfer for disk I/O and also uses a logical record length of 1. The use of a LRL=1 somehow is uncorrectly assumed when a one byte transfer is used. The source code generated by this program cannot be read by MULTIDOS. The following PATCH to DLDIS will not effect its ability to function with other DOS'es, but will make it compatible with MULTIDOS.

PATCH DLDIS/CMD (REC=Ø,BYTE=254) Ø

TRS-80 Address Marks for 5 1/4" Mini Floppies

All of the popular operating systems for the TRS-80 read only one sector (record) at a time. However, multiple record or track reads may be performed by various ZAP programs.

In order to read a sector, the FDC (Floppy Disk Controller) is given the track and sector number to read, then the command to read the sector. Once the FDC receives the read command, it reads the track immediately under the R/W (read/write) head. If the ID (track, sector, and side) matches, the FDC transfers this information one byte at a time to the host computer. If an ID cannot be found, or the ID does not match, a record not found error is generated and the read command is terminated by the FDC. It is up to the host computer to detect this error and terminate and/or retry the sector read function.

After the sector is sucessfully read, the FDC generates the record type information - type of ADDRESS MARK.

The type of ADDRESS MARK is determined by the FDC itself. The 1771, single density FDC (in the MODEL I) has the capability of four types of ADDRESS MARKS:

ADDRESS MARK	byte on diskette
1. Data mark	ØFBH
2. User defined	ØFAH
3. User defined	ØF9H
4. Deleted data mark	ØF8H

However, the single/double density FDC's, 1791 (double density hardware modifications) and 1793 (MODEL III), can only generate two types of ADDRESS MARKS:

ADDRESS MARK	byte on diskette
1. Data mark	ØFBH
2. Deleted data mark	ØF8H

The 1791 is a 1793 with an inverted data bus. (Signal information exchange - not to be confused with ADDRESS MARKS). The hardware will properly interface this 'data bus' with the host computer.

The reason for the different ADDRESS MARK, is for each DOS to verify that it has found the directory before it does a read or write to the diskette. Therefore, all but TRSDOS MODEL III, use the a different ADDRESS MARK for the file sectors than the directory sectors. (TRSDOS MODEL III does not put the ADDRESS MARK on a file sector until it writes to the sector.)

When double density for the MODEL I came along, the ADDRESS MARK problem caused the necessity of keeping the 1771 FDC for compatibility with the current inventory of diskettes using the User defined, ØFAH, mark to detect a directory sector.

In switching between single and double density on the MODEL I, the two different FDC's are switched in and out. If the MODEL I original DOS had used the Deleted mark, ØF8H, for the directory sectors, then the double density hardware would not need the 1771 socket, nor the 1771 chip. (Like changing from 4K to 16K).

On the MODEL I, DOSPLUS and NEWDOS/8Ø use the ØFAH when writing to single density directory sectors; however, LDOS and MULTIDOS use the ØF8H when writing to single density directory sectors.

NOTE:
MULTIDOS can be "patched" to use the ØFAH mark.

Since the MODEL III will not recognize an ØFAH mark on a sector as a directory sector, a utility/command is provided with DOSPLUS, LDOS, MULTIDOS, and NEWDOS/8Ø to convert (Dictionary meaning "change the state of") the single density directory sector ADDRESS MARKS to ØF8H.

The second generation of double density DOS's, DOSPLUS, LDOS, MULTIDOS, and NEWDOS/8Ø, coded their respective disk I/O to recognize both the, User defined, ØFAH, and the Deleted mark, ØF8H, as directory sectors. The ØF8H mark would be put on these single density diskettes if they were truly converted by a MODEL III utility.

NOTE:
MODEL III TRSDOS has a CONVERT which does not alter the ADDRESS MARKS, but simply TRANSFERS files to a TRSDOS MODEL III diskette. MODEL III TRSDOS assumes the directory is on track 11H, 17d and doesn't care which type of mark is used. If a directory is not on 11H, 17d, MODEL III, TRSDOS will fail to transfer the files.

Since LDOS and MULTIDOS use the ØF8H mark on single density diskettes - MODEL I and MODEL III, diskettes created by these operating systems are 1ØØ%, convert-free, portable between the MODEL I and the MODEL III versions of these operating systems.

TRSDOS 2.3, NEWDOS/21, and ULTRADOS, used the Data mark, ØFBH, for the file sectors and the User defined, ØFAH, for the directory sectors. Disk I/O in TRSDOS, NEWDOS/21 and ULTRADOS can be 'ZAPPED' to recognize the ØF8H mark, otherwise they will read each sctor 1Ø times then exit with the CORRECT error code (ERROR = 6) to signify a directory sector.

ZAPS for TRSDOS, NEWDOS/21, and ULTRADOS
In each case we will change the following three bytes:
Hex FE 20 28 to CB 6F 20.

TRSDOS 2.3 -- @ 46B6H
Track 0, sector 7, byte 0B7H - File sector 2 of SYS0/SYS.F3GUM

NEWDOS/21 -- @ 46B8H
Track 0, sector 9, byte 70H - File sector 4 of SYS0/SYS.F3GUM

ULTRADOS 4.2 -- @ 46BBH
Track 0, sector 8, byte 0BBH - File sector 3 of SYS00/SYS

In addition TRSDOS, and NEWDOS/21 need Track 0, sector 0, byte 0DCH changed from 5CH to 1CH in order for these systems to boot up when directory sectors have been written to with the Deleted mark, 0F8H.

The MODEL I versions of DOS'es have a utility/command to convert the 0F8H marks to 0FAH marks. However, the MODEL I version of MULTIDOS, as distributed, has a command (DDAM - Version 1.4 up) to perform either 0FAH to 0F8H or 0F8H to 0FAH conversion.

Summary

Type of ADDRESS MARKS placed on diskette sectors when written to by the popular DOS's.

DOSPLUS and NEWDOS/80:

	MODEL I		MODEL III	
	SINGLE	DOUBLE	DOUBLE	SINGLE
File	0FBH	0FBH	0FBH	0FBH
Directory	0FAH	0F8H	0F8H	0F8H

LDOS and MULTIDOS

File	0FBH	0FBH	0FBH	0FBH
Directory	0F8H	0F8H	0F8H	0F8H

TRSDOS

File	0FBH	0FBH	0F8H	NA
Directory	0FAH	0F8H	0FBH	NA

DBLDOS

File	0FBH	0FBH	NA	NA
Directory	0FAH	0F8H	NA	NA

Logical Record Length LRL

Any file other than a BASIC RANDOM file should have a "sector length" logical record. (In fact, a BASIC program or command file should have its LRL equal to the file's length). In these cases, the LRL=Ø (this is interpreted as 256) which is the correct value. However, there are programmers who use the ROM's one byte read and write routines and also set the LRL to 1. This is not necessary! The "one byte" used in reading and writing is the quantity of bytes (one in this case) placed in the buffer or retrieved from the buffer with each call. This has no conneection with the LRL of the file. We will give you a routine to read or write to a file using any popular DOS available today. The labels START, FINISH, and WHERE respectively contain a two byte word indicating the RAM address for the start of the file to be written, the last byte to be written, or where the loaded file is to go. This routine cannot be used for command files. An explanation of command file structure will follow this routine.

However, there are unique MULTIDOS methods to read or write sectors directly to a diskette. These methods were briefly explaned in the technical section of the manual. Speaking of the manual. If you have any suggestions and/or comments about the manual, please feel free to drop us a line. All useful suggestyions will be placed in the next edition. Currently the manual is not a tutorial documention, and limited tutorial information is presented in it. However, if you feel a tutorial should be available, let us know and we will use future issues of the newsletter a a medium to communicate the infoormation to the users.

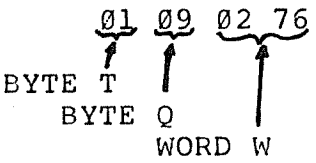
The following routine is slower than sector I/O, but is portable from one DOS to another. Sector I/O is not portable between DOS'es because the way they handle the NEXT byte in an open DCB. You can figure out the method used by each DOS by opening a known file on a diskette, read one record, then check the DCB. After you have determined the method used, configure your sector I/O routine to handle it!.

The nice thing about this routine is its ability to correctly update the EOF byte in the directory. Most sector I/O routines we have seen do not have the correct EOF byte set in the directory.

```
01000 ROMRD   DEFL    13H       ;ROM INPUT ONE BYTE FROM DEVICE
02000 ROMWT   DEFL    1BH       ;ROM OUTPUT ONE BYTE TO DEVICE
03000 DOS     DEFL    402DH     ;ENRTY TO 'DOS READY'
04000 ERROR   DEFL    4409H     ;ERROR INTERPRETER
05000 FILK    DEFL    441CH     ;MOVE FILE FROM @ (HL) TO
06000                           ;@ (DE), & CHECK FOR VALIDITY
07000 INIT    DEFL    4420H     ;INITIALIZE FILESPEC
08000 OPEN    DEFL    4424H     ;OPEN EXISTING FILESPEC
09000 CLOSE   DEFL    4428H     ;CLOSE FILE
10000 FBUF    DEFL    5200H     ;FILE DCB
11000 START   DEFL    FBUF+32H  ;CONTENTS OF 1st BYTE TO WRITE
12000 FINISH  DEFL    START+2   ;CONTENTS OF LAST BYTE TO WRITE
13000 WHERE   DEFL    FINISH+2  ;CONTENTS OF WHERE TO READ INTO
14000 MODE    DEFL    WHERE+2   ;NON-ZERO = READ
15000 IOBUF   DEFL    5300H     ;SECTOR BUFFER
16000 ;
17000 ;
19000         ORG     MODE+1
20000 DISK    LD      DE,FBUF   ;PUT DE @ DCB
21000         CALL    FILK      ;CHECK FOR VALID FILENAME
22000         JR      NZ,BADNAM ;NON-ZERO IF BAD NAME
23000         LD      HL,IOBUF  ;WHERE TO READ/WRITE SECTORS
24000         LD      B,0       ;LRL=256
25000         LD      A,(MODE)  ;WHAT TYPE
26000         OR      A
27000         JR      NZ,RDISK  ;NON-ZERO = READ
28000         CALL    INIT      ;INIT FILE OROVERWRITE IF THERE
29000         JR      NZ,UERR   ;IF NON-ZERO SHOW ERROR
30000         LD      HL,(START) ;FIRST BYTE TO WRITE
31000         LD      BC,(FINISH) ;LAST BYTE TO WRITE
32000 MORE    LD      A,(HL)    ;GET BYTE
33000         CALL    ROMWT     ;PUT IT INTO BUFFER
34000                           ;DOS WILL WRITE SECTOR
35000                           ;TO DISK WHEN FULL.
36000         JR      NZ,UERR   ;IF NON-ZERO SHOW ERROR
37000         PUSH    HL        ;SAVE POSITION
38000         SBC     HL,BC     ;ARE WE DONE?
39000         POP     HL        ;RESTORE POSTION
40000         INC     HL        ;INCREMENT ONE POSITION
41000         JR      NZ,MORE   ;IF NON-ZERO MORE TO WRITE
42000         JP      CLOSE     ;CLOSE FILE
43000 BADNAM  LD      A,13H     ;13H = IMPROPER FILE NAME
44000 UERR    OR      40H       ;MASK TO PROHIBIT
45000                           ;DISPLAYING 'DOS ERROR = XX'
46000         JP      ERROR     ;GET ERROR MESSAGE
47000 RDISK   CALL    OPEN      ;OPEN EXISTING FILE
48000         JR      NZ,UERR   ;IF NON-ZERO SHOW ERROR
49000         LD      HL,(WHERE) ;GET PLACE TO PUT IT
50000 GDISK   CALL    ROMRD     ;GET BYTE
51000         JR      NZ,EOF    ;IF END OR ERROR = NON-ZERO
52000         LD      (HL),A    ;PUT THE BYTE WHERE YOU WANT IT
53000         INC     HL        ;INCREMENT POSITION
54000         JR      GDISK     ;GET MORE
55000 EOF     CP      1CH       ;IF 1CH NATURAL EOF
56000         RET     Z         ;GO HOME IF OK
57000         JR      UERR      ;SHOW ERROR
58000         END     DISK
```

Command files are loaded with 'LOAD MARKs' interdispersed throughout the files object code. Each 'LOAD MARK' signifies the length of an 'object block' and where the object code is to be loaded, these object blocks may vary in length from 1 to 256 bytes.

How MULTIDOS interprets an object block.

Typical LOAD MARK 

BYTE T

Value	Meaning
ØØ	Skip over BYTE Q bytes of object code.
Ø1	Load, BYTE Q-2, bytes of object code at WORD W
Ø2	Invalid unless BYTE Q is also Ø2 which specifies end of object code.
Ø3-1F	skip over BYTE Q bytes of object code (same as ØØ)
2Ø-FF	invalid for object code - "LOAD FILE FORMAT ERROR".

The value of BYTE Q plus one is the length of the object block. However, the length of object code is the value of BYTE Q less two.

WORD W is the RAM location where BYTE Q less two bytes of object code is loaded.

In above typical LOAD MARK, 7 bytes will be loaded starting at location 76Ø2H.

After the object block has directed MULTIDOS to load BYTE Q less 2 bytes of object code, an addition object block is expected. However, there is a special LOAD MARK which signifies completed load and the transfer address.

A typical LOAD MARK signifying end of object code. Ø2 Ø2 86 9E

Once the preceding object code is loaded in RAM, and the user directed MULTIDOS to load and execute this file, then MULTIDOS will place 9E86H in the program counter and start the execution of the object code at this address.

If a file is loaded - LOAD FILE/OBJ - MULTIDOS will IGNORE the transfer address (9E86) and jump to location 4Ø2DH.

Suppose the source/object code looked like this

```
7000              ORG   7000H
7000  23 START    INC   HL
7001  7E          LD    A,(HL)
7002  FE4C        CP    4CH
7004  23          INC   HL
7005  CA3044      JP    Z,4430H
7008  FE52        CP    52H
700A  CA3344H     JP    Z,4433H
700D  C9          RET
7000              END   START
```

```
01 10 00 70 23 7E FE 4C 23 CA 30 44 FE 52 CA 33
44 C9 02 02 00 70
```

The first four bytes, 01 10 00 70, tells MULTIDOS that there is an object block 14D bytes (10H = 16D, 16D - 2D = 14D) long starting at 7000H. The next 14D bytes, 23 7E FE 4C 23 CA 30 44 FE 52 CA 33 44 C9, are the object code. The last four bytes, 02 02 00 70, tell MULTIDOS to begin execution at location 7000H.

Several programs (PENCIL, VISACALC, etc.), use relative sector in getting to the directory. This works fine in single density and "P" density, but requires patching to work with double density. If you are an experienced assembly language programmer or are willing to 'tinker' with a backup copy of a program, here is the MULTIDOS (version 1.4 up MODEL I or version 1.3 up MODEL III) way to get to a directory (double density MODEL I TRSDOS is not supported with this code).

Usually the code will have a buffer assigned to the area the directory sectors are to go. We will define this label as DIRBUF. Also the registers are usually saved before the varoius programs execute their directory function.

```
    GETDT   DEFL    44F7H
    DIRRD   DEFL    44E8H
    LAST    DEFL    4427H
    ERROR   DEFL    4409H
            LD      HL,DIRBUF
            CALL    GETDT
            LD      E,2         ;FOR FILE SECTORS
            CALL    DIRRD
            RET     NZ          ;RETURN IF ERROR
            LD      A,(LAST)
            XOR     80H
            AND     88H
            LD      A,8
            JR      NZ,SDEN
            ADD     A,A
SDEN        DEC     A
            LD      B,A         ;B NOW CONTAINS THE NUMBER OF
                                ;ADDITIONAL SECTORS TO READ
            XOR     A
            RET
```

MULTIDOS now has a ZAP program.

We now have a ZAP program - Easy Zap - which uses MULTIDOS' resident I/O for accessing diskettes. Easy Zap is available for only $15.00, which would include a diskette containing the latest version of MULTIDOS, and new documentation (no binder). Also included with this upgrade is a disk drive timer and a tape utility for transferring system files FROM tape to MULTIDOS.

This offer is ONLY available from C.E.C. Just let us know you want the upgrade, and we will send you the new diskette and documentation. There is no diskette credit available. Please keep you original diskette, we do no need you to return it.

Cosmopolitan Electronics Corporation
P.O. BOX 89
Plymouth, Mi. 48170

Please send me the upgrade to MULTIDOS at $15.00 plus $1.50 for shipping and handling. Foreign orders include $3.00 for shipping and handling. Michigan residents add 4% for sales tax.

Registraton number..... Density desired.....

Charge my: .....MASTER CARD .....VISA

CARD NUMBER.....

EXPIRATION DATE.....

SIGNATURE.....

CHECK ENCLOSED.....

Allow 3 weeks for personal checks to clear.

C.O.D. orders are cash or certified checks. Include an additional $1.50 for C.O.D. orders.

Send to.....

Address.....

City.....State.....ZIP.....

MULTIDOS USER NEWSLETTER

FEBRUARY 1983, COSMOPOLITAN ELECTRONICS CORPORATION

Happy New Year and welcome to 1983. In the new year we hope to bring you a hard disk operating system along with some other new products.

In this newsletter we bring you several zaps for application programs. There are only a few mandatory system zaps for version 1.6. Some of you will find your system already has some of these zaps applied.

We have moved since the last newsletter. Our new address and telephone numbers are:

Cosmopolitan Electronics Corporation
5700 Plymouth Road
Ann Arbor, MI 48105

Technical line & Michigan orders: (313) 668-6660
Toll-free orders only: 800 392-3785
Business hours: 9:30 am to 6:00 pm EST, Monday thru Friday

ABOUT ZAPS AND THE TECHNICAL INFORMATION LINE...

We depend on our users and other manufacturers of popular software to help supply zaps for application software to operate with MULTIDOS. We will research and implement zaps for the most requested programs. The amount of time it takes to research zaps does not allow us to supply the zaps for all application programs that users have requested. Most popular software, if correctly written, should have no problem operating with MULTIDOS. We say 'correctly written' meaning that any DOS level routines the program accesses uses the 4400H page. We have noticed that some application programs use CALLs right into the middle of DOS and that usually these CALLs are only applicable to TRSDOS.

The technical line can be used to answer technical questions relating to MULTIDOS. We have received some questions which are not easily answered on the phone. Questions which can be answered on the phone should relate to the general system and not to specific inter-system details or incompatibilities which may not be immediately addressable. Please re-read and double check the reference manual before calling.

Some common errors will occur because a diskette has gone bad. We have received diskettes with reported problems and the problems occur simply because the diskette has gone bad.

Program incompatibility problems are best solved by sending us a copy of the diskette (as gifts to C.E.C.) and an explanation of what the circumstances are. We cannot and will not return the diskette and will destroy its contents after studying the reported problem. If the problem is solved you will be notified by mail. We cannot guarantee that all problems can be solved or that all questions can or will be answered.

ABOUT SUBMITTING LETTERS....

Letters will be answered if you send an addressed stamped envelope along with the letter. We will not answer a letter without the envelope. Letters with numerous questions can take a great deal of time to answer and may not be supported. Some questions in the past have dealt with how to change the internal system and request specific patches. We do not support this except through the use of this newsletter. Some questions which are submitted are simply unanswerable or cannot be supported.

FROM OUR VIEWPOINT....

We at Cosmopolitan have developed MULTIDOS with several original (innovative) ideas. However, we have generally maintained the same syntax for most of the common commands. (e.g. APPEND, CLOCK, DEVICE, KILL, LOAD, PRINT etc.). Our BASIC still remains the most efficeint of all DOSes and will undoubtly remain that way. (Unless someone copies our code the way "they" did with MICROSOFT's). Primarily a software company, we listen to our users for enhancements and nice additional features. We cannot stand still! Our competitors are methodically incorporating MULTIDOS innovations into their operating systems.

EXAMPLES:

FUNCTION	Introduced
Repeat last DOS command	March 1981
Boot at high speed	Sept 1981
Alphabetized directory	Dec 1980
Backup only used granules	Sept 1981
Lockout Granules in track during format	Sept 1981
ASCII Find function for Basic	Sept 1981
Keyword cross-reference for Basic	March 1981
Delete an array function of Basic	March 1981
Chaining of Basic programs	July 1980

But we still have many, many more which are not yet implemented in the other DOSes. We have to keep enhancing and adding more functions to keep ahead of the other guys. After all, in many respects, we are the leaders and they are the followers. Believe it or not, this raises our egos to know that our ideas are being implemented into other operating systems. DOSPLUS II (Model II) has implemented the MULTIDOS AUTO ! and AUTO # functions. Being that we are totally a two man operation (one until September 1982) and one of us still holds a full time job, there can't be too much said for the other DOSes. In one of our competitors attempts to reduce the size of Basic, they use overlays; perhaps they will add an overlay for the keyboard driver to free more room! (Take a wild guess at which competitor? clue: they are west of the Mississippi.)

Anyway, we are open for suggestions for improvements to the DOS. PLEASE do not call in suggestions, instead send them in on a postcard.

Many of you have requested patches for the new Radio Shack word processor SuperScripsit. The patch for this is in this newsletter. In the process of making up the patches for this program we examined some of the code in its various files. The code is poorly written for a program that costs as much as it does. If some of you were planning to purchase the SuperScripsit program we highly recommend that you do not. Normally we do not take such a stand on any product, but felt this should be an exception.

SOME INFORMATION ON MULTIDOS ROUTINES....

ICAT - 4419H - Model III only

This call is in TRSDOS, LDOS, DOSPLUS and MULTIDOS, however, its implementation in MULTIDOS is unique. This routine will display a directory on the display or place the filenames into a user defined RAM area.

ENTRY	CONDITIONS
A	not used
B	function
C	drive number
DE	not used
HL	RAM area. Unused if not directed to RAM.

The user may determine if an error encountered should be displayed via a call to 4409H (DOS4/SYS) or returned in the accumulator.

Error in A Function	Error to display Function	Action
80	00	User files to display
81	01	User files to RAM
82	02	User & Invisible files to display
83	03	User & Invisible files to RAM
84	04	User, Invisible & System files to display
85	05	User, Invisible & System files to RAM

ON EXIT:
BC, DE, HL are unchanged.
Z = no error.
NZ = error. If 'Error to display function' is used you can still get the error code, if desired, by rotating the accumulator right and stripping off the high bits. (RRCA, AND 3FH)

KEYBOARD OVERRIDE CAPABILITIES DURING POWER-UP REBOOT...

Certain keys, when pressed during a reboot/powerup will override certain KEYBRD attributes.

MODEL I	MODEL III	EFFECT
<SHIFT>	Not Applicable	Override MULTIDOS keyboard
<SPACE>	Not Applicable	Override Lowercase Detection
<Up Arrow>	<Up Arrow>	Override cursor character
<Down Arrow>	Not Applicable	Override key repeat
<Left Arrow>	<Left Arrow>	Override blinking cursor & cursor character
<Right Arrow>	<SHIFT>	Override keyboard graphics
<CLEAR>	<CLEAR>	Override "CLEAR" off
Not Applicable	<BREAK>	Override "BREAK" off
Not Applicable	<Right Arrow>	Override "EPSON" setting

USING FORMAT Utility....

The Format utility has an undocumented feature in it. It allows you to specify parameters in the command line.

FORMAT [:]dA!-[diskname]

! = use all defaults
- precedes the diskname. If nothing follows then the default * DATA * will be used
A = absolute format. Format the diskette without checking for data already present

This is handy if you have want to format a diskette from a Basic program without user intervention.

Details on the 4000H-4500H pages of MULTIDOS....

Although we have detailed the 4400H page, listed below are some unique characteristics of **MULTIDOS.** Although the addresses may be different between the Model I and Model III, the functions are similar.

Access	MOD I	MOD III	Ref.	Function
S8	4015-401C	4015-401C		Keyboard DCB
S8	401D-4025	401D-4025		Video DCB
S8	4026-402C	4025-402C		Printer DCB
V	402D	402D		Exit from DOS function
V	4030	4030		Exit from DOS function with system mount prompt
B	404C	404C	SPOOL	BITS SET = 0=LF after CR 1=Serial 5=Data in buffer 6=Buffer full
B	404D	404D	DERR	Stores error code for CMD"E"
W	404E	404E		Scratch pad used during CMD"DOSCMD"
V	4050	4203	DESS	Panic exit. OK for DOS or BASIC
W	4053	429E	VSYS	Inner System module pointer
W	4053	42BF	CDRN	Accesable drive # scratchpad
W	4056	4265	MEMTOP	TOPMEM prior to 'DO' file active
W	4058	42B0	DOBUF	Pointer to active 'DO' buffer
W	405A	42BC	LRO	BIT 3 = Spooler in RAM
B	405C	42BE	TAW	Non-zero during initial motor on
S100H	4200	4300	BUFF	Disk I/O buffer
S4	4300-4303	4280-4283	DRTBL	Head location for drives 0-3
S4	4304-4307	4284-4287		DIRectory track for drives 0-3
S4	4308-430B	4288-428B		CONFIG byte for drives 0-3
B	430C	426D	SM0	
B	430D	426E	SM1	
B	430E	4215	MMB	Zero to load Mighty-Multi
B	430F	426A	OLC	Indicator of overlay prior to Mighty-Multi
B	4310	426B	OLB	Indicator of previous
B	4311	426C	OLA	Indicator of current overlay in RAM

Access	MOD I	MOD III	Ref.	Function
V	4312-4314	427D	BKVEC2	A C9 at first byte enables <BREAK> A C3 disables <BREAK>
V	4315-4317	4267	EXDB	
S40H	4318	4225	DOSBUF	
B	4423	428D	NULLS	

[B=byte, S=section, V=vector, W=word (two bytes)

Comments on MULTIDOS

Memory Usage:

This operating system has two means of establishing an area of RAM which will not normally be overlayed by any of the DOS commands or a SUPERBASIC program. However, the user inputs are treated differently.

TOPMEM sets the highest byte that the command files will use, where SuperBasic sets the lowest byte reserved. A TOPMEM of 31999 is the same as a MEMORY SIZE of 32000.

FIELDing buffers on RANDOM files:

If you would like to use variables to define the length of the fields in your FIELD statements, place the type declaration character behind the variable.

i.e. FIELD 1, D% AS A$, 256-D% AS B$

********** ZAPS **********

The following ZAPS are divided into three categories. Mandatory Zaps must be applied to correct bugs in the system. Optional Zaps may be applied to configure the system or to ease system operation when used with certain hardware (i.e. certain brand disk drives). Application Zaps are Zaps used to alter application programs (i.e. SuperScripsit) to work correctly with MULTIDOS.

*** Mandatory Zaps will apply to MULTIDOS version 1.6 only. ***

Some mandatory zaps will be applied by using the PATCH command of MULTIDOS, others by using the ZAP utility of version 1.6. You should become familiar with the operation of ZAP before applying any zaps using ZAP/CMD. DO NOT under any circumstances apply these ZAPs and PATCHes to your master system diskette or master application diskette (i.e. SuperScripsit master diskette).

About the ZAP utility....

Please read the MULTIDOS reference manual before applying any zaps using the ZAP utility of version 1.6. Basically, to apply zaps you will be using either of two options:

Diskette Sectors
or
File Sectors

When a zap specifies diskette sectors choose the diskette sectors option. ZAP will then ask you for

the drive number. Specify the drive number that the "target" diskette is in (the diskette that is to be modified). Next ZAP will ask you to specify the track number. Enter the track number that is listed for the particular zap you are applying. Finally, the sector number is asked for. Enter the sector number that is listed for the particular zap.

When a zap specifies file sectors choose the file sectors option. ZAP will then ask you for a filespec. Enter the listed filespec for the particular zap you are applying. If the target diskette is in a drive other than zero be sure to append the drive number to the filespec by using a colon and the drive number. ZAP will then search the target diskette for the filespec. If it finds the file then ZAP will ask you for the relative file sector. Enter the listed relative file sector for the particular zap.

When the sector is displayed...

Become familiar with the display of the sector. Refer to the reference manual, SYSTEM UTILITIES section – ZAP/CMD part IV. MODIFICATION AND UPDATE. (Most printings of the manual have this part located on page 48 or thereabouts.) The sector can be modified by pressing the M key. The cursor will be placed to relative byte 0 in the sector. Preceeding the cursor you will see 00 displayed and down the column you will see 10, 20, 30 etc. The top row is labeled 00 and the last row is labeled as F0. This column indicates the relative byte in the sector at the leftmost position in the row. Use the arrows to move the cursor around in the sector. Type in the modifications at the indicated bytes. Press the ENTER key to terminate modification and the ENTER again to update the sector.

Example:

```
 HEX 00   07FE 11ED 563A 0042 D607 207B E5FD E111  ....V:.B.. {....
     10   EA42 21E5 41F9 EB01 1600 EDB0 EB0E 0311  .B!.A...........
     20   4440 EDB0 3A81 44FE CD28 031B AF12 1100  D@..:.D..(......
 DRV 30   430E 06CD 7242 2100 3C4D 04EB EDB0 EB22  C...rB!.<M.....'
  0  40   2040 3E1F CD33 006A 2220 4011 0043 0E09   @>..3.j" @..C..
     50   CD72 42FD 3600 FFFD 3602 80FD 7001 4804  .rB.6...6...p.H.
     60   14CD 7242 0C14 7AE6 0F20 F621 004D E5C3  ..rB..z.. .!.M..
 TRK 70   1943 CD8D 42C8 21DB 427E CD33 0023 B720  .C..B.!.B~.3.#.
 000 80   F832 2040 CD49 003E 1FCD 3300 76D5 CD93  .2 @.I.>..3.v...
 00H 90   42D1 C8CD CC42 ED43 EE37 361B CDD5 42CB  B....B.C.76...B.
     A0   4620 FCC5 01EF 3736 88CD D542 CB46 28FC  F ....76...B.F(.
 SEC B0   CB4E 28FC 0A12 1CCB 4E20 F9CB 4E20 F5CB  .N(.....N ..N ..
 000 C0   4E20 F1CB 4620 F0C1 7EE6 BCC9 FD36 F501  N ..F ..~....6..
 00H D0   FDE5 E136 D03E 073D 20FD C91C 1F17 CA44  ...6.>.= ......D
     E0   4953 4B20 4552 524F 5200 496E 7365 7274  ISK ERROR.Insert
 STD F0   2053 5953 5445 4D20 3C65 6E74 6572 3E0D   SYSTEM <enter>.
```

↑ └This is relative byte F0 hex.

This column indicates the relative byte in the sector at the leftmost position in the row.

*** Apply ZAPs only to backups ***

NOTE: There is a non-zappable correction to BASIC and BBASIC on the Model III to allow continuation of a Basic program after recovering it with BASIC *. Whenever division is encountered in a recovered program Basic will bomb. To get around this do not continue the program. Save the program to disk. Go back to DOS and back to Basic (or BBasic) (don't use BASIC * or BBASIC * though) and then reload the program to correct the problem. This problem is on MULTIDOS Model III diskettes with DOS/SYS dated on and after 12/13/82 AND either BASIC (BBASIC/CMD or BASIC/CMD) dated before 01/27/83.

*** MANDATORY ZAPS ***

*** MZAP 001, MULTIDOS Version 1.6 Model III ***

This zap fixes the PATCH command on the Model III, therefore you must use the ZAP utility. DO NOT use the PATCH command - that is what you are fixing with this zap.

Use ZAP utility only.
Specify file sectors.
Filespec: DOS8/SYS
Dated on and only on: 01/10/83
Relative file sector: 16 decimal, 10 hex
Relative byte: C6 hex

```
Change:   20 9D 60 42  to:
          20 9E 60 42
```

*** MZAP 002, MULTIDOS Version 1.6 Model III ***

Use ZAP utility.
Specify file sectors option.
Filespec: DOS0/SYS
Dated before: 12/18/82
Relative file sector: 0
Relative byte: A1 hex

```
Change: 01 16 00 09 to:
        01 00 16 09
```

or you may use the PATCH command from DOS...

```
PATCH DOS0/SYS (REC=0,BYTE=162) 0;22
```

This corrects the system hang problem when there was less than 12,000 bytes of free memory and a CMD"uuuuu" command was executed from Basic. (Another case of transposition by the nontypist programmer. Why do you think MULTIDOS has short file names and many single key commands?)

*** MZAP 003, MULTIDOS Model III Version 1.6 ***

This will allow you to access a Basic program with protection level of EXECute via the update password.

Use ZAP utility.
Specify file sectors option.
Filespec: DOS2/SYS
Dated before: 12/18/82
Relative file sector: 1
Relative byte: 78 hex

```
Change: 28 32 79 to:
        28 1A 79
```

Or you may use the PATCH command from the DOS level...

```
PATCH DOS2/SYS (REC=1,BYTE=121) 26
```

*** MZAP 004, MULTIDOS Model III Version 1.6 ***

Use ZAP utility.

Filespec: DOS1/SYS
Dated before: 12/13/82
Specify diskette sectors option.
Track 18 decimal (12 hex), Sector 1
Relative byte: 80 hex

```
Change:  FE 1A 28 02  to:
         FE 1B 38 02
```

or you can use the PATCH command from the DOS level...

```
PATCH DOS1/SYS (REC=18,BYTE=129) 27;56
```

This corrects the problem of BBASIC clobbering a BASIC program in memory after a SAVE with 0 file buffers OPEN.

*** MZAP 005, ZAP/CMD Version 1.2 and 1.3 Model I and Model III ***

This zap corrects ZAP/CMD to pick the correct drive to fix a really bad directory if the question:

Is this a MULTIDOS double density diskette?

is asked by the ZAP utility. ZAP/CMD utilities which were in error would always select drive 1 when the above question was answered. Do not apply this zap to ZAP/CMD version 1.3a.

ZAP/CMD MODEL I and Model III, Version 1.2
Use PATCH command from DOS level.

```
PATCH ZAP/CMD (REC=2,BYTE=22) 217
PATCH ZAP/CMD (REC=2,BYTE=79) 217
```

or you may use ZAP to fix ZAP Version 1.2...

Filespec: ZAP/CMD
Relative file sector: 2
Relative byte: 14H

```
Change:  20 40 E5 DD E5 CD   to:
         20 40 D9 DD E5 CD
```

Filespec: ZAP/CMD
Relative file sector: 2
Relative byte: 4EH

```
Change: E1 E1 E6 5F  to:
        E1 D9 E6 5F
```

If you have ZAP Version 1.3 then....

Use PATCH command from DOS level.

```
PATCH ZAP/CMD (REC=2,BYTE=24) 217
PATCH ZAP/CMD (REC=2,BYTE=81) 217
```

or you may use ZAP to fix ZAP Version 1.3...

Filespec: ZAP/CMD
Relative file sector: 2
Relative byte: 16 hex

```
Change:  20 40 E5 DD E5 CD   to:
         20 40 D9 DD E5 CD
```

Filespec: ZAP/CMD
Relative file sector: 2
Relative byte: 50 hex

```
Change: E1 E1 E6 5F  to:
        E1 D9 E6 5F
```

** OPTIONAL ZAPS **

** OZAP 001, MULTIDOS Model I, Version 1.6, densities listed **

Use ZAP utility.
Specify diskette sectors option.

Single density: Track 0, Sector 9
Double density: Track 1, Sector 1
P density: Track 0, Sector 7
All densities: Relative byte DE hex

```
Optional change:  00 B7 20  to:
                  00 AF 20
```

This optional change allows for a 1 second delay on read. This may be necessary if you have disk drive(s) which have trouble coming up to speed quickly when selected.

** OZAP 002, **MULTIDOS** Model I, Version 1.6, Single density **

Use ZAP utility.
Specify Diskette Sectors.
Track 1, Sector 1
Relative byte: 18 hex

Optional change: 44 **EE 80** CD to:
44 **E6 37** CD

This optional change defeats automatic density recognition feature on a single density system. This may be necessary if you are experiencing 'Data record not found during read' errors. This is a "ghost" problem with newer expansion interfaces.

** OZAP 003, **MULTIDOS** Model I, Version 1.6, densities listed **

Use ZAP utility.
Specify diskette sectors.

If Single density: Track 0, Sector 7
If Double density: Track 0, Sector 9
If 'P' density: Track 0, Sector 5

Relative byte: 19 hex

Optional change: 3E **00** D3 **FF** to:
3E **ub** D3 **up**

Where: **ub** = user defined byte 00 to FF hex
up = user defined port 00 to FF hex

This allows you to do an OUT byte '**ub**' to port '**up**'. You can use this to turn on high-speed modifications, inverse video etc., on system boot.

** OZAP 004, **MULTIDOS** Model I & Model III, Version 1.6, densities listed **

Use ZAP utility.
Specify diskette sectors.
All densities Model I, and Model III: Track 17, Sector 0
Relative byte: C4 hex

You may change this byte to any value between 0 and 1F hex. If your master diskette has a value of 4 at this location then this byte allows you to set the number of disk I/O tries before error to the value + 1. If your master diskette has a value of 5 at this location then this byte allows you to set the number of disk I/O tries before error to the value; in this case a 0 will cause 256 tries. You will have a 5 at this location if DOS/SYS is dated before 12/08/82 on the Model I. You will have a 5 at this location if DOS/SYS is dated before 12/13/82 on the Model III.

** OZAP 005, MULTIDOS Model I Version 1.6, densities listed. **

Use ZAP Utility.
Specify Diskette Sectors.

Single Density: Track 1, Sector 6
Double Density: Track 1, Sector 8
P Density: Track 1, Sector 14

All densities: Relative byte 6E hex

```
Change:  F1 A6 C8 21  to:
         F1 00 00 21
```

This optional zap corrects the lost key problem some users have been experiencing on the Model I with version 1.6. This zap eliminates key-debounce checking.

** OZAP 006, MULTIDOS Model I and Model III Mighty-Multi

Use PATCH command.

For the Model I:

```
PATCH DOS7/SYS (REC=1,BYTE=147) 126.230.248.254.16.0
```

For the Model III:

```
PATCH DOS7/SYS (REC=2,BYTE=16) 126.230.248.254.16.0
```

This optional patch limits the directory command of MIGHTY-MULTI to only display user type files.

** OZAP 007, MULTIDOS Model I & Model III **

Since November 1981, thats right 1981 NOT 1982, MULTIDOS has had the capability of accepting numbers as the first character of a filename, graphics characters anywhere in a filename, and certain other ASCII characters in a file name. We had intended to include this as a KEYBRD parameter in Version 1.6 but DOS6/SYS was filled up! Instead of completely remapping MULTIDOS (Single density & "P" density Model I, has directory space limitations), or offering more on a MODEL III than a MODEL I (we may do something like this anyway!) we elected to use a byte on the GAT sector for this funciton.

Since we have enchanced DOS6/SYS to write out and declare the number of DISK I/O error tries and DOS/SYS to recognize them on power-up/reboot we used this byte to share the enchanced filename capability simply because the DISK I/O tries is limited to 5 bits. The relative byte is C4H. The bits pattern and its effect is shown below:

```
BIT 7   if set   does NOT convert case; permits graphics > ? @
BIT 6   if set   allows numbers as first character
BIT 5   if set   allows use of ! " # $ % & <
```

* APPLICATION ZAPS *

* AZAP 001, SuperScripsit, Model I and Model III *

Use PATCH command.

Warning: Apply these ZAPs (001, 002, and 003) to a **MULTIDOS** SuperScripsit disk only (a Multidos diskette with SuperScripsit files copied over to it). DO NOT use the TRSDOS diskette.

This set of patches allows **MULTIDOS** with SuperScripsit to read the correct free diskette space.

```
PATCH SCRIPSIT/CMD (REC=9,BYTE=7)  79.205.247.68.33.0.66.93.205
PATCH SCRIPSIT/CMD (REC=9,BYTE=16) 232.68.192.58.39.68.203.95
PATCH SCRIPSIT/CMD (REC=9,BYTE=24) 32.12.46.192.43.52.40.252
PATCH SCRIPSIT/CMD (REC=9,BYTE=32) 125.214.95.46
PATCH SCRIPSIT/CMD (REC=9,BYTE=40) 204.119.70.107.85.126.31.56
PATCH SCRIPSIT/CMD (REC=9,BYTE=48) 1.19.31.56.1.19.31.56.1.19
PATCH SCRIPSIT/CMD (REC=9,BYTE=58) 35.16.240.98.107.41.41.25.58
PATCH SCRIPSIT/CMD (REC=9,BYTE=67) 39.68.238.128.230.136.32.1
PATCH SCRIPSIT/CMD (REC=9,BYTE=75) 25.124.183.32.2.125.33.62
PATCH SCRIPSIT/CMD (REC=9,BYTE=83) 255.50.34.126.175.201
```

Shown below is relative file sector 9 of SCRIPSIT/CMD with the ZAP applied. Users of 1.6 may find it easier to use the ZAP utility to apply this zap. Areas of the sector which need changes are boxed in. Good luck !

```
S HEX00 221F 7EC9 3AD9 AB4F CDF7 4421 0043 5DCD ".~.:..O..D!.C].
C    10 E844 C03A 2744 CB5F C00C 2EC0 2B34 28FC .D.:'D._ ....+4(.
R DR 20 7DD6 5F2E 0102 005B CC77 466B 557E 1F38 }._....[.wFkU~.8
I  1 30 0113 1F38 0113 1F38 0113 2310 F062 6B29 ...8...8..#..bk)
P    40 2919 3A27 44EE 80E6 8820 0119 7CB7 2002 ).:'D.... ..|. .
S TR 50 7D21 3EFF 3222 7EAF C900 00E5 D5C5 DDE5 }!>.2"~.........
I 28 60 E101 0500 ED5B 137E AFCD 8667 ED53 137E .....[.~...g.S.~
T 1C 70 2A03 7E09 2203 7EC1 D1E1 C9E5 D5C5 21D3 *.~."."~.......!.
/    80 A922 497E 2137 AC22 157E EB21 705B CD1C ."I~!7."."~.!p[..
C SE 90 44C1 D1E1 CD94 57C9 5359 5354 454D 2F43 D.....W.SYSTEM/C
M 15 A0 544C 0DD5 C5E5 DD7E 00CD 1957 CD9D 57C4 TL.....~...W..W.
D 0F B0 1968 D1D5 21D4 C1CD E865 200D 11AB BA2A .h..!....e ....*
     C0 497E 2323 0105 00ED B0DD 6E01 DD66 0201 I~##......n..f..
FILE D0 F900 2228 7EB7 ED42 3004 ED4B 287E 2A49 .."(~..B0..K(~*I
0009 E0 7E11 0700 19D1 CDCC 5BC4 1968 0105 00DD ~.......[..h....
009H F0 09C1 D1C9 D5D1 C5ED B0C1 D52A 287E B7ED ..........*(~..
```

WARNING: It has been reported that SuperScripsit goes "out to lunch" on occasion, particularly when trying to go into the insert mode. One user complained that when attempting to go into the insert mode the program exited to DOS and did not close the file! If this happens blame Radio Shack, not us.

* AZAP 002, SuperScripsit for the Model III *

Use PATCH command.

PATCH SCR17/CTL (REC=2,BYTE=70) 6;0;230;15;79

PATCH (SCR17/CTL) allows SuperScripsit to read a directory when used with MULTIDOS Model III.

* AZAP 003, SuperScripsit Model I *

Use ZAP utility.

If you do not have MULTIDOS Version 1.6 you will not be able to correct this minor problem in SuperScripsit. It corrects the LRL on the ERRORS/CTL file. All you will get is bad error messages in SuperScripsit by not applying this ZAP.

NOTE: This zap is not necessary on the Model III version of SuperScripsit.

Specify diskette sectors.
Track 17, Sector 2.
The display will show a directory file sector. Example is shown below:

```
     HEX 00  5F0C 6A00 0044 4F53 2020 2020 2053 5953  _.j..DOS      SYS
         10  FFFF FFFF 1200 0002 FFFF FFFF FFFF FFFF  ................
         20  1001 0A21 0044 5732 2020 2020 2043 544C  ...!.DW2      CTL
     DRV 30  9642 9642 0500 0720 FFFF FFFF FFFF FFFF  .B.B... ........
      1  40  100F 0811 0044 5750 3431 3020 2043 544C  .....DWP410   CTL
         50  9642 9642 0500 0740 FFFF FFFF FFFF FFFF  .B.B...@........
         60  1001 0A40 4045 5252 4F52 5320 2043 544C  ...@@ERRORS   CTL <--
     TRK 70  9642 9642 0900 0801 FFFF FFFF FFFF FFFF  .B.B............
     017 80  1001 0A00 0048 454C 5020 2020 2043 544C  .....HELP     CTL
     11H 90  9642 9642 1200 0842 FFFF FFFF FFFF FFFF  .B.B...B........
         A0  160C FA56 0056 4655 2020 2020 2043 4D44  ...V.VFU      CMD
     SEC B0  FFFF 9642 1100 0A02 FFFF FFFF FFFF FFFF  ...B............
     003 C0  1003 0A00 004C 4543 5455 5245 5320 2020  .....LECTURES
     03H D0  9642 9642 0D00 0940 0B41 FFFF FFFF FFFF  .B.B...@.A......
         E0  1001 0A00 004C 5034 2020 2020 2043 544C  .....LP4      CTL
     RPT F0  9642 9642 0600 0C20 FFFF FFFF FFFF FFFF  .B.B... ........
```

Each group of 32 bytes is a directory entry. The first two rows are for the first entry. The third and fourth for the second file and so on. Notice the filenames located at the right end of every other row.

On the video look for the filename: ERRORS CTL

If it does not appear in this sector then use the right-arrow key to bump ZAP to the next sector. Continue the search until you find ERRORS CTL. If you bump up to track 18 then you did not copy ERRORS CTL over to the disk or you overlooked the entry.

Press the M key.

Move the cursor (using the down-arrow key) to the line that contains the ERRORS CTL entry. Move the cursor to the right using the right-arrow key until it is placed on the fourth relative byte in the

row. Make sure it is placed at the fourth relative byte. (The leftmost byte is considered as relative byte 0 for the row.)

Type in: 40
Press the <ENTER> key twice to update the sector.

This patches the LRL (logical record length) for the file ERRORS/CTL. Correct error messages will be returned from inside SuperScripsit. The Model III version of SuperScripsit has the LRL byte set to the correct value.

* AZAP 004, PROFILE Model I Version 3.1 *

Use PATCH command.

```
PATCH INIT (REC=0,BYTE=106) 8
```

Corrects "Drive not available" message from being returned.

* AZAP 005, PROFILE Model III Version 3.4 *

Use PATCH command.

```
PATCH INIT (REC=0,BYTE=102) 8
```

Corrects "Drive not available" error message being returned.

* AZAP 006, PROFILE Model I version 3.1, Model III version 3.4 *

Use PATCH command.

```
For the Model III
PATCH INIT (REC=3,BYTE=154) L0.H0.L1.H1.L2.H2.L3.H3

For the Model I
PATCH INIT (REC=3,BYTE=152) L0.H0.L1.H1.L2.H2.L3.H3
```

This patch allows the uses to gain maximum use from PROFILE.

L0.H0 etc. are a decimal value between 0 and 255. L0 is the LSB and H0 is the MSB for the value of the total number of physical records (sectors) available at drive 0. L1.H1 for drive 1, L2.H2 for drive 2, L3.H3 for drive 3.

The number of available records for each drive may be calculated by setting up a **MULTIDOS** system diskette with:

MULTIDOS minimum system consisting of:
DIR/SYS, DOS/SYS, DOS0/SYS, DOS1/SYS, DOS2/SYS, DOS3/SYS
You may add other files if you choose.

Add the following PROFILE files to the system disk:

ACCESS, FORMFILE, INIT, PRINT, PROFILE/CMD, SORT.
You may use data diskettes for drive(s) 1 to 3.

Do a FREE then use the following equation to determine the number of sectors available on the diskette at each drive:

Number of free grans on diskette multiplied by

6 if the diskette is formatted in double density
5 if the diskette is formatted in single or 'P' density

For example: Free grans = 80 on double density diskette

80 * 6 = 480

480 sectors (physical records) are available. Therefore

H = INT(480/256) = 1, L = ((480/256)-INT(480/256))*256 = 224

for the given drive.

For drive 0 you must subtract one gran from the returned value to leave room for INFOFILE (which is created when initializing a data base under PROFILE for the first time). You may wish to leave a few grans free on any or all of the diskettes. This patch will enable you to get the maximum amount of available data space when using PROFILE. Please do not call us with problems regarding this zap since they could be caused by multitude of things.

* AZAP 007, PROFILE Model III Version 3.4 *

Use PATCH command.

```
PATCH INIT (REC=0,BYTE=73) 66
PATCH INIT (REC=0,BYTE=78) 0
```

First patch changes the page size to 66 (what it should be).
Second patch sets the line counter to 0.

* AZAP 008, SuperScript (not SuperScripsit) Model I *

Use PATCH command.

```
PATCH SCRIPT/CMD (REC=44,BYTE=61) 247.68
PATCH SCRIPT/CMD (REC=44,BYTE=70) 232.68
PATCH SCRIPT/CMD (REC=44,BYTE=124) 232.68
```

If the above does not work with your version of SuperScript then copy over the original SCRIPT/CMD file and try the following PATCH:

```
PATCH SCRIPT/CMD (REC=41,BYTE=241) 247.68
PATCH SCRIPT/CMD (REC=41,BYTE=250) 232.68
PATCH SCRIPT/CMD (REC=42,BYTE=48) 232.68
```

We can't guarantee that either will work. This PATCH allows directory reads when using **MULTIDOS** and SuperScript.

* AZAP 009, ELECTRIC PENCIL, Model I *

Warning: this may not work on all releases of ELECTRIC PENCIL.

This patch will enable PENCIL (when operated under **MULTIDOS** V1.4 or greater) to read directories on anything **MULTIDOS** can read a directory on. Good Luck! We cannot guarantee that it will work with all releases of PENCIL.

Use PATCH command.

```
PATCH PENCIL/CMD (REC=0,BYTE=98) 0.0.0
PATCH PENCIL/CMD (REC=3,BYTE=253) 209.19.19.26.183.40.10.254
PATCH PENCIL/CMD (REC=4,BYTE=5) 52.48.247.254.48.56.243.230
PATCH PENCIL/CMD (REC=4,BYTE=13) 3.79.33
PATCH PENCIL/CMD (REC=4,BYTE=20) 0.60.34.199.88.46.13.34.201
PATCH PENCIL/CMD (REC=4,BYTE=29) 88.46.26.34.203.88.33.0.82
PATCH PENCIL/CMD (REC=4,BYTE=38) 30.2.205.247.68.205.232.68
PATCH PENCIL/CMD (REC=4,BYTE=46) 58.39.68.238.128.230.136.62
PATCH PENCIL/CMD (REC=4,BYTE=54) 8.32.1.135.71.205.232.68.126
PATCH PENCIL/CMD (REC=4,BYTE=63) 230.248.254.16.204.126.87
PATCH PENCIL/CMD (REC=4,BYTE=70) 62.32.133.111.32.242.28.16
PATCH PENCIL/CMD (REC=4,BYTE=78) 236.205.121.101.254.13.56
PATCH PENCIL/CMD (REC=4,BYTE=85) 249.195.214.97.197.213.229
PATCH PENCIL/CMD (REC=4,BYTE=92) 205.152.87.225.209.193.201
PATCH PENCIL/CMD (REC=4,BYTE=99) 0.0.0.0.0.0.0.0
```

Or if you choose to use the ZAP utility the zapped relative file sectors are shown below. The boxed in areas show where the changes are made.....

```
  L TR 50 22E4 6621 9054 22E5 662A 1640 2230 5321 ".f!.T".f*.@"0S!
  / 00 60 0A54 0000 0021 FD59 11CD 6B01 0300 EDB0 .T...!.Y..k.....
  C 00 70 2100 5A11 ED6B 0103 00ED B021 035A 119A !.Z..k.....!.Z..
  M    80 5D01 0300 EDB0 2109 5A11 B568 0103 00ED ].....!.Z..h....
  D SE 90 B021 0D5A 1113 6601 0300 EDB0 3E20 322C .!.Z..f.....> 2,
    06 A0 6621 BD59 22B1 5C21 EB69 22DF 613E FE00 f!.Y".\!.i".a>..
    06 B0 0000 C36F 5CE6 2FFE 20C0 3E1E C9F3 C5E5 ...o\./. .>.....
       C0 F5C3 D06B E1C1 FBC9 F53E 20CD AA6B F1CD ...k.....> ..k..
  FILE D0 AA6B C3B8 683E 00C9 2A30 5322 1640 2100 .k..h>..*0S".@!.
→ 0000 E0 3C22 2040 C32D 403A 6964 FE00 2836 C5D5 <" @.-@:id..(6..
  000H F0 E5ED 4BF5 5911 3253 2137 66ED B0ED 4BF5 ..K.Y.2S!7f...K.
```

```
       C0 FE1E 282E 2179 59FE 2028 27FE 2128 2321 ..(.!yY. ('.!(#!
  FILE D0 8F59 FE23 281C 21AD 59CD 0A57 21A0 5918 .Y.#(.!.Y..W!.Y.
→ 0003 E0 11C6 0027 3603 ED6F 0F0F 0F0F 2B36 03ED ...'6..o....+6..
  003H F0 6FC9 22F3 59E1 C9FE 61D8 E65F C9D1 1313 o.".Y...a.._....
```

```
P HEX00 1AB7 280A FE34 30F7 FE30 38F3 E603 4F21 ..(..40..08...O!
E     10 0102 8959 003C 22C7 582E 0D22 C958 2E1A ...Y.<".X..".X..
N DR  20 22CB 5821 0052 1E02 CDF7 44CD E844 3A27 ".X!.R....D..D:'
C  1  30 44EE 80E6 883E 0820 0187 47CD E844 7EE6 D....>. ..G..D~.
I     40 F8FE 10CC 7E57 3E20 856F 20F2 1C10 ECCD ....~W> .o .....
L TR  50 7965 FE0C 38F9 C3D6 61C5 D5E5 CD98 57E1 ye..8...a.....W.
/ 00  60 D1C1 C900 0000 0000 0000 0000 0000 0000 ................
C 00  70 0000 00C5 D5E5 110D 0019 7EFE 5020 1123 ..........~.P .#
M     80 7EFE 4320 0B23 7EFE 4C20 05E1 D1C1 1804 ~.C .#~.L ......
D SE  90 E1D1 C1C9 1105 0019 C53E FF32 C458 CD3F .........>.2.X.?
   10 A0 5806 087E 23FE 20C4 6F58 10F7 180D 0603 X..~#. .oX......
   0A B0 7EFE 2028 06CD 5D58 2310 F5D5 E52A C758 ~. (..]X#....*.X
      C0 7CB5 200F 2AC9 5822 C758 2ACB 5822 C958 ¦. .*.X".X*.X".X
FILE  D0 2AC7 5801 0D00 EB21 2053 EDB0 2AC7 5811 *.X....! S..*.X.
→0004 E0 4000 197C FE40 3803 2100 0022 C758 E1D1 @..¦.@8.!..".X..
004H  F0 C1C9 C5D5 E5CD 1A65 0610 212A 3C11 EB69 .......e..!*<..i
```

* AZAP 010, SCRIPSIT/UC and SCRIPSIT/LC Model I *

Same patches apply to SCRIPSIT/LC. This fixes the EOF problem, lets SCRIPSIT use DOS TOPMEM address 4049H-404AH, enables interrupts so Mighty-Multi can be used and, allows SCRIPSIT's END command to exit to DOS (or BASIC).

Use PATCH command.

```
PATCH SCRIPSIT/UC (REC=11,BYTE=119) 58.185.124
PATCH SCRIPSIT/UC (REC=11,BYTE=252) 50.182.124.196.239.93.0.17
PATCH SCRIPSIT/UC (REC=0,BYTE=100) 42.73.64.0.0.0.0.0.0.0
PATCH SCRIPSIT/UC (REC=0,BYTE=196) 0
PATCH SCRIPSIT/UC (REC=19,BYTE=228) 195.45.64
PATCH SCRIPSIT/UC (REC=0,BYTE=68) 229
PATCH SCRIPSIT/UC (REC=0,BYTE=212) 229
PATCH SCRIPSIT/UC (REC=4,BYTE=51) 227
PATCH SCRIPSIT/UC (REC=7,BYTE=1) 227
PATCH SCRIPSIT/UC (REC=12,BYTE=0) 239.93.0
PATCH SCRIPSIT/UC (REC=12,BYTE=100) 227
PATCH SCRIPSIT/UC (REC=17,BYTE=83) 227
PATCH SCRIPSIT/UC (REC=28,BYTE=206) 227
PATCH SCRIPSIT/UC (REC=30,BYTE=238) 227
PATCH SCRIPSIT/UC (REC=40,BYTE=230) 227
```

Or if you choose to use the ZAP utility the changes are listed below. File sector 0 is displayed for your convenience. Changes are boxed in. All other changes to file sectors are listed but a full display of file sector is not shown.

```
S HEX00 0102 0052 183D 434F 5059 5249 4748 5420 ...R.=COPYRIGHT
C    10 3139 3739 2042 5920 5241 4449 4F20 5348 1979 BY RADIO SH
R DR 20 4143 4B20 4120 4449 5649 5349 4F4E 204F ACK A DIVISION O
I  1 30 4620 5441 4E44 5920 434F 5250 4F52 4154 F TANDY CORPORAT
P    40 494F 4E31 E541 3E0A 32E8 37AF 3234 7C32 ION1.A>.2.7.24¦2
S TR 50 327C 322F 7C32 7D7C 3256 6032 3A60 3263 2¦2/¦2}¦2V`2:`2c
I 01 60 7C32 627C 2A49 4000 0000 0000 0000 225D ¦2b¦*I@......."]
T 01 70 7C22 2D7C 2255 7CE5 DDE1 2162 7F22 437C ¦"-¦"U¦...!b"C¦
/    80 2253 7C22 577C 222B 7C36 00EB 3E1E 32B1 "S¦"W¦"+¦6..>.2.
U SE 90 7C32 327E 3273 7E3E FF32 D37E 3E8C 3261 ¦22~2s~>.2.~>.2a
C 12 A0 7FFD 212F 7C0E 3CFD 7135 FD36 078F FD36 .!/¦.<.q5.6...6
  0C B0 0805 218B 6F22 297C CD1D 6921 F657 CDC8 ..!.o")¦..i!.W..
     C0 6BCD 6157 00ED 5343 7CAF 3240 7C32 317C k.aW..SC¦.2@¦21¦
FILE D0 3263 7C31 E541 21C0 52E5 FD36 078F CDCC 2c¦1.A!.R..6....
0000 E0 5FF5 CDE9 6BF1 DA08 5632 3E7C ED53 437C _...k...V2>¦.SC¦
000H F0 2A2B 7C36 0021 9379 CDF0 58C3 766F 626B *+¦6.!.y..X.vobk
```

Shown above is relative file sector 0 of SCRIPSIT/UC.

Remaining changes....

```
Relative file sector: 4 decimal
Relative byte: 32 hex

Change: 31 FA 41 0C   to:
        31 E3 41 0C

Relative file sector: 7 decimal
Relative byte: 00 hex

Change: 31 FA 41 3A  to:
        31 E3 41 3A

Relative file sector: 11 decimal
Relative byte: 74 hex

Change: 7C 47 00 CD 6E 7A 4F 09  to:
        7C 47 00 3A B9 7C 4F 09

Relative file sector: 11 decimal
Relative byte: FA hex

Change: 79 B7 C4 EF 5D 79  to:
        79 B7 32 B6 7C C4

Relative file sector: 12 decimal
Relative byte: 00 hex

Change: 32 B9 7C 11  to:
        EF 5D 00 11
```

```
Relative file sector: 17 decimal
Relative byte: 52 hex

Change: 31 FA 41 2A  to:
        31 E3 41 2A

Relative file sector: 19 decimal
Relative byte: E4 hex

Change: C3 00 00 CD  to:
        C3 2D 40 CD

Relative file sector: 28 decimal
Relative byte: CC hex

Change: 6E 31 FA 41  to:
        6E 31 E3 41

Relative file sector: 30 decimal
Relative byte: EC hex

Change: 6F 31 FA 41  to:
        6F 31 E3 41

Relative file sector: 40 decimal
Relative byte: E4 hex

Change: C9 31 FA 41  to:
        C9 31 E3 41
```

* AZAP 011, RACET'S Infinite Basic, Model I *

Use PATCH command.

```
PATCH IBLOAD/CMD (REC=8,BYTE=22) 17.244.121.205.19.0.50.182.121
```

This patch corrects IBLOAD/CMD module to enable it to run under **MULTIDOS.**

* AZAP 012, Visicalc Model III TRSDOS Version 1.2 *

Use PATCH command.

```
PATCH VC/CMD (REC=0,BYTE=248) 125
```

This patch is mandatory to allow Visicalc to load the proper BREAK key enable/disable address.

* AZAP 013, DLDIS/CMD Model I/III *

Use PATCH command.

PATCH DLDIS/CMD (REC=0,BYTE=254) 0

This zap allows DLDIS to be compatible with MULTIDOS. This program uses one byte transfer for disk I/O and also uses a logical record length of 1. The use of a LRL=1 somehow is incorrectly assumed when a one byte transfer is used. The source code generated by this program cannot be read by MULTIDOS. The above PATCH to DLDIS will not affect its ability to function with other DOS's, but will make it compatible with MULTIDOS.

* AZAP 014, SCRIPSIT Model I to work on the Model III

To enable Scripsit Model I version to operate on the Model III first apply all ZAPs listed in AZAP 010 then apply the following zaps.

Use PATCH command.

PATCH SCRIPSIT/UC (REC=0,BYTE=100) 42.17.68.0.0.0.0.0.0.0
PATCH SCRIPSIT/UC (REC=15,BYTE=56) 31.140.71.122
PATCH SCRIPSIT/UC (REC=15,BYTE=64) 132.254.96.48.15.203.8.48
PATCH SCRIPSIT/UC (REC=15,BYTE=72) 51.87.253.126.78.183.32.45
PATCH SCRIPSIT/UC (REC=15,BYTE=80) 203.234.24.41.214

Or for you ZAP users here is relative file sector 15 with the zap applied. The changes are boxed in. (Don't forget to apply the PATCH listed above for record 0. We do not show that sector here to save space.)

```
S HEX00 C95F B920 0FFE 0120 0B3A 8038 FDAE 4E32 . _ . ... .:.8..N2
C    10 7D7C 1889 C501 EE02 CD86 61C1 0AA3 C8FD }|¦........a.....
R DR 20 750C 7A07 0707 570E 0179 A320 0514 CB01 u.z...W..y. ....
I  1 30 18F7 FD71 0D3A 8038 1F8C 477A 0102 0061 ...q.:.8..Gz...a
P    40 84FE 6030 0FCB 0830 3357 FD7E 4EB7 202D ..`0...03W.~N. -
S TR 50 CBEA 1829 D670 3016 C640 FE3C 3802 EE10 ...).p0..@.<8...
I 04 60 CB08 3018 FE30 2002 3E50 EE10 180E 07CB ..0..0 .>P......
T 04 70 0830 013C 2183 794F 0600 097E 573A 0138 .0.<!.yO...~W:.8
/    80 E601 7A28 22FE 9B20 043E 1B18 2CFE 9C20 ..z(".. .>..,..
U SE 90 043E 1C18 24FE 9D20 043E 1F18 1CFE 4138 .>..$.. .>....A8
C 15 A0 06FE 8030 02E6 9FFE 4138 0EFE 8030 0AFD ...0....A8...0..
  0F B0 344E FD35 4E20 02EE 2057 01AC 0DCD 8661 4N.5N .. W.....a
     C0 7AC9 0000 0004 0B78 B120 FBC9 C5D5 DDE5 z......x. ......
FILE D0 ED5B 537C CDEC 61CD E853 CDC9 61DD E1D1 .[S¦..a..S..a...
0015 E0 C1C9 CDFA 52CD 336B 3A35 7CF5 C5D5 DDE5 ....R.3k:5¦.....
00FH F0 DD2A 557C ED5B 577C 060E 2100 3CCD C961 .*U¦.[W¦..!.<..a
```

Diskette Revision Updates....

Some minor enhancements have been made to version 1.6. To receive the current revision of the system with any mandatory zaps applied and any minor enhancements we are offering a revision update for $7.50. The revision update is for owners of MULTIDOS version 1.6 only. Density changes (Model I) are not applicable to this offer. The revision includes any mandatory zaps and minor enhancements to the system. We require that you return your master system diskette. **For shipment please protect the diskette in heavy cardboard on both sides to avoid damage and mail it in a box.** Diskette Revision service does not include optional zaps, zaps to application programs, custom modifications, or upgrade to version 1.6 from previous versions. Your diskette will be revised and any personal or application programs/data that you may have saved to the master diskette will be erased. Please do not send letters with your diskette.

Minor enhancements include:

* Revision to VFU/CMD to allow cursor to move off last file after selecting it. This revision is not in effect for VFU/CMD dated before 12/31/82. Although one user suspected this as a bug, we had designed VFU not to remove the "+" symbol if the cursor is placed over a previous selection. If you pressed "Y" or "N" at the last file displayed the cursor would remain there but the selection you made is still valid. Place the cursor to a previous file to verify this. (Read the documentation!)

* Revision to DDT/CMD to display the maximum and minimum timed drive speed. This revision is not in effect for DDT/CMD dated before 02/07/83 (both models).

* Revision to VFU/CMD to avoid displaying DIR/SYS, DOS/SYS, and NoDOS/Ind. This revision is not in effect for VFU/CMD dated before 12/31/82.

* Revision to both Basics to allow PRINT TAB & LPRINT TAB 0 to 255. This revision costs 4 bytes of memory. Basics dated before 12/27/82 Model III or 12/24/82 Model I do not have this revision.

* Revision to ZAP so it will not display an extraordinary number of errors during a Fix Directory of a very bad directory track. This revision is not in effect for ZAP/CMD dated before 12/26/82 on the Model I or 12/28/82 on the Model III.

* Revision to DOS/SYS to set the number of disk I/O tries to the value on the GAT table relative byte C4 hex minus 1. Previously the number of tries was set equal to the value. (Please note that only the lower five bits may be used.) This revision is not in effect for DOS/SYS dated before 12/08/82.

* Revision to DOS6/SYS to allow CONFIG library command to set the number of disk I/O tries by specifying CONFIG (T=number of tries) where 0 < T < 33. The parameter only goes into effect after a reboot power-up. This revision is not in effect for DOS6/SYS dated before 12/31/82.

* Revision to DOS/SYS (Model III only) to allow disk reads without initial 1 second delay. This revision is not in effect for DOS/SYS (Model III only) dated before 12/13/82.

* Revision to DOS8/SYS (Model III only) to allow it to visually indicate to the user that a file has been RESTORed when using the RESTOR command. This revision is not in effect for DOS8/SYS (Model III only) dated before 12/12/82.

* Revision to ZAP/CMD to allow display of diskette sectors and fixing of directories on opposite side of a two-headed disk drive. This revision is not in effect for ZAP/CMD, both Models, dated before 02/07/83.

Non-zappable correction:

* Non-zappable correction to PACK/SYS to avoid rare-occurence of losing text of a Basic program when the LSB of a 'next line pointer' was 0. This correction is not in effect for PACK/SYS dated before 12/12/82.

* Non-zappable correction to BASIC and BBASIC on the Model III to allow continuation of a Basic program after recovering it with BASIC *. Whenever division is encountered in a recovered program Basic will bomb. To get around this do not continue the program. Save the program to disk. Go back to DOS and back to Basic (or BBasic) (don't use BASIC * or BBASIC * though) and then reload the program to correct the problem. This problem is on MULTIDOS Model III diskettes with DOS/SYS dated on and after 12/13/82 AND either BASIC (BBASIC/CMD or BASIC/CMD) dated before 01/27/83.

To check if your diskette has the mandatory zaps applied refer to the section on mandatory zaps in this newsletter.

To order a revision update please see the order form in this newsletter.

LAST CHANCE FOR UPGRADE to Version 1.6 for $15.00

The upgrade to version 1.6 is still available at $15.00 plus $1.50 shipping & handling (foreign orders $3.00). Michigan residents please include 4% for sales tax.

The upgrade includes a new set of documentation (no binder) and new diskette. Please do not send us your original diskette. There is no diskette return credit available. This offer is available only from C.E.C. Offer is available until March 31, 1983 and orders must be postmarked by that date.

The enhancements in version 1.6 include:

* ZAP/CMD which allows you to display/modify diskette/file sectors, memory and fix directories

* DDT/CMD, a Disk Drive rotation speed Timer

* MEM/CMD, a Memory Test utility

* TAPE/CMD, a Tape to Disk utility

* Two more features to Super & BOSS Basic which include a remark remover and program packer!

* Ability of BOSS Basic to save and retrieve Basic programs to/from high memory.

Please use this order form for ordering revision or upgrade.
Indicate items below.

( ) Please send me a REVISION at $7.50. My master system diskette is enclosed.

( ) Please send me an UPGRADE at $15.00 plus $1.50 shipping/handling
(foreign orders add $3.00)

For UPGRADE to Version 1.6 please indicate desired density.....
(NOTE: density change (Model I) is not available for REVISION)

Registration number..... (we must have this to fill your order)

Charge my: .....MASTER CARD .....VISA

CARD NUMBER.....

EXPIRATION DATE.....

SIGNATURE.....

CHECK ENCLOSED.....

Allow 3 weeks for personal checks to clear.

C.O.D. orders are cash or certified checks. Include an additional $1.50 for C.O.D. orders.

Send to:.....

Address.....

City.....State.....ZIP.....