TABLE OF CONTENTS

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

TABLE OF CONTENTS

TABLE OF CONTENTS

INTRODUCTION

MULTIDOS

MULTIDOS is the only TRS-80® disk operating system (DOS) developed that uses a similar format in communicating among the TRS-80® MODEL I, MODEL III, MODEL 4, and the LOBO MAX-80, i.e., *some* machine language programs and *most* BASIC programs will run under MULTIDOS in all four machines. Things that usually require modifications are: PEEK, POKE, USR, CMD, CLOAD, and CSAVE.

"TYPE" is used to classify the operating systems, and is determined by four factors:

1. The type of address marks used to identify the directory. U indicates USER DEFINED. D indicates DELETED, and R (reversed) indicates that the directory has normal address marks and the files have DELETED address marks.
2. The use of pseudo-logical tracks. P indicated pseudo-logical tracks (referred to as LUMPS by NEWDOS/80v2). T indicates true tracks.
3. Track zero formatted differently than the balance of the diskette. Z indicates the same density. X indicates a different density.
4. The lowest sector number on each track. 0 = Zero. 1 = One.

				MODEL I		MODEL III/4, MAX-80	
MODEL	SYSTEM	DENSITY	"TYPE"	READ	WRITE	READ	WRITE
I	TRSDOS®	SINGLE	UTZ0	OK	NOTE 1	NOTE 2	NOTE 2
I	NEWDOS	SINGLE	UTZ0	OK	NOTE 1	NOTE 2	NOTE 2
I	VTOS	SINGLE	UTZ0	OK	NOTE 1	NOTE 2	NOTE 2
I	ULTRADOS	SINGLE	UTZ0	OK	NOTE 1	NOTE 2	NOTE 2
I	NEWDOS/80v1	SINGLE	UTZ0	OK	NOTE 1	NOTE 2	NOTE 2
I	DOSPLUS	SINGLE	UTZ0	OK	OK	NOTE 2	NOTE 2
I	NEWDOS/80v2	SINGLE	UTZ0	OK	OK	NOTE 2	NOTE 2
I	LDOS	SINGLE	DTZ0	OK	OK	OK	OK
I	MULTIDOS"S"	SINGLE	DTZ0	OK	OK	OK	OK
I	DBLDOS	DOUBLE	DPX0	OK	OK	OK	OK
I	VTOS	DOUBLE	DPX0	OK	OK	OK	OK
I	NEWDOS/80v1	DOUBLE	DPX0	OK	OK	OK	OK
I	NEWDOS/80v2	DOUBLE	DPX0	OK	OK	OK	OK
I	MULTIDOS"P"	DOUBLE	DPX0	OK	OK	OK	OK
I	DOSPLUS	DOUBLE	DTX0	OK	OK	OK	OK
I	MULTIDOS"D"	DOUBLE	DTX0	OK	OK	OK	OK
I	LDOS	DOUBLE	DTZ0	OK	OK	OK	OK
I	TRSDOS®	DOUBLE	DTX1	***NO***	***NO***	***NO***	***NO***
III	TRSDOS®	DOUBLE	RTZ1	***VFU***	***NO***	***VFU***	***NO***
III	DOSPLUS	DOUBLE	DTZ0	OK	OK	OK	OK
III	LDOS	DOUBLE	DTZ0	OK	OK	OK	OK
III	NEWDOS/80v2	DOUBLE	DPZ0	OK	OK	OK	OK
III	MULTIDOS	DOUBLE	DTZ0	OK	OK	OK	OK
4	TRSDOS®	DOUBLE	DTZ0	OK	OK	OK	OK
4	DOSPLUS	DOUBLE	DTZ0	OK	OK	OK	OK
4	MULTIDOS	DOUBLE	DTZ0	OK	OK	OK	OK
MAX-80	LDOS	DOUBLE	DTZ0	OK	OK	OK	OK
MAX-80	MULTIDOS	DOUBLE	DTZ0	OK	OK	OK	OK

VFU = VFU/CMD (or FU/CMD) can copy a file from these types.
NOTE 1: These types will have their address marks changed from U to D.
NOTE 2: Requires CONVERT/CMD to alter the directory address marks.

INTRODUCTION

BACKING UP MULTIDOS

The first thing that must be done with your MULTIDOS master diskette is make a backup. A blank diskette without the write protect notch covered is required for this procedure.

WRITE PROTECT THE MULTIDOS MASTER DISKETTE by placing a write protect tab over the notch (if your diskette already has the write protect notch covered, leave it on).

Turn on the computer, insert the MULTIDOS diskette into drive 0, and press the reset button.

When the diskette boots, the MULTIDOS banner is displayed and you are presented with the date and time prompt. Press <ENTER> or input today's date and time if you wish.

Key in BACKUP, then press <ENTER>.

Answer: Which logical drive contains the source diskette?

with 0.

Answer: Which logical drive for the destination diskette?

with a drive number having the same track capacity, if two or more drives are available, otherwise answer with 0.

The next prompt is: <ENTER> when the source disk is in logical drive 0.

Press <ENTER>. At this point BACKUP/CMD examines the source diskette for track count and density, then displays the cylinder count and the density.

Answer: Physical cylinderss for the destination diskette (38 to 96)?

with <ENTER>. BACKUP/CMD formats the blank diskette, then duplicates the contents of the MULTIDOS master diskette.

If only one drive is used, BACKUP/CMD indicates when to insert the destination (blank) diskette and when to re-insert the source (MULTIDOS master) diskette.

During a one drive BACKUP, several diskette changes are required.

When BACKUP has completely duplicated the MULTIDOS master diskette,

Completed
Insert SYSTEM <ENTER>

is displayed. Press <ENTER>.

Place your MULTIDOS master diskette in a safe place, and create all future backups from the copy just produced.

MANUAL NOTATIONS

CAPITALS and punctuation must be keyed exactly as it appears in this manual.

A space represents at least one mandatory space. Single quotes enclose responses displayed on the video monitor.

Symbol	Meaning
<ENTER>	Press the "ENTER" key.
<BREAK>	Press the "BREAK" key.
<SPACE>	Press the "SPACE-BAR".
<LF>	Press the "↓" key, the linefeed character.
<*key*>	Press the key that is labeled *key*.
filespec	A valid MULTIDOS file having the general form: *filename/ext.password:drivespec*
drivespec	A particular logical disk drive number (0 to 7).
Diskette	Refers exclusively to a floppy diskette.
Disk	Refers to a floppy diskette, hard disk, or memory disk.
command	Represents the MULTIDOS command you want to execute. Typically, commands may be either upper or lower case letters.
(parameters)	One or more arguments separated by commas that may be needed by the command. Some commands have no parameters. Most parameters are optional.
[ ]	brackets [ ] around any word in a command line indicate that the word is optional. The "[" and "]" are not keyed in.
...	Ellipses represent the repetition of an optional parameter.
{ }	Braces enclose manual remarks.

The term A and/or B implies inclusive or, i.e., either A or B or both.

The term A or B implies exclusive or, i.e., either A or B but not both.

This manual is intended for all versions of MULTIDOS. Whenever you see: MODEL I, MODEL III, MAX-80, MODEL 4, and/or ESOTERIC, then the command and/or function is applicable to that particular version of MULTIDOS operating in the appropriate hardware or emulation.

MODEL I and MODEL III: <SHIFT·0> is used to toggle between CAPS-LOCK and upper/lower case.
MAX-80: <SHIFT·CTRL> is used to toggle between CAPS-LOCK and upper/lower case.
MODEL 4 and ESOTERIC: <CAPS> is used to toggle between CAPS-LOCK and upper/lower case.

COMMANDS

Whenever the command prompt, 'MULTIDOS', is displayed, you are in the (DOS) command mode and you may enter a LIBRARY command, or the name of a system utility, or the name of an application program.

EXAMPLE:

```
    DIR<ENTER>
```

The command is terminated with <ENTER>. The letter's case in the command is disregarded unless a control byte, SM0, has a specified bit set. Refer to the Entry points for MULTIDOS section for the function of the bits in this byte.

MULTIPLE COMMANDS

Multiple DOS commands are executed one at a time if they are separated by commas immediately behind the command(s).

EXAMPLE:

    DIR 1, BASIC<ENTER>

    Executes the LIBRARY command DIR then loads and executes SUPERBASIC.

REPEAT COMMANDS

MULTIDOS repeats the last DOS command if <ENTER> is the first response to the command prompt and a comma does not follow the command.

EXAMPLE:

    DIR :1<ENTER>

After the command prompt is displayed, pressing <ENTER> as the first keystroke again displays the directory of the disk in logical drive 1 (Diskettes may be changed between the <ENTER>s).

This feature can be useful if a lengthy command is used to write to a write protected diskette. Simply write enable the diskette and press <ENTER> to repeat the aborted command.

A comma behind a command disables repeating the command with <ENTER>. <BREAK> and the LIBRARY command RESET also disable the last DOS command.

EXAMPLE:

    RESET,DIR 1<ENTER>

When this command is executed, RESET is performed; however, DIR 1 is ignored and the command RESET,DIR 1 is erased.

EDITING COMMANDS (MAX-80, MODEL 4, and ESOTERIC)

Limited editing of commands are possible with <CTRL·S>, <CTRL·L>, and the left and right arrow keys. <CTRL·S> is used to delete the character under the cursor. <CTRL·L> is used to toggle between the insert and overstrike modes. The cursor is positioned with the use of the left and right arrow keys. To see the present command, one character at a time , press <→>. To see the entire command, press <SHIFT·→>. Pressing <←> does not erase the command; however, the command is terminated where <ENTER> is pressed (repeat: "the command is terminated *where* <ENTER> is pressed.").

QUICK DIRECTORY

MULTIDOS provides an easy method to display a directory by pressing the numbers 0 to 7 as the first character on the command line. For the Model I and Model III, you must do this before any other key is pressed. If you have pressed any other key(s), you must press <BREAK> instead of <←> to position the cursor to the first position. For the MAX-80, Model 4, and ESOTERIC, you may use <←> to position the cursor to the first position.

Let's say you want to display the directory for drive 2. Simply press the <2> key.

If you have a two volume diskettes, side 1 can be displayed by pressing <SHIFT> number. e.g., for DIR :6' press <SHIFT·6>

MINIDOS

This feature enables you to interrupt a program while that program is executing and perform several selected commands before returning to the program without any changes in the program's state during this interruption.

MINIDOS is activated if both the following conditions are met:

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

To exit MINIDOS:

1. Press <BREAK> or
2. Key <!> and <ENTER>

Upon exit an extraneous : and/or ; may be present.

MINIDOS commands require one letter and do not require a space between the letter and any parameters that may follow. However, spaces may be keyed in after the command letter.

EXAMPLE:

D1<ENTER> is interpreted the same as D 1<ENTER>

There are six MINIDOS commands for the Model I, Model III, and ESOTERIC. There are seven MINIDOS commands for the MAX-80 and Model 4. The commands available are:

C = COPY - Duplicate a file from one mounted disk to another mounted disk.

C fileone[:d] [to] filetwo[:d]<ENTER>

NOTE: both filespecs are required.

D = DIR - Display the directory of the specified drivespec.

Dd, **DI**d, **DS**d, **D**:d, **DI**:d, or **DS**:d<ENTER>

d = 0 to 7 for the logical drive number.
If d is omitted, then zero is used.
I = display files with an invisible attribute.
S = display files with an invisible and/or system attribute.

K = KILL - Remove the specified filespec.

K filespec<ENTER>

L = LIST - List the specified filespec.

L filespec<ENTER>

The listing is suspended if <SPACE> is held down, or terminated if <BREAK> is pressed.

M = DEBUG - Enables DEBUG, except in command mode

M<ENTER>

This command enables DEBUG immediately; however, you cannot activate DEBUG until you exit MINIDOS.

P = PRINT - Send specified two digit decimal character to the printer.

Pnn<ENTER>

This command sends byte **nn** to the printer. Its main feature is to send control bytes, 0 to 31, to the printer without leaving the interrupted program in progress.

MAX-80 and MODEL 4

V = VIDEO - Flip the video format between 64X16 and 80X24.

V<ENTER>

This command clears the display then flips the video between the 64X16 format and the 80X24 format. If the present video format is 80X24, then V<ENTER> changes the video to 64X16. If the present video format is 64X16, then V<ENTER> changes the video to 80X24.

JKL Screen dump to printer, excluding graphic characters.

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

This command sends the contents of the display to the device defined in the printer DCB. Graphic blocks are converted to periods. If graphic blocks are not to be converted to periods, then the HJK command, described below, should be used.

HJK Screen dump to printer, including graphics characters.

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

This command is similar to the JKL command, except graphics blocks are sent to the printer instead of being converted to periods.

The ASCII Codes

DEC.	HEX	CHAR.	DEC.	HEX	CHAR.	DEC.	HEX	CHAR.	DEC.	HEX	CHAR.
0	00	NUL	32	20	SPACE	64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	±

FILESPECS

When specifying a disk file the term "filespec" is used in this manual. A filespec with all the options would look like this:

filename[/ext][.password][:drivespec]['']

filename is the name of the disk 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 and/or numeric, A-Z and/or 0-9. You cannot use TO as a MULTIDOS filename.

/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, and the remaining characters may be alphabetic and/or numeric. All file extensions, except /CMD, must be keyed in for proper identification of the file. More on this later.

.password is an optional password and consists of a period and at least one, but not more than eight characters. The first of these characters must be alphabetic and the remaining may be alphabetic and/or numeric.

:drivespec is an optional logical disk drive number, 0 to 7, preceded by a colon.

' is a drivespec modifier to access side 1 of a diskette when it is configured as two volume.

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 drivespec makes a file unique. The following are interpreted as different filespecs:

wonder:1 wonder:2 wonder:3

The following are filename extensions generally employed:

/ASC	A BASIC program stored in ASCII form.
/ASM	An assembly language source file.
/BAS	A BASIC program stored in compressed format.
/RIM	A file that is a direct copy of memory (RAM Image Memory).
/CMD	A machine language file executable directly from DOS.
/TXT	An ASCII text file (Typically sequential files).

To execute a file that ends with the /CMD extension, the /CMD extension is optional in the command mode. i.e., MULTIDOS inserts the extension /CMD if a filespec is entered without an extension.

EXAMPLE:

```
BACKUP<ENTER>
```

MULTIDOS adds the extension /CMD to BACKUP, then loads and executes the file BACKUP/CMD.

To execute a file without an extension, append just the /, and MULTIDOS loads and executes the file without adding the default extension /CMD.

EXAMPLE:

```
CREATE/<ENTER>
```

MULTIDOS loads and executes the file CREATE.

If a drivespec is not assigned, MULTIDOS searches all active drives, in ascending logical order, starting with logical drive 0 for the filespec entered. If a filespec is initialized (new file created) without a drivespec, MULTIDOS searches for the first active drive with the disk write enabled to initialize the file.

EXAMPLE:

```
PROG:2<ENTER>
```

MULTIDOS looks for the file PROG/CMD on logical drive two.

EXAMPLE:

```
PROG/<ENTER>
```

MULTIDOS searches the drives, in ascending logical order, then loads and executes the program PROG on the first drive PROG exists on. If PROG is on logical drives 3 and 4, the version on logical drive 3 is loaded and executed.

EXAMPLE:

```
PROG/RAM<ENTER>
```

MULTIDOS searches the drives, in ascending logical order, then loads and executes the program PROG/RAM on the first drive PROG/RAM exists on. If PROG/RAM is on logical drives 3 and 4, the version on drive 3 is loaded and executed. If you want to load and execute the file PROG/RAM on logical drive 4, then PROG/RAM:4<ENTER> must be keyed on the command line.

MULTIDOS SYSTEM DISK

MULTIDOS requires specific DOS overlay (/DOL) files present on a MULTIDOS disk, mounted in logical drive 0, for proper operation. And MULTIDOS requires a MULTIDOS system disk to be in logical drive 0 to load and execute /CMD files from a MULTIDOS disk, a data disk, or an alien system diskette. MULTIDOS does not execute programs specifically written for an alien DOS, e.g., PROFILE III+.

Proper operation of certain MULTIDOS utilities does not require a MULTIDOS system disk to remain in logical 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 disk be in logical drive 0 when any MULTIDOS utility is initially loaded and executed.

To remove system files, use the utility VFU/CMD (or FU/CMD) with the % parameter.

```
VFU %<ENTER>
```

The % parameter enables VFU/CMD to access the /DOL and /EXT files.

A minimum system MULTIDOS disk contains the following six files:

SYSRES/SYS — is the executive program loaded during initialization. SYSRES/SYS's main function is to interface with the hardware being used. SYSRES/SYS contains the code for vectors to internal routines, drive control table (DCT), all I/O routines, program loader, file positioning, and system scratch pad. Once loaded during initialization, you may remove the diskette containing SYSRES/SYS, provided you have the corresponding (same version) /DOL files present on the replacement diskette. For ESOTERIC the file name is ESORES/SYS.

DIR/SYS — contains data about every file on the disk.

Command/DOL — is the command interpreter and also contains the code for some of the LIBRARY commands. See table on next page – indicated with C.

Open/DOL — contains the code to open and initialize a file.

Allocate/DOL — Allocate granules and positions a file to the fifth or greater segment.

Close/DOL — contains the code to close and remove files.

Other MULTIDOS files are required to execute the balance of the LIBRARY commands and other functions. These files may be purged from your system diskette to obtain more room. However, once purged, these files cannot easily be recovered.

Error/DOL — Disk access related error library.

Debug/DOL — DEBUG file.

Minidos/DOL — Executor of MINIDOS. Minidos/DOL also contains the code to execute the system command CAT (MULTIDOS: 4419H).

LibraryN/EXT — N = 1, 2, or 3. Executors for LIBRARY commands. See table on next page.

INTRODUCTION

Library Command	MODEL I		MAX-80		MODEL III		MODEL 4		Combo DOS		ESOTERIC	
Append	2	1	2	1	2	1	2	1	2	1	2	1
Attrib	2	2	2	2	2	2	2	2	2	2	2	2
Auto	1	3	1	3	1	3	1	3	1	3	1	3
Blink	C	4	C	4	C	4	C	4			C	4
Boot	C	5			C	5			C	4		
Build	1	6	1	5	1	6	1	5	1	5	1	5
Cat			3	6			3	6			3	6
Cdir			3	7			3	7			3	7
Clear	C	7	C	8	C	7	C	8			C	8
Clock	C	8	C	9	C	8	C	9			C	9
Clrdsk	2	9	2	10	2	9	2	10			2	10
Cls	C	10	C	11	C	10	C	11	C	6	C	11
Comp											1	12
Config	2	11	2	12	2	11	2	12	2	7	2	13
Copy											1	14
Date	1	12	1	13	1	12	1	13	1	8	1	15
Ddam	2	13										
Dead	C	14			C	13			C	9		
Debug	C	15	C	14	C	14	C	14	C	10	C	16
Device	1	16	1	15	1	15	1	15	1	11	1	17
Dir	1	17	3	16	1	16	3	16	1	12	3	18
Do	1	18	1	17	1	17	1	17	1	13	1	19
Dump	1	19	1	18	1	18	1	18	1	14	1	20
Fmap			3	19			3	19			3	21
Forms	2	20	2	20	2	19	2	20	2	15	2	22
Free	1	21	3	21	1	20	3	21	1	16	3	23
Lib	C	22	C	22	C	21	C	22	C	17	C	24
Link	1	23	1	23	1	22	1	23	1	18	1	25
List	1	24	1	24	1	23	1	24	1	19	1	26
Load	C	25	C	25	C	24	C	25	C	20	C	27
Patch	2	26	2	26	2	25	2	26	2	21	2	28
Prot	2	27	2	27	2	26	2	27	2	22	2	29
Remove	2	28	2	28	2	27	2	28	2	23	2	30
Rename	2	29	2	29	2	28	2	29	2	24	2	31
Reset	1	30	1	30	1	29	1	30	1	25	1	32
Restor	2	31	2	31	2	30	2	31	2	26	2	33
Route	1	32	1	32	1	31	1	32	1	27	1	34
Screen	C	33	C	33	C	32	C	33	C	28	C	35
Setcom - III mode					2	33			2	29a		
Setdir - I mode										29b		
Spool			1	34			2	34			2	36
Time	1	34	1	35	1	34	1	35	1	30	1	37
Topmem	1	35	1	36	1	35	1	36	1	31	1	38
Type			C	37	C	36	C	37			C	39
Verify	C	36	C	38	C	37	C	38	C	32	C	40
V64			C	39			C	39				
V80			C	40			C	40				

TOTAL	36	40	37	40	32	40
Command/Dol	11	12	12	12	8	10
Library1/EXT	14	13	14	12	14	14
Library2/EXT	11	10	11	11	10	11
Library3/EXT	0	5	0	5	0	5
	MODEL I	MAX-80	MODEL III	MODEL 4	Combo DOS	ESOTERIC

LIBRARY COMMANDS

MULTIDOS and ESOTERIC LIBRARY commands are usually entered when the 'MULTIDOS' or 'ESOTERIC' prompt is displayed. Also, LIBRARY commands can be executed from SUPERBASIC, or an assembly language routine (please consider the memory usage of the LIBRARY command - see LIB).

A mandatory space is required after all LIBRARY commands if there are additional parameters and/or arguments.

EXAMPLE:

```
        DATE<ENTER>           {No space required. No arguments given}
```

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

```
        DATE 11/08/15<ENTER>  {At least one space after DATE}
```

  This establishes November 8, 2015 as the current RAM date.

Multiple LIBRARY commands are executed, in sequence, if a comma follows the last character of a given command.

EXAMPLE:

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

  This multiple command displays the <A>llocation directory for logical drives 0, 2, and 1, in that order.

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

  This multiple command renames NEW/CMD on the disk in logical drive 2 to OLD/CMD, then displays the directory of the disk in logical drive 2.

Whenever the term, switch, is part of a LIBRARY command line, switch represents an ON/YES or OFF/NO condition. If switch is omitted, then ON/YES is used.
MULTIDOS and ESOTERIC accept the following responses:

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

Please note the required parentheses surrounding the switch parameter.

EXAMPLE:

```
        CLOCK (N)<ENTER>                      {suppresses the clock display}
        CLOCK (Y)<ENTER> or CLOCK<ENTER>      {displays the clock}
```

Whenever a library command syntax includes an (H) parameter, then a brief help is displayed.

```
        REMOVE (H)<ENTER>

      Displays
        REMOVE FILE<ENTER>

        Resets FILE active byte in directory
```

APPEND Append a file to the end of another file.

APPEND filespec1 [to] filespec2<ENTER>
APPEND (H)<ENTER>

APPEND adds filespec1 to the end of filespec2, increasing the length of filespec2 by the length of filespec1, leaving filespec1 unaffected. The disk with filespec2 must not be write protected and have sufficient space to lengthen filespec2 by the length of filespec1.

To append BASIC programs, both files must be stored in ASCII, and the highest line number in filespec2 must be lower than the lowest line number in filespec1 for proper execution of the appended BASIC program.

EXAMPLE:

```
10 REM CLASS1/ASC
20 DATA "ENGLISH","FRENCH","GERMAN","GREEK","PIG LATIN"

200 REM CLASS2/ASC
250 DATA "SOCIAL SCIENCE","HISTORY","CHEMISTRY","ALGEBRA"

APPEND CLASS2/ASC to CLASS1/ASC<ENTER>
```

The file CLASS1/ASC has the contents of file CLASS2/ASC added to the end of the original contents of CLASS1/ASC, and the file CLASS2/ASC remains unchanged. Here is the results of the APPEND.

CLASS1/ASC after CLASS2/ASC is appended:

```
10 REM CLASS1/ASC
20 DATA "ENGLISH","FRENCH","GERMAN","GREEK","PIG LATIN"
200 REM CLASS2/ASC
250 DATA "SOCIAL SCIENCE","HISTORY",CHEMISTRY",ALGEBRA"
```

CLASS2/ASC after it is appended to CLASS1/ASC:

```
200 REM CLASS2/ASC
250 DATA "SOCIAL SCIENCE","HISTORY",CHEMISTRY",ALGEBRA"
    {file unchanged}
```

APPEND cannot append files from two diskettes to be mounted in the same logical drive. To append files, all disks must be mounted.

EXAMPLE:

```
APPEND NEXT/ASC:2 BEFORE/ASC:1<ENTER>
```

The file NEXT/ASC on the disk in logical drive 2 will be appended to the file BEFORE/ASC on the disk in logical drive 1.

ATTRIB Assign a filespec's attributes.

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

This command sets file protection attributes.

PARAM	MEANING
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.
P=level	level = The protection level (detailed below).

The protection level is entered as words, e.g., READ, EXEC, etc.

LEVEL	ACCESS PASSWORD PRIVILEGE
KILL or KI	TOTAL ACCESS (lowest level)
RENAME or NA or REN	RENAME, WRITE, READ, EXECUTE
WRITE or WR	WRITE, READ, EXECUTE
READ or RE	READ, EXECUTE
EXEC or EX	EXECUTE ONLY
NONE or NO	LOCKED OUT (highest level)

If a file has a protection level of WRITE, then the file can be accessed by any of the levels equal to or greater than WRITE with the ACCESS password. The file cannot be RENAMEd or REMOVEd without the UPDATE password.

All files have an ACCESS and UPDATE password. The default ACCESS password is 8 spaces, and the default UPDATE password is the disk's master password. MULTIDOS and ESOTERIC are shipped with the master password set to PASSWORD (upper case). With the ACCESS password set to 8 spaces (blank), a password is not required to access the file. The LIBRARY command DIR displays a P after the filename if a protection level is set.

EXAMPLE:
ATTRIB FILEA/BAS (A=USER,P=READ)<ENTER>

FILEA/BAS requires the ACCESS password, USER, to execute, load, and/or read. However, to REMOVE, RENAME, and/or WRITE to the file, the UPDATE password must be used. e.g., RENAME FILEA/BAS.PASSWORD to FILEA1/BAS<ENTER>.

EXAMPLE:
ATTRIB FILEB/BAS (V)<ENTER>

FILEB/BAS is visible and is displayed when the LIBRARY command DIR is executed.

NOTE: The password **TERRI** can be used to access and/or update **any** file.

AUTO Automatic command upon power-up/re-boot.

AUTO[[!][#]DOSCMD][<LF>DOSCMD]<ENTER>
AUTO (H)<ENTER>

DOSCMD is any valid MULTIDOS or ESOTERIC LIBRARY command or a filespec for an executable command file. The AUTO command stores up to 31 characters after the mandatory space following AUTO onto the GAT sector of the directory. If more than 31 characters are entered, no error message is printed; however, only the first 31 characters are stored. Because the AUTO command writes to the directory, the disk must not be write-protected when an AUTO command is established and/or removed.

EXAMPLE:
AUTO DIR<ENTER>

On future power-ups/re-boots the LIBRARY command DIR is executed after 'DIR' appears on the display.

AUTO provides for automatic operation of one and/or more MULTIDOS or ESOTERIC commands and/or executable command files upon power-up/re-boot. Multiple AUTO commands require a <LF> between each command and/or filespec to be executed upon power-up/re-boot.

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

On subsequent power-up/re-boots, MULTIDOS or ESOTERIC displays 'DIR', executes the LIBRARY command DIR, displays 'FREE', executes the LIBRARY command FREE, displays 'BASIC', and executes the file BASIC/CMD.

MULTIDOS and ESOTERIC accept a multiple command for an AUTO command. The multiple command is displayed once.

EXAMPLE:
AUTO DIR,FREE,BASIC<ENTER>

On subsequent power-up/re-boots, MULTIDOS or ESOTERIC displays 'DIR,FREE,BASIC', executes the LIBRARY command DIR, executes the LIBRARY command FREE, and executes the file BASIC/CMD.

An AUTO command is removed if only <ENTER> follows AUTO.

EXAMPLE:
AUTO<ENTER>

This removes an AUTO command, provided the diskette in logical drive zero is write enabled.

Once an AUTO command is set up, it is often desirable to suppress the AUTO command during subsequent power-up/re-boots. To suppress an AUTO command hold down the <ENTER> key during the power-up/re-boot sequence. On the other hand, you may want an AUTO command to execute without the ability to suppress the AUTO command. To inhibit the suppression of an AUTO command, key in an ! as the first character after the mandatory space following AUTO.

EXAMPLE:

```
AUTO !BASIC<ENTER>
```

On subsequent power-ups, the file BASIC/CMD executes after 'BASIC' is displayed, whether the <ENTER> key is held down or not.

You can make an AUTO command invisible if a # is placed in front of the first command.

EXAMPLE:

```
AUTO #BASIC<ENTER>
```

On subsequent power-ups, the file BASIC/CMD executes, but BASIC is not displayed on the screen.

To have an AUTO command that cannot be suppressed and is invisible, key both the ! and # (any order) before the command. However, to suppress the date prompt, key the ! character first.

When setting up AUTO commands, you may wish to observe the results of the AUTO command and change it as necessary, without re-booting the computer. To execute an AUTO command without re-booting, key AUTO %<ENTER>. AUTO % converts a multiple AUTO command to a multiple command AUTO.

To query an AUTO command without re-booting the computer, key AUTO ?<ENTER>.

RULES FOR AUTO COMMANDS:

1. If BASIC is part of an AUTO command, BASIC must be the last command.
2. If a DO file is part of an AUTO command, then the DO file must be the last command.

EXAMPLE:

```
AUTO DO STARTUP,DIR 1<ENTER>
```

This AUTO command is incorrect —— the DO file is not the last command. The command DIR 1 does not execute. You should include DIR 1 in the DO file, or rearrange the AUTO command to:

```
AUTO DIR 1,DO STARTUP<ENTER>
```

BLINK Disable or enable the blinking cursor.

BLINK[switch]<ENTER>

This command enables the blinking cursor when switch is ON/YES, or disables the blinking cursor when switch is OFF/NO.

EXAMPLE:

```
BLINK<ENTER>        {Enables the blinking cursor}
BLINK (OFF)<ENTER>  {Disables the blinking cursor}
```

MODEL I and MODEL III

BOOT Reset the computer.

BOOT<ENTER>

This command performs a software reset. BOOT requires no arguments and/or parameters. Anything following BOOT is ignored.

BUILD Create a DO file.

BUILD filespec [(A)]<ENTER>
BUILD (H)<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. BUILD overwrites the filespec unless the **A** parameter is specified. If the **A** parameter is specified, then BUILD appends the input to the end of filespec.

EXAMPLE:

```
BUILD STARTUP<ENTER>      {Build input overwrites STARTUP.}
BUILD STARTUP (A)<ENTER>  {Build input appends to the end of STARTUP.}
```

To BUILD a DO file, follow these steps:

1. Key in BUILD filespec<ENTER>.
2. When the prompt 'Build input' appears, key in (up to 255 characters per build line) the keystrokes desired for DO execution. <ENTER> terminates a build line and stores an <ENTER> in the file.
3. Repeat step 2 to store all of the desired keystrokes.
4. When the next 'Build input' appears press <BREAK> to close the file.

Although BUILD accepts 255 characters per line, the command buffer can hold only 79 characters. The 255 byte line was incorporated into BUILD to accept long comments (see the % explanation on the next page.)

BUILD has three control characters: **$**, **%**, and **&**. These control characters are used to display a message, suppress video and provide you with a partial prompt or understood prompt. These control characters are recognized only if they are the first character in a BUILD line.

If the $ character is the only input in a BUILD line, then the video is suppressed and the DO file continues. If text follows the $ character, then the text is put in the command buffer and displayed without the terminating <ENTER>. This should be used for completing the last command in a DO file.

EXAMPLE:

```
Build input
$<ENTER>    {suppress video}
```

Suppresses video output and continues the DO file. The video display is turned on whenever a DO file is complete. Although MULTIDOS and ESOTERIC nest DO files, video output is enabled when any DO file is complete regardless of the DO file that suppressed the display.

The % character is BUILD's <BREAK> and comment character. If the BUILD line only consists of a % character, then a <BREAK> is entered into the DO file. If additional text follows the % character, then the text is treated as comment text and is displayed but not put in the command buffer. You can insert linefeeds between individual lines of a large comment, because BUILD allows 255 characters for each input line (keep in mind that the command buffer can only handle 79 characters).

EXAMPLE:

```
%We the people of the United States, in Order to form a more<LF>
Perfect Union, establish Justice, insure domestic Tranquillity,<LF>
Provide for the common defense, promote the general Welfare,<ENTER>
%And secure the Blessings of Liberty to ourselves and our<LF>
Posterity, do ordain and establish this Constitution for the<LF>
United States of America.<ENTER>
```

The & character enables you to input responses while a DO file is executing. If text follows the & character, then the text is input into the command buffer and is expected to be part of the response.

EXAMPLES:

```
Build input
%Enter a DOS command.<ENTER>
Build input
&<ENTER>
```

The DO file prints 'Enter a DOS command.' and waits for your input

```
Build input
&DIR <ENTER>
```

The DO file prints 'DIR', places "DIR " into the command buffer, and waits for your input.

Example of BUILDing a file for DO execution:

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

When DO BLAST is entered from the command prompt:

```
The screen clears.                              {CLS}
The message 'LOADING BASIC' is displayed.       {%LOADING BASIC}
BASIC is executed with memory set to 61440.     {BASIC 61440}
The screen clears.                              {CLS}
The message 'GET READY FOR A BLAST!' appears.   {%GET READY FOR A BLAST!}
The program BLAST/BAS loads and executes.       {RUN"BLAST/BAS"}
```

BUILD control characters are summarized in the following table:

CONTROL CHARACTER	ACTION/DESCRIPTION
$	suppress video output and continue.
%	insert <BREAK> in DO file.
%zzzzz	display zzzzz and continue.
&	pauses DO file for user command.
&zzzzz	pauses DO file for completion of command zzzzz.

ACTION/DESCRIPTION	HOW TO
To display a message and continue:	%Message<ENTER>
To suppress video output and continue:	$<ENTER>
To display a message, suppress video output, and continue:	%Message<ENTER> $<ENTER>
To input a command:	&<ENTER>
To complete a command:	&command<ENTER>
Final incomplete command for a DO file:	$command<ENTER>

The final command syntax enables you to receive the 'MULTIDOS' prompt when the command is complete (instead of pressing <BREAK> to receive the prompt).

CAT Display a directory of a TRS-80® disk.
CAT/CMD: MODEL I and MODEL III

CAT[[[-]wild[!wild…]][[:]d] ([A][,I][,P][,R][,S][,T][,X])]<ENTER>
CAT (H)<ENTER> (MAX-80, MODEL 4, and ESOTERIC)

wild = wildcard mask. See the LIBRARY command DIR for the explanation of wild.
d = logical drive number
A = disk map of files.
I = display invisible files (displayed with an I following the filename).
P = Send display to printer also.
R = Display removed files (displayed with an R following the filename).
S = display system files (displayed with an S following the filename).
T = Perform auto-skip and/or auto-side configuration without display.
X = write DCT parameters to the GAT sector of NEWDOS/80 disks.

CAT[/CMD] displays the directory on practically any TRS-80® disk, including Model I, Model III, Model 4, and MAX-80 disks, regardless of the address marks, density, sector/granule format, and performs auto skip and/or auto side configuration.

CAT[/CMD] sets bit 5 of DCT+1, locking the DCT for the logical drive thus disabling the default sector/granule format and automatic data recognition (ADR) if the target diskette has a special format. Special formats include user created and NEWDOS/80 diskettes with the GPL greater than two. Bit 5 of DCT+1 is reset if the target diskette is a normal MULTIDOS, ESOTERIC, or any LDOS formatted diskette.

The X parameter updates the GAT sector of a NEWDOS/80 diskette with the logical tracks (LUMPS) in relative byte 0CCH, and the number of physical tracks in relative byte 0CDH to ensure the entire NEWDOS/80 disk space can be used when MULTIDOS or ESOTERIC has to write to the NEWDOS/80 diskette. NEWDOS/80 doesn't care that these two bytes are set to a non-zero values. If the X parameter is used to update these two bytes, the affected disk can function normally in a NEWDOS/80 environment. MULTIDOS and ESOTERIC limit the tracks accessed if the bytes at 0CCH and 0CDH are zero by prematurely indicating a full diskette to avoid writing to a non-existing track.

CDIR Zero unused directory entries.
CDIR/CMD: MODEL I and MODEL III

CDIR[[:]d]<ENTER>
CDIR (H)<ENTER> (MAX-80, MODEL 4, and ESOTERIC)
CDIR[[:]d][(CLEAR[,C=cc][,D=dc])]<ENTER> (MODEL I and MODEL III)

cc = cylinder count for floppies
dc = directory cylinder

CDIR[/CMD] zeros unassigned directory entries. When a file is REMOVEd, MULTIDOS and ESOTERIC simply reset a bit to indicate the directory entry is available. The LIBRARY command RESTOR, simply sets this bit. CDIR[/CMD] completely zeroes the entire 32-byte directory entry, thus preventing any recovery of a REMOVEd file via RESTOR. MODEL I and MODEL III: CDIR (CLEAR) is used to remove all files except DIR/SYS and BOOT/SYS. This can be used instead of reformatting a known good diskette. Use this powerful command with extreme caution. If the target diskette is a floppy, then the number of cylinders is obtained from the GAT sector unless C=nn is specified.

CLEAR Zero Random Access Memory (RAM).

CLEAR<ENTER>

CLEAR zeroes RAM from 5200H (4A00H if ESOTERIC) to TOPMEM. If the contents of TOPMEM is FDFFH, then CLEAR sets all bytes from 5200H (4A00H if ESOTERIC) through FDFFH to 00.

CLOCK Enable or disable the real time clock display.

CLOCK[switch]<ENTER>

The ON switch displays the clock in the top right of the video. To set the clock, use the LIBRARY command TIME.

CLRDSK Erase sectors of unassigned granules.

CLRDSK[[:]d]<ENTER>
CLRDSK (H)<ENTER>

CLRDSK writes a continuous pattern of bytes on each sector of an unassigned granule. Sectors are written with the default FORMAT/CMD patterns.

EXAMPLE:
CLRDSK :3<ENTER>

This writes the appropriate pattern on all sectors of unassigned granules on the disk in logical drive 3.

CLRDSK can be aborted if <BREAK> is pressed before another cylinder is checked for unassigned granules. i.e., can only be aborted at a cylinder boundary.

CLS Clear the screen.

CLS<ENTER>

CLS clears the screen and positions the cursor to the upper left hand corner. CLS can be used in conjunction with other commands to erase the present screen data.

EXAMPLE:
CLS, FREE<ENTER>

ESOTERIC
COMP Compare two files.

Please see COMP/CMD in the System Utilities section.

CONFIG Default power-up/re-boot drive attributes.

CONFIG[[[:]d] ([SK=q][,m][,P=p][,w][,N][,SI=k] {continued on next line}
[,V=v][,ST=s][,H=h][,C=c][,X])]<ENTER>
CONFIG (H)<ENTER>

d = logical drive number, 0 to 7.
q = Y or N for double stepping (SKIP) an 80 track drive
m = media type, F = floppy, or R = hard disk.
p = physical drive number. (Floppies can be physical drive 0 - 3)
w = write protect condition, WE = write enable, or WP = write protect
N = nil logical drive. (Software remove the logical drive number)
k = number of sides, 1 or 2 for floppies, or 1 to 8 for hard disks.
v = floppy volume, 1 or 2.
s = stepping rate for floppies, or step code for hard disks.
o = hard disk head offset, 0 to 7.
c = hard disk cylinder offset, 0 to 999.
X = write current RAM configuration to logical drive zero (provided a memory disk is not configured).

CONFIG<ENTER> (no parameters) displays the current RAM configuration.

The CONFIG parameters can be more than the minimum letters if you feel more comfortable by spelling out the words.

EXAMPLE:
CONFIG :3 (PHYSICAL DRIVE = 2, SKIP, STEP = 20, WPROTECT)<ENTER>
CONFIG :1 (SKIP TRACKS = N, SIDES = 2, VOLUME = 1)<ENTER>

CONFIG parameter detail:

d = logical drive. This is the drivespec recognized when a drive number is part of a filespec, or the target disk in a LIBRARY command.

EXAMPLE:

```
VALUE/BAS:1       {logical drive 1}
DIR :2            {logical drive 2}
BACKUP :1 to :2   {logical drive 1 to logical drive 2}
```

q = skip flag. CONFIG can program a drive to read every other cylinder. CONFIG (SKIP), or CONFIG (SKIP=Y) enables MULTIDOS or ESOTERIC to read a 40 track diskette in an 80 track drive. CONFIG (SKIP=N) instructs the system to read every cylinder. You only need the first two letters of SKIP, i.e., SK. Whenever SK or SK = Y is specified, CONFIG automatically adds the WP parameter (and sets the number of cylinders in the DCT to 0, reference CAT[/CMD] and DCT information in Entry points for MULTIDOS).

m = media type. MULTIDOS or ESOTERIC can have the configuration set for floppies (F) [5¼" and/or 3½"] or hard disks (R). A separate hard disk driver is required to use hard disks. The Entry points for MULTIDOS section provides the necessary information to interface a hard disk driver with the DCT.

p = physical drive. The physical drive is the actual drive accessed as directed by the DCT. Your MULTIDOS or ESOTERIC master diskette, as created, assigns logical drive 0 to 3 to physical drive 0 to 3 respectively.

EXAMPLE:

```
        CONFIG<ENTER>
        :0, Phy = 0, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :1, Phy = 1, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :2, Phy = 2, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :3, Phy = 3, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :4, Nil
        :5, Nil
        :6, Nil
        :7, Nil
```

However, you may need to change the physical drive to be accessed as the system drive - logical drive 0. Logical reassignment requires two CONFIG commands to complete. The first has two logical drives assigned to the same physical drive, and the second command would reassign one of the logical drives to another physical drive. For floppies, the programmer normally would not have a physical drive with more than one logical drive assigned to it at the same time. Whenever logical drive 0 is to be reassigned, it is necessary to assign another drive to physical drive 0, prior to logical drive 0 assignment to another physical drive.

EXAMPLE:

```
        CONFIG :1 (P=0)<ENTER>
        CONFIG :0 (P=1)<ENTER>
```

The first CONFIG command would have both logical drive 1 and 0 assigned to physical drive 0. The second CONFIG command assigns logical drive 0 to physical drive 1. The system diskette is now moved from physical drive 0 to physical drive 1. After performing the above two CONFIG commands, the resulting RAM configuration would be:

```
        :0, Phy = 1, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :1, Phy = 0, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :2, Phy = 2, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :3, Phy = 3, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :4, Nil
        :5, Nil
        :6, Nil
        :7, Nil
```

Suppose you have three drives: DRIVE 0 = 40 tracks, DRIVE 1 = 40 tracks, DRIVE 2 = 80 tracks. And you want to copy information between several 40 cylinder diskettes. Sometimes the data is going from drive 0 to drive 1, and sometimes it is going from drive 1 to 0. You could use VFU/CMD to copy from DRIVE 0 to DRIVE 1, and DRIVE 1 to DRIVE 0. However, you do not want to exit VFU/CMD (insert your system diskette into logical drive 0) to see the files on each disk. You could use SKIP on DRIVE 2, but any time a write is required for the diskette in DRIVE 2, it would have to be swapped with the diskette in DRIVE 1. Never write to a diskette in a drive with SKIP active. Our solution could be: 1) assign logical drive 2 to physical drive 0, 2) assign logical drive 0 to physical drive 2, and 3) SKIP logical drive 0.

1) CONFIG 2(P=0)<ENTER>

```
        :0, Phy = 0, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :1, Phy = 1, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :2, Phy = 0, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :3, Phy = 3, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :4, Nil
        :5, Nil
        :6, Nil
        :7, Nil
```

2) CONFIG 0(P=2)<ENTER>

```
        :0, Phy = 2, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :1, Phy = 1, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :2, Phy = 0, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :3, Phy = 3, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :4, Nil
        :5, Nil
        :6, Nil
        :7, Nil
```

3) CONFIG 0(SKIP)<ENTER>

Now move the system diskette to physical drive 2, and copy between logical drive 1 and logical drive 2, without swapping diskettes.

EXAMPLE:

```
        COPY DATA/TXT:1 to :2<ENTER>
        COPY MEMO/TXT:2 to :1<ENTER>
```

If you made an 80 track system diskette, then step three could be eliminated by placing your 80 track system diskette into DRIVE 2.

w = write protect condition. You can change the write protect condition of a diskette without removing the diskette and sliding the write protect switch for 3½" floppies, or applying the write protect tab for 5¼" floppies.

EXAMPLE:

```
CONFIG 2(WP)<ENTER> {write protect the disk in logical drive 2}
CONFIG 2(WE)<ENTER> {write enable the disk in logical drive 2}
```

N = nil logical drive. This parameter removes a logical drive from the system. It is useful for a setup with only two drives, and you do not want MULTIDOS or ESOTERIC to continually search 4 drives when looking for a filespec. If a physical drive is not in the system, then the logical drive assigned to it should be NILed.

EXAMPLE:

```
CONFIG 2(N)<ENTER>
CONFIG 3(N)<ENTER>

:0, Phy = 0, Floppy, double density,
    one sided, one volume, step rate = 06 mS.
:1, Phy = 1, Floppy, double density,
    one sided, one volume, step rate = 06 mS.
:2, Nil
:3, Nil
:4, Nil
:5, Nil
:6, Nil
:7, Nil
```

To reverse the effects of **N**, simply assign the logical drive to a physical drive:

EXAMPLE:

```
CONFIG 2(P=2)<ENTER>

:0, Phy = 0, Floppy, double density,
    one sided, one volume, step rate = 06 mS.
:1, Phy = 1, Floppy, double density,
    one sided, one volume, step rate = 06 mS.
:2, Phy = 2, Floppy, double density,
    one sided, one volume, step rate = 06 mS.
:3, Nil
:4, Nil
:5, Nil
:6, Nil
:7, Nil
```

k = number of sides, and v = volume. The **k** parameter defines the number of surfaces to be accessed in the logical drive. If you have a two-headed drive, then perhaps you would like to treat this drive as two-sided single volume drive.

EXAMPLE:

```
        CONFIG 1(SI=2)<ENTER>

        :0, Phy = 0, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :1, Phy = 1, Floppy, double density,
            two sided, one volume, step rate = 06 mS.
        :2, Nil
        :3, Nil
        :4, Nil
        :5, Nil
        :6, Nil
        :7, Nil
```

This configuration would access drive 1 as a single logical drive with one directory (256 file capacity) and each track would have 36 sectors. This configuration is compatible with LDOS and NEWDOS/80. The single volume method permits a very large data base (710K for an 80 track data diskette) to occupy one logical drive.

However, there is an alternate configuration method for two-headed drives.

EXAMPLE:

```
        CONFIG 1(V=2)<ENTER>

        :0, Phy = 0, Floppy, double density,
            one sided, one volume, step rate = 06 mS.
        :1, Phy = 1, Floppy, double density,
            two sided, two volume, step rate = 06 mS.
        :2, Nil
        :3, Nil
        :4, Nil
        :5, Nil
        :6, Nil
        :7, Nil
```

The two volume method places two directories (file capacity of 128 each) on the diskette. The directory on side 0 (the side readable on a single-headed drive) would be accessed by the logical number without any additional arguments, i.e., DIR :1. The directory on side one (the side not readable with a single-headed drive) would be accessed by placing an apostrophe, ', immediately behind the logical drive, i.e., DIR :1'. The two volume method may be desirable for diskette portability of the data on side 0 to be accessed in a single-headed drive.

s = stepping rate. The **s** parameter sets the stepping rate for the logical drive indicated. The stepping rate, also referred to as the track to track access speed in milliseconds, should not be set to a faster rate than the disk drive manufacturer's specification. The values accepted for 3½" and 5¼" floppies are 6, 12, 20, and 30; and for hard drives 0 to 15 (this is the step code for hard drives).

o = head offset. The **o** parameter changes the head offset for hard drives. You should use SYSGEN/CMD to set up a hard drive. Nevertheless, CONFIG enables you to change the head offset for flexibility.

c = cylinder offset. The **c** parameter changes the cylinder offset for hard drives. You should use SYSGEN/CMD to set up a hard drive. Nevertheless, CONFIG enables you to change the cylinder offset for flexibility.

X = save configuration. The **X** parameter writes the current RAM configuration to logical drive 0. The **X** parameter is used to save the current RAM configuration, stored in the Drive Control Table (DCT) located in RAM from 4500H to 457FH, for future power-up/re-boots. When a CONFIG parameter is changed, only the DCT is RAM is modified. To save the RAM configuration, remove MEMDISK if present, and enter CONFIG with the **X** parameter specified to write the configuration to logical drive 0.

EXAMPLE:

```
CONFIG 1(SI=2)<ENTER>
```

This sets up logical drive 1 for two side operation, but does not save the configuration.

EXAMPLE:

```
CONFIG (X)<ENTER>
```

This writes the current RAM configuration to the disk in logical drive 0.

EXAMPLE:

```
CONFIG :3 (SI=2,X)<ENTER>
```

This sets up logical drive 3 for two sided operation, and saves this configuration to logical drive 0 for future power-up/re-boots.

ESOTERIC

COPY Duplicate a single file.

Please see COPY/CMD in the System Utilities section.

DATE Set or display the date stored in RAM.

DATE[mm/dd/yy]<ENTER>
DATE (H)<ENTER>

DATE sets the date stored in RAM to mm/dd/yy as entered. If no arguments are entered, DATE returns the date stored in RAM.

EXAMPLE:
DATE 12/25/97 {Set RAM date to December 25, 1997}
DATE 11/18/14 {Set RAM date to November 18, 2014}
DATE<ENTER> {Display current date stored in RAM}

NOTE: any non-numeric character may be used as a separator; and, one digit may be used for values between 0 and 9.

DATE 1.4.12<ENTER>

This sets RAM's date to January 4, 2012

When MULTIDOS or ESOTERIC close a file, MULTIDOS and ESOTERIC dates the file to the current date stored in RAM. If you want your files dated, then enter the date at the power-up prompt. However, if you did not enter the date at the power-up prompt, you may enter a date with the LIBRARY command DATE. The range of dates used to date the files are January 1, 1980 through December 31, 2043 (Reference FIXDATE/CMD in the SYSTEM UTILITIES section).

MODEL I
DDAM Change address marks on a single density diskette's directory.

DDAM[[:]d] (param)

d = logical drive number
param = U for USER DEFINED: used by TRSDOS® 2.3 and NEWDOS/21 on the Model I.
= D for DELETED: used by all double density systems, MULTIDOS and LDOS on the Model I.

MODEL I and MODEL III
DEAD Software power-up.

DEAD<ENTER>

This command performs a software reset after placing a 00H bytes in RAM from 4003H through FFFFH. DEAD requires nor arguments and/or parameters. Anything following DEAD is ignored.

DEBUG Real time debugger.

DEBUG[switch]<ENTER>

DEBUG is a real time debugging package to use with machine language programs. DEBUG examines and alters the contents of RAM or the Z80 registers, jumps to a specified address or begins execution with optional breakpoints, and single step Z80 instructions in RAM or execute Z80 CALLs from RAM. DEBUG cannot be use with MULTIDOS or ESOTERIC overlays and/or utilities that load into the overlay are of RAM (MULTIDOS: 4D00H to 51FFH, ESOTERIC: 1600H to 1AFFH), because DEBUG loads into this area, e.g., BACKUP/CMD, FORMAT/CMD, and RS/CMD. Once DEBUG is enabled - via DEBUG (ON) - it does not load and execute until:

1. before the first instruction is executed in an executable program with protection level of less than EXEC, or
2. <SHIFT·BREAK>, or <CTRL·BREAK> is pressed.

Upon DEBUG execution, if the display is blank, press <O> to enable the display.

FORMAT [1]: Register display only - on the right hand side on the screen.

```
    {DEBUG pokes VIDEO refresh RAM instead of using}    AF 019B
    {the display routine to display data.          }    S  H  NC
    {Therefore, while debugging a program with out-}    BC 2000
    {put directed to the display, use format [1] to}    DE 0180
    {keep abreast of display output while debugging}    HL 4009
    {a program.                                    }    AF'FFFF
    {                                              }    SZ H PNC
    {The <CLEAR> key is routed through the display }    BC'FFFF
    {routine to reset the VIDEO from a WIDE display}    DE'FEFF
    {or erase any extraneous data present on the   }    HL'FFFF
    {display when DEBUG is entered with the display}    IX 4015
    {set to format [1].                            }    IY 50C2
    {                                              }    SP 42EA
    {DEBUG will function with the display disabled.}    PC 0081
```

FORMAT [2]: Register display with indirect RAM plus any 64 bytes of RAM.

```
    AF 019B S  H  NC
    BC 2000 FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
    DE 0180 2D36 803E 58B7 C8DD CB05 66C8 57AF C50E -6 >X.....f.W...
    HL 4009 0900 0000 0000 0000 0000 0000 010A 0100 ................
    AF'FFFF SZ H PNC
    BC'FFFF 00C3 2406 1750 C32D 407E E3BE 23E3 C200 ..$..P.-@~..#...
    DE'FEFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
    HL'FFFF 00C3 2406 1750 C32D 407E E3BE 23E3 C200 ..$..P.-@~..#...
    IX 4015 010A 0100 0020 8F00 074D 0370 3F00 0000 .........M.p?...
    IY 50C2 6540 E521 AF51 060C CD9F 50CD 9550 E37E e@.!.Q....P..P.~
    SP 42EA C250 2542 F109 1540 4F4F 4C00 8805 2542 .P%B...@OOL...%B
    PC 0081 FDE1 E1DD E1D1 C1C9 4FF1 D5DD E5E5 FDD5 ........O.......
       401D 074D 0307 3F00 000B 06FB 4A3C 00FF 0000 .M.p?.....J<....
       402D C300 44C3 A307 F637 3EAF C3F2 0101 9706 ..D....7>.......
       403D 02B0 06F7 55EF 000C 4C9C 4B9C 0175 4C20 ....U...L.K..uL
       404D 0023 7EFE 2028 FAFE 3AD0 FE30 3D3C 3FC9 .#~. (..:..0=<?.
```

FORMAT [3]: Full screen, 256-byte page of RAM.

```
4400 3E10 EF00 003E 40EF 56F5 3E19 EFC3 5504 >....>@.V.>...U.
4410 C3D1 02C3 CE02 C362 043E C7EF 3E90 EF28 .......b.>..>..(
4420 3E12 EF02 3E42 EFA0 3E13 EF00 3E43 EF00 >...>B..>...>C..
4430 C3B2 4AC3 D243 C311 48C3 2848 C327 48C3 ..J..C..H.(H.'H.
4440 8C47 C3AB 47C3 9347 C382 4718 26C4 434F .G..G..G..G.&.CO
4450 47C3 A104 1820 C3D1 43C3 C043 E601 C33A G.... ..C..C...:
4460 46C3 0302 C3EE 473E 57EE AF18 17C3 B206 F.....G>W.......
4470 C3C9 063E 50EF 3E80 EFC9 0034 FDCB 0166 ...>P.>....4...f
4480 C0C3 8546 C63B 328F 447E 23FE 03C8 CD33 ...F.;2.D~#....3
4490 00FE 0D20 F4C9 2F32 EC37 C506 1410 FEC1 ... ../2.7......
44A0 C9DD 660D DD6E 0CDD 7E08 C9CD DA44 FDCB ..f..n..~....D..
44B0 0166 C230 40CD 8145 E61B CD96 44CD 3446 .f.0@..E....D.4F
44C0 3AEC 370F 30F7 C3BD 46C3 D245 C300 02C3 :.7.0...F..E....
44D0 B74B C33A 04C3 3D03 954B 79C3 7946 C3CD .K.:..=..Ky.yF..
44E0 46C3 CA46 AFC3 D946 C348 4AC3 D746 C335 F..F...F.HJ..F.5
44F0 4AC3 1D4A C3DB 46C3 944A C3E1 48C3 1949 J..J..F..J..H..I
```

In the register display, FORMAT [1] or [2], DEBUG displays all the Z80 registers, organized for interpretation either as two 8-bit registers and/or as 16-bit register pairs. 16 bytes of indirect data are presented with each register pair in format [2], because most programs use several sets of register pairs as indirect pointers or indexing registers. Each of the flag registers is shown with an ASCII representation of its flag bits. For the flag registers, the hex contents are displayed, along with a bit-by-bit alphabetic code that makes it easier to interpret the flag status. For example, bit 0 (right most bit) is the carry flag, the alphabetic code shows a C in that position whenever the carry flag is set.

Table of codes for all the flag bits:

BIT STATUS	IF BIT 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

DEBUG memory address changes are immediate upon pressing one of the following keys:

KEY	ADDRESS change
<↑>	Decrements memory display by 16d, 10H bytes
<SHIFT·↑>	Decrements memory display by 4096d, 1000H bytes
<↓>	Increments memory display by 16d, 10H bytes
<SHIFT·↓>	Increments memory display by 4096d, 1000H bytes
<→>	Increments memory display by 1 byte
<SHIFT·→>	Increments memory display by 256d, 100H bytes
<←>	Decrements memory display by 1 byte
<SHIFT·←>	Decrements memory display by 256d, 100H bytes

DEBUG Commands

Commands are executed as soon as the specified command key is pressed, or when <SPACE> or <ENTER> is pressed, as indicated below. All values entered are hexadecimal without the H suffix.

COMMAND	TERMINATOR REQUIRED	OPERATION PERFORMED
Baaaa	<ENTER>	Set minimum single-stepping address for CALLs.
C	none	Single-steps next RAM instruction, with CALLs executed in full.
Dqqqq	<SPACE>	Sets memory display starting address to qqqq.
E	none	Produces continuous C commands until <SHIFT·SPACE> is pressed.
F	none	Disables all display formats (screen contents is unaffected).
G[,kkkk]]	<ENTER>	Executes with optional breakpoint at kkkk (see detail - next page).
I	none	Single-steps next RAM instruction.
M[cccc]	<SPACE>	Sets the current modification address to cccc. Modification information is displayed in the lower left of the screen. If cccc is currently in the display, it is overlaid by a block cursor. If cccc is omitted, the last modification address is used for cccc.
N	none	Produces continuous I commands until <SHIFT·SPACE> is pressed.
O	none	Enables all display formats.
P	none	Set display to register only - format [1][3]
Rrp aaaa	<SPACE>	Loads register pair rp with the value aaaa.
Q	none	Sets display to previous format [2][3] or [3][3]
S	none	Sets display to full screen - format [3][3]
U	none	Dynamic display update until <SHIFT·SPACE> is pressed.
X	none	Sets display to register format [2][3]

Note 3: Display format is enabled via the **O** command. The **F** command merely disables DEBUG printing to the display.

If an error is made while typing an address, key in the correct address immediately after the incorrect address. DEBUG looks only at the last four characters entered.

EXAMPLE:

DFAE944<SPACE>

Displays the page of memory starting with address E944H.

The B command detail.

The B command enables you to limit the address where you can single-step a CALL. This can be helpful when you are debugging a program and the program makes a call into DOS that does not have to be single-stepped.

EXAMPLE:

B0000 900<ENTER>

This sets the lowest address for single-stepping a CALL to 900H. If you were to encounter an instruction similar to CALL 33H in memory location 788DH, pressing **I** would execute this CALL in full and not single-step this CALL.

When you press <B>, DEBUG displays the present lower limit and a space. Key in the new lower limit (hexadecimal without the H suffix) and press <ENTER>.

The G command detail.

The G command enables you to start executing at any location by keying Gjjjj, where jjjj is the address where you want execution to take place. Also, the G command has two exclusive breakpoints. i.e., whichever occurs first resets the other breakpoint. Therefore the full G command syntax is:

G[jjjj][,kkkk][,tttt]

Start executing at location jjjj and return to DEBUG's command mode at either location kkkk or location tttt.

The M command detail.

Any time you wish to alter the contents of a memory location, key **Mcccc**<SPACE>. This sets the memory modification address to cccc and puts a memory modification prompt in the lower left corner of the display. To modify the contents of cccc, key in the new, two-digit contents and press <SPACE>. The display and RAM is updated immediately, and DEBUG increments the modification address by one.

To increment the modification address and leave the current address unchanged, key <SPACE>, <COMMA> or <→>. To decrement the modification address and leave the current address unchanged, press the <←>. Similarly <↑> decrements the modification address by 16d, 10H, and the <↓> increments the modification address by 16d, 10H. Pressing any non-hex key exits the modify memory mode.

To exit DEBUG

To exit debug if the PC register was not altered, key <G><ENTER>. Otherwise (the PC register was altered), key <G><4><0><3><0><ENTER>. The latter instruction works from SUPERBASIC as well. Exiting DEBUG does not disable debug. If one of the conditions for invoking DEBUG is met, then DEBUG will load and execute until:

DEBUG (N)<ENTER>
or DEBUG (NO)<ENTER>
or DEBUG (OFF)<ENTER>

DEVICE List I/O devices, tasks, and function keys.

DEVICE[(H)] <ENTER>

This command lists the defined I/O devices and their routine entry points. If a device is LINKed or ROUTEd, then DEVICE indicates the LINK or ROUTE condition. The defined I/O devices are: KI = keyboard, DO = video display, PR = line printer, SI = RS 232-C input, and SO = RS 232-C output. For the MAX-80, MODEL 4, and ESOTERIC, DEVICE also indicates the task slot assignments and the function key entry points. And the MODEL I, does not have SI and SO.

EXAMPLE:

```
DEVICE<ENTER>
KI = I at X'45FE'
DO linked to PR = O at X'01CC'
PR = O at X'47AE'
SI = I at X'075D'
SO routed to :5 = O at X'FF00'
Slot 2 = X'4790'
F1 = X'07DE'
F2 = X'07E1'
F3 = X'07E4'
F4 = X'07E7'
F5 = X'07EA'
F6 = X'07ED'
```

For the above: the keyboard is an input device with the routine starting at RAM address 45FEH. The display is an output device with the routine starting at 01CCH, and linked to the printer whose routine starts at 47AEH. The RS 232-C output is ROUTEd to a file on the disk in logical drive 5. Task slot 2 is assigned to the routine beginning at 4790H. The function key entry points are also displayed.

DIR Display the directory for one and/or more disks.

DIR[[[-]mask[!mask][!...]][[[:]d[([A][,D][,I]] {continued on next line}
[,P][,Q][,R][,S][,T][,X][,Z])]]<ENTER>
DIR (H)<ENTER>

DIR displays the logical drive number, disk title, disk date, number of logical tracks, number of physical cylinders, free space in granules, free space in kilobytes, and the names of all visible and non-system files on logical drive d in alphabetical order.

Mask is used to limit the display to selected files that match the mask. The mask is optional, and has two wild cards. The ? character is used as a substitute for any character, and the * character is used as a filler substitute for any character. You may use ! to effect a logical OR with up to eight masks. And, be aware that the mask option does not override the **I** and/or **S** parameters discussed on the next page.

EXAMPLE:

INPUT	FILES DISPLAYED
DIR */CMD<ENTER>	with an extension of /CMD.
DIR -*/ASM!*/SRC<ENTER>	without an extension of /ASM and/or /SRC.

```
    DIR B*/*!*/?B*<ENTER>         with a B in the first position and/or a B in the
                                  second position of the extension.
    DIR -??A*/*<ENTER>            without an A in the third position.
    DIR */<ENTER>                 without an extension.
    DIR ???/*<ENTER>              with 1 to 3 characters in the filename.
```

If a mask is specified without a drive number, then MULTIDOS or ESOTERIC searches all mounted disks for the specified mask.

EXAMPLE:

```
    DIR */CMD<ENTER>        displays all /CMD files on all mounted disks.
    DIR */CMD:1<ENTER>      displays all /CMD files on logical drive 1.
    DIR */* (I,S)<ENTER>    displays all files on all mounted disks.
```

The **A** parameter displays, for each file, the date and time of the last update (Date/Time), the physical allocation including the end of file sector and the last byte within the end of file sector (EOF), the file size in granules (Grans), the number of segments - contiguous granules - (Seg), the number of logical records (Lrec), and the logical record length (LRL). A maximum of 10 files (64X16 screen) or 18 files (80X24 screen) is displayed at one time. To display the next file, press <SPACE>; to display another page of files, press <ENTER>; or, to exit, press <BREAK>.

EXAMPLE:

```
        DIR ZAP/* :0 (A)<ENTER>

        0  MULTIDOS 01/01/05  40 log  40 phy cyls    8 grans     12.00K
        Filename          Date     Time    EOF    Grans  Seg  Lrec  LRL

        ZAP/CMD P       09/11/04 01:29:38  29/219     5    1    30  256
```

The file ZAP/CMD was last updated on 9/11/2004 at 1:29:32 AM, uses 29 full sectors and 219 bytes in the 30th sector, consumes 5 granules in 1 segment (the granules are contiguous), and has 30 logical records of 256 bytes each.

The **D** parameter displays the files in descending date order - not considering time.

The **I** parameter displays invisible files as well as visible files. Invisible files are indicated with an I after the filename.

The **P** parameter directs the output to the printer as well as the display with the directory listing scrolling until the last file is displayed.

The **Q** parameter displays the files in descending size order, i.e., the largest file is displayed first.

The **R** parameter displays removed files, provided the directory entry location was not overwritten. (CDIR[/CMD] clears out a removed file's directory entry). Removed files are indicated with an R after the filename.

The **S** parameter displays the system files as well as visible files. System files are indicated with an S after the filename.

The **T** parameter displays the files in descending date order considering time.

The **X** parameter displays the files sorted by the extension with the extension separated from the filename.

The **Z** parameter reverses the sort order.

You have three ways to control the directory sort: the **D/T** parameters sorts by date, the **Q** parameter sorts by size, and the **X** parameter sorts by extension. However, any combination of the **D/T** and/or **Q** and/or **X** parameters sorts by date first, then by size, then by extension.

EXAMPLE:

```
        DIR CO*/CMD!U*/BOL!R*/BOL 0(T,Q,X,I,S)<ENTER>

        0  MULTIDOS 01/01/05  40 log  40 phy cyls    8 grans =   12.00K
        Filename          Date     Time     EOF   Grans  Seg  Lrec  LRL

        COMP     CMD    10/06/04 16:46:46    9/67     2    1    10  256
        UTIL2    BOL SP 08/04/04 23:12:22    3/123    1    1     4  256
        COPY     CMD I  03/27/42 01:45:42    5/249    1    1     6  256
        RENUM    BOL SP 10/05/03 08:50:14    5/125    1    1     6  256
        UNPACK   BOL SP 09/24/03 18:39:48    5/124    1    1     6  256
        RESOLVE  BOL SP 04/20/03 03:43:28    5/15     1    1     6  256
        UTIL     BOL SP 08/27/02 22:47:40    5/112    1    1     6  256
```

DIRECTORY, QUICK Display the directory for one disk.
At the command level, DIR can be invoked with one keystroke. The key pressed represents the logical drive desired. The keystrokes are:

```
    <0> = DIR :0    <SHIFT><0> = DIR :0'
    <1> = DIR :1    <SHIFT><1> = DIR :1'
    <2> = DIR :2    <SHIFT><2> = DIR :2'
    <3> = DIR :3    <SHIFT><3> = DIR :3'
    <4> = DIR :4    <SHIFT><4> = DIR :4'
    <5> = DIR :5    <SHIFT><5> = DIR :5'
    <6> = DIR :6    <SHIFT><6> = DIR :6'
    <7> = DIR :7    <SHIFT><7> = DIR :7'
```

The keystroke must be in the first position. For the Model I and Model III, you must do this before any other key is pressed. If you have pressed any other key(s), you must press <BREAK> instead of <←> to position the cursor to the first position. For the MAX-80, Model 4, and ESOTERIC, you may use <←> to position the cursor to the first position.

DO Substitute disk file for keyboard input.

DO filespec<ENTER>
DO (H)<ENTER>

DO executes the filespec entered. Usually the filespec has been previously created with the BUILD command. The DO command adds the extension /IDO to the filespec, if the filespec was entered without an extension. DO reads the contents of TOPMEM, creates a 290d, 122H byte buffer from TOPMEM down, executes the DO file, and resets TOPMEM after to the value prior to the DO file executing. If high memory is to be reserved for an application with/by the DO file, then TOPMEM should be set before the DO file is executed. If the DO file is to be activated during power-up/re-boot, use a multiple command AUTO, and/or the multiple AUTO command to accomplish this task.

EXAMPLE:
AUTO TOPMEM xxxxx,DO STARTER<ENTER>

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

DUMP Transfer RAM contents to a disk file.

DUMP filespec (START=ssss,END=eeee[,TRA=tttt][,CIM][,TITLE])<ENTER>
DUMP (H)<ENTER>

ssss = starting address
eeee = ending address
tttt = optional transfer address.

DUMP transfers the contents of memory starting at **ssss** and ending at **eeee** to the filespec, and adds the extension /CMD to the filespec if none was entered and the CIM parameter is not specified. The transfer address, **tttt**, defaults to 402DH, and specifies the entry point for execution of the file.

EXAMPLE:
DUMP BUG (START=X'FD00',END=X'FFFF',TRA=X'FE00')<ENTER>

The contents of memory locations FD00H through FFFFH is transferred to a disk file named BUG/CMD. When the command BUG<ENTER> is entered, MULTIDOS or ESOTERIC loads memory from FD00H to FFFFH and begins execution at FE00H.

The TITLE parameter writes a TITLE comment block (up to 12 characters) on the first sector of the file created with the DUMP command. The parameter CIM, core image memory, specifies direct transfer to disk sectors without load marks. The CIM parameter is used to create a disk file with an exact copy of RAM, and is not an executable file.

EXAMPLE:
DUMP DCT (START=X'4500',END=X'457F',CIM)<ENTER>

The contents of memory locations 4500H through 457FH is transferred to a disk file named DCT/CIM.

FMAP Disk allocation map.
FMAP/CMD: MODEL I and MODEL III (different legends with same mapping)

FMAP[[[:]d] ([A][,P])]<ENTER>
FMAP (H)<ENTER> (MAX-80, MODEL 4, and ESOTERIC)

d = logical drive
A = display a cylinder map by files
P = send to printer also

FMAP[/CMD] displays and/or prints a map of allocated granules, directory granules, and any locked out granules. FMAP[/CMD] uses the following symbols to indicate how a disk is allocated:

. = unassigned granule X = assigned granule
D = directory granule L = locked out granule

EXAMPLE:

```
FMAP 6<ENTER>
6  * DATA * 07/04/85  35 log  35 phy cyls   66 grans =   82.50K
-----------------------------------------------------------------
  0- 17 X. .. .. .. .. .. .. .. .. .. .. .. .. L. .. .. .. DD
 18- 34 .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
-----------------------------------------------------------------
Locked  1 Grans,  Media: SINGLE
```

This single-density diskette has granule 0 on cylinder 0 allocated, granule 0 on cylinder 13 locked out, and the directory uses granules 0 and 1 on cylinder 17.

FMAP[/CMD] with the A parameter displays a map of the target disk indicating the file occupying the granules for each cylinder (LUMP if NEWDOS/80). If a cylinder is unoccupied, then FMAP[/CMD] does not display and/or print the cylinder.

EXAMPLE:

```
FMAP :2 (A)<ENTER>

2  MDOS1.72 05/26/85  80 log  80 phy cyls  211 grans =  316.50K

  0 | NoDOS/Ind    | DOSM/ASM     | DOSM/ASM     |
  1 | DOSM/ASM     | DOSM/ASM     | DOSM/ASM     |
  2 | DOSM/ASM     | DOSM/ASM     | DOSM/ASM     |
  3 | DOSM/ASM     | DOSM/ASM     |              |
 17 | DIR/SYS      | DIR/SYS      | DIR/SYS      |
```

The third granule, granule 2, is unoccupied on cylinder 3; and, cylinders 4 through 16 and cylinders 18 through 79 are unoccupied.

If you are creating a printout using both the A and P parameters, be sure to check your FORMS width. The printout width is the number of granules per cylinder times 15 plus three for the cylinder number. The cylinder map for double-sided double density diskettes produces line lengths of 93 characters (6x15+3), and for MEMDISK the line length is 123 characters (8x15+3). If necessary, use MINIDOS to output the control codes to place a printer into a compressed format.

FORMS Set or display printout format parameter values.

FORMS[(I[,M=m][,W=w][,T=t][,P=p][N=n][,V=switch]
[,L=switch][,C=switch][,F=switch][,OFF][,X])]<ENTER>
FORMS (H)<ENTER>

I = initialize line counter and character counter to zero.
m = left margin. (spaces on the left hand side before printing begins)
w = width of text. (printer width should be equal to or greater than [m+w])
t = text length. (number of printed lines of text)
p = page length. (paper length in lines)
n = the number of nulls (decimal 0) sent after each linefeed (decimal 10)
L = issue a linefeed (decimal 10) after each carriage return (decimal 13)
C = route formatted printer output to RS-232-C
OFF = sets M to 0, W to 255, T to 60, P to 60, N to 0, V off, L off, C off, F off, and the line and character counters to 0.
X = save current forms setting to the system disk for default parameter values. The disk in drive zero must be write enabled. The saved values are loaded via RESET or subsequent power-up/re-boots.
MODEL I and MODEL III:
MAX-80, MODEL 4, and ESOTERIC
F = hard form feed (if a decimal 12 is LPRINTED, then decimal 12 is passed directly to the printer and sets the line counter to 0 – "LINE 1")
V = software vertical tab: if OFF sends decimal 11 directly to printer

The maximum line width is determined by the **M** and **W** values. When a line exceeds the sum of these two values, FORMS inserts a carriage return (decimal 13) to start a new line. If the **W** parameter is set to 255, then FORMS does not insert carriage returns.

The page length, the length of the form in lines, is controlled by the **P** parameter. The text length, the number of lines that are printed on a page, is controlled by the **T** parameter. For example, in printing on standard 11" paper (with the printer set to 6 lines per inch), and you want to print on 58 lines, leaving 4 blank lines at the top and 4 blank lines at the bottom, set the **P** parameter to 66 (6 X 11 = 66), and the **T** parameter to 58 (66 - 4 - 4 = 58).

EXAMPLE:
FORMS (T=58,P=66)<ENTER>

When you decide an application requires forms control, be sure to set your form to the correct position to start printing, and initialize the line and character counter to zero by using the **I** parameter.

EXAMPLE:
FORMS (I,M=0,W=80,T=60,P=66)

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

This is the default FORMS setting.

With the **V** parameter not set, FORMS interprets a decimal 11 as a linefeed without a carriage return (vertical tab one line) if (**M** + **W**) is set to the printer's width.

When ESC, decimal 27, is encountered, FORMS does not count this and/or the next character in the page width. i.e., LPRINT CHR$(27);CHR$(12) sends both codes directly to the printer without generating a form feed, and leaves the character counter where it was before this ESC pair was sent to the printer. For printers that require extended escape sequences such as "esc&18d", you can issue the following sequence to keep the character counter in sync: 27 27 27 27 38 49 56 100. The printer is sent 27 38 49 56 100 (yes, only one 27).

The RAM address for some of the FORMS parameter values are:

		MULTIDOS	ESOTERIC
Text length.	(T)	4028H	1225H
Line counter.		4029H	1226H
Width of text.	(W)	402AH	1227H
Character counter.		402BH	1228H
Page length less text length.	(P)-(T)	402CH	1229H

When a FORMS command is issued, the current values are displayed, unless the **I** and/or **OFF** parameter is specified. This is useful for initializing the FORMS parameters from within SUPERBASIC without messing up the display.

To list the current parameter values, key:

```
FORMS<ENTER>
```

If **T** and **P** are equal, then FORMS does not paginate the text. If **T** and **P** are equal, and **W** is 255, then FORMS passes all characters directly to the printer. FORMS (OFF) sets **T** = **P**, and **W** = 255; therefore, passing all characters to the printer. FORMS (OFF) also disables LPRINT TAB(N) and LPRINT IND(N). To enable LPRINT TAB(N) and/or LPRINT IND(N), set the **W** parameter to the printer's width.

FREE Display the free space on all mounted disks.

FREE [(H)]<ENTER>

FREE displays the drive number, the disk's name and date, the number of free file spaces, the number of free granules and the equivalent K bytes (1024 bytes) on each mounted disk. In addition, FREE totals all the free file spaces and available disk space in kilobytes.

EXAMPLE:

```
FREE<ENTER>
0 MULTIDOS 01/01/05  156 files, and  896 grans =  896.00K
1 Megadisk 06/16/10  252 files, and 1018 grans = 3054.00K
2 Scratch  03/20/94  109 files, and  568 grans =  568.00K
3 ACoCuSSZ 07/16/00  224 files, and  101 grans =  151.50K

The total free directory space is  741 files.
The total free disk space is   4669.50 kilobytes.
```

LIBRARY COMMANDS

LIB Display the MULTIDOS or ESOTERIC LIBRARY commands.

LIB<ENTER>

This command lists the available LIBRARY commands for a full system disk. Some LIBRARY commands are contained in Command/DOL that uses RAM from 4D00H to 51FFH (ESOTERIC: 1600H to 1AFFH). The other LIBRARY commands use RAM from 5200H to 68FFH (ESOTERIC: 4A00H to 60FFH), and several LIBRARY commands for MAX-80, Model 4, and ESOTERIC also use RAM designated as application area 2. Application area 2 is defined as follows:

MAX-80	MODEL 4	ESOTERIC
3400H to 36FFH 3900H to 3BFFH	3000H to 37FFH	4200H to 49FH

LINK Simultaneous output for output devices.

LINK[dv1 [to] dv2]<ENTER>
LINK (H)<ENTER>

LINK links an output device (DO, PR or SO) to another output device.

EXAMPLE:
LINK DO to PR<ENTER>

This command echoes to the display anything going to the printer.

To un-LINK all LINKed devices, key LINK<ENTER>.

If dv1 is LINKed to dv2, then dv2 cannot be LINKed to dv1, dv1 cannot be ROUTEd to dv2, and dv2 cannot be ROUTEd to dv1. However if dv1 is LINKed to dv2, and dv2 is LINKed to dv3, and dv3 is LINKed to dv1, the system crashes.

LINK to a filespec can be accomplished by the indirect usage of the ROUTE command. To link the display to a filespec key:

```
ROUTE PR to filespec<ENTER>
LINK PR to DO<ENTER>
```

This method is better than ROUTEing the display to a disk file, because it lets you see what you are doing. Now every byte (including control codes) going to the display also goes to the filespec. A subsequent LINK<ENTER> stops sending to the filespec the information going to the display; however, a ROUTE<ENTER> is required to close the filespec. If ROUTE<ENTER> was executed first, then the display contents goes to the printer also. The LIBRARY command RESET can be used in place of both LINK and ROUTE to remove all LINKs and ROUTEs.

To filter out control codes going to the filespec key:

```
PRT cc,cc,cc,cc,cc,cc,cc,cc<ENTER>  {cc are control codes to filter}
FORMS (C)<ENTER>
ROUTE SO to filespec<ENTER>
LINK PR to DO<ENTER>
```

LIST Display disk file.

LIST filespec[([A]|[C][,E][,G][,P])]<ENTER>
LIST (H)<ENTER>

LIST displays the filespec's contents to the display or printer. If the file contains control codes (decimal 1 to 31), LIST suppresses the control codes unless the **C** parameter is specified. Graphic characters are suppressed unless the **G** parameter is specified. The **E** parameter strips the high-bit and supersedes the **G** parameter. The **P** parameter directs the listing to the device in the printer DCB. LIST can be terminated by pressing <BREAK> or paused by pressing <SHIFT>@. To continue from a listing pause, press any key other than <SHIFT>@.

The **A** parameter overrides the **C**, **E**, **G** and **P**, parameters, and performs a sector by sector dump of the filespec, pausing after each full sector is displayed. Press any key to continue with the sector dump. When the last sector is reached, only the bytes within the last sector are displayed.

LOAD Place an object file from disk into RAM.

LOAD filespec<ENTER>

LOAD transfers the filespec into RAM and returns to the command mode.

PATCH Modify the contents of a disk file.

(1) **PATCH** filespec (REC=nn[,BYTE=yy]) b1[;b2][;b3][;b4]<ENTER>
or (2) **PATCH** filespec (REC=nn) T=t1[.t2][.t3]>b1[.b2][b3]<ENTER>
PATCH (H)<ENTER>

nn = physical record of filespec (the first record is 0)
yy = relative byte in physical record nn.
b1, b2, b3, b4, etc. = decimal value of replacement bytes.
t1, t2, t3, etc. = decimal value of target bytes.

The target or byte separator can be either a semicolon and/or a period.

PATCH changes any file, regardless of the password protection level.

EXAMPLE:
PATCH CREF/BOL (REC=1) T=2;245.32>2.245;48<ENTER>

This patches the file CREF/BOL to show the total references during a full reference listing instead of a specified reference.

Format (1) is used when you know ***exactly*** where to make the change. Format (2) is used when you know the target sequence occurs only ***once*** in the physical record **nn**. For (REC=nn), **nn** must be the ***physical*** record, because PATCH ignores the file's logical record length.

PROT Change file(s) password; disk's master password, or name and/or date

PROT[[:]d] [(LOCK][,UNLOCK][,DATE][,PW])]<ENTER>
PROT (H)<ENTER>

LOCK = assign master password to all user files
UNLOCK = remove all passwords from user files
PW = change the master password
DATE = date all user files to current RAM date (used with LOCK and/or UNLOCK)

If no parameters are entered with PROT, you are prompted to change the disk's name and the date. If you want to change only one of the name/date pair, simply <ENTER> for the one not to be changed.

PROT can change the disk's master password, change the password for all visible and non-system files to the disk's master password (**LOCK**), or remove the passwords for all visible and non-system files (**UNLOCK**). The drivespec referenced must be write enabled.

EXAMPLE:
PROT (LOCK,DATE)<ENTER>

This assigns the master password to all user files, and re-dates all user files to the current RAM date.

NOTE: The master password on your MULTIDOS master disk is PASSWORD (upper case).
The master password on your ESOTERIC master disk is PASSWORD (upper case).

REMOVE Remove a filespec.

REMOVE filespec<ENTER>
REMOVE (H)<ENTER>

REMOVE resets the directory in-use bit and de-allocates the disk space used by the filespec, if the disk containing the filespec is not write protected. If a drivespec is not included in the filespec, then MULTIDOS or ESOTERIC searches for the file in ascending logical drive order and removes it, if found. For compatibility with BASIC's keyword "KILL" and MINIDOS' "K", MULTIDOS and ESOTERIC may use KILL to remove a filespec.

RENAME Change filename.

RENAME [#]filespec1 [to] filespec2<ENTER>
RENAME (H)<ENTER>

= renamed file's date is changed to the current RAM date.

RENAME changes the name of filespec1 to filespec2. Filespec2 contains the protection level, password, and directory attributes of filespec1. RENAME does not duplicate an existing filename on the same disk, and changes only the primary HIT (Hash Index Table [see ZAP]) entry. The "Fix directory." option in ZAP/CMD indicates that a HIT byte is wrong for a renamed file if the file uses more than four segments (multiple HIT entries), because RENAME does not change the HIT code for other than the primary entry. Obviously, MULTIDOS and ESOTERIC do not require the correct HIT code for other than the primary directory entry.

RESET Reset I/O devices and restore TOPMEM.

RESET[(T[,F)]]<ENTER>
RESET (H)<ENTER>

T = do not reset background task (only reset I/O devices and restore TOPMEM).
F = do not reset I/O devices (only removes all bakground tasks).

RESET un-LINKs, un-ROUTEs, removes all bakground tasks, and restore TOPMEM to the power-up/re-boot value. This stops the spooler and disables type-ahead because these functions are tasks in the interrupt chain - background tasks. A **RESET (T)**<ENTER> resets I/O devices and restores TOPMEM without disturbing anything in the interrupt chain.

If a ROUTE to filespec is active, then **RESET** or **RESET (T)** closes the file, and un-ROUTEs the device.

RESET (F) is used to remove tasks from the interrupt chain without disturbing any active LINKs and/or ROUTEs.

RESTOR Recover a REMOVEd filespec.

RESTOR filespec:d<ENTER>

The drive number is mandatory for RESTOR. RESTOR attempts to recover a REMOVEd file. If the file space on the disk has been re-assigned you are notified of this condition and RESTOR aborts. RESTOR should be used immediately after you inadvertently removed a file. Although you may receive the message:

'The file, REMOVED/FIL, has been restored.'

after many other files have been created and removed from a particular disk, the CONTENTS may be residue from another file. Please check the contents of the recovered file.

ROUTE Redirect one device to another.

ROUTE[dv1 [to] dv2]<ENTER>
or **ROUTE** dv1 [to] filespec<ENTER>
ROUTE (H)<ENTER>

ROUTE changes the routine entry point of dv1 to dv2's entry point. If format (2) is used, ROUTE creates a 290 byte buffer in high memory and directs the output destined for dv1 to the specified filespec. The first two letters in the filespec cannot match a device name. e.g., ROUTE SO to **PR**T/TXT is interpreted as ROUTE SO to PR

EXAMPLE:
ROUTE PR to DO<ENTER>

This command directs all bytes normally going to the printer to go to the display instead.

EXAMPLE:
ROUTE PR to LINE/TXT:2

This command creates a 290 byte buffer in high memory, opens the file LINE/TXT on the disk in logical drive 2, and redirects all bytes normally going to the printer to go to the file LINE/TXT. To un-ROUTE all devices, and close the file key:

ROUTE<ENTER> or RESET<ENTER> {see RESET for side effects}

SCREEN Dump the screen contents to the printer.

SCREEN<ENTER>

SCREEN transfers the screen contents to the printer. This command is ignored if the printer is not ready when SCREEN is entered. If the display is in the 80X24 mode, then SCREEN outputs 1920 bytes (twenty-four 80 character lines). And, if the FORMS is set to less than W=80, then the printout takes at least two lines for each line of video information.

This function calls the JKL routine at 4461H. If you are in SUPERBASIC, I recommend using CALL &4461 instead of CMD"SCREEN" to print the display because SUPERBASIC has the ability to directly call an address. For ESOTERIC, the call is to 877H.

The use of SCREEN can be imbedded in an assembly language program by pointing the HL register to the literal SCREEN followed by a carriage return, and calling 4405H.

```
00230         LD    HL,FUNCT
00240         CALL  4405H         ;865H for ESOTERIC
 ...
01940 FUNCT   DEFM  'SCREEN'
01950         DEFB  0DH
```

NOTE: The address of FUNCT cannot be in the overlay address space. The overlay address space is 4D00H to 51FFH for MULTIDOS and 1600H to 1AFFH for ESOTERIC.

SETCOM Set or display RS-232-C parameter values (LIBRARY command MODEL III)
SETCOM/CMD: MODEL 4 and ESOTERIC

SETCOM[([BAUD=b][,WORD=w][,par][,sel][,STOP=s][,tsw][,DTR][,RTS][,SAVE])]<ENTER>
SETCOM (H)<ENTER>

b = Baud rate: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, or 19200.
w = Word length: 5, 6, 7, or 8.
par = Parity switch. PE for parity enabled, or PD for parity disabled.
sel = Parity selection: ODD or EVEN
s = Number of stop bits: 1 or 2.
tsw = Transmit switch. TE for transmitter enabled, or TD for transmitter disabled.
DTR if entered sets DTR high, otherwise DTR is set low.
RTS if entered sets RTS high, otherwise RTS is set low.
SETCOM<ENTER> displays the current RS-232-C settings.

Formatted output to a disk file is achieved with the following two commands:

FORMS (C)<ENTER>
ROUTE SO [to] FILESPEC<ENTER>

This enables you to save a printing to a disk file, then use the LIBRARY command LIST *filespec* (P) to print the information. This is even possible without an RS-232-C board installed. To avoid double linefeeds, set FORMS (L=N) for the disk copy, and FORMS (L) for the printing if you need to have linefeeds after each carriage return.

MAX-80, MODEL 4, and ESOTERIC
SPOOL RAM print buffer (SPOOLER)

SPOOL[switch]<ENTER>
SPOOL (H)<ENTER>

The spooler is activated with the command:

SPOOL<ENTER>

The spooler uses task slot 1 and 32K of expansion RAM. The spooler is disabled, with the contents intact, by the command:

SPOOL (OFF)<ENTER>

Also, the LIBRARY command RESET removes the spooler from the task slot and leaves the spooler contents intact. If SPOOL is activated again, the printout commences when the spooler receives an additional character.

You have some control of spooling while the spooler is active. If you want to suspend spooling with the contents intact, then press <CTRL·@>. To resume spooling, press <CTRL·C>. If you want to empty the spooler, press <CTRL·E>. <CTRL·E> flushes the spooler and sends a carriage return to the printer.

TIME Set or display the time stored in RAM.

TIME[hh:mm:ss]<ENTER>
TIME (H)<ENTER>

hh = hour, mm = minute, ss = second

This command displays the current RAM time without any arguments, or sets the RAM time to the given argument.

EXAMPLES:
TIME 22:21:00<ENTER>

This sets RAM's time to 10:21 PM.

TIME 12.7<ENTER> NOTE: any non-numeric character may be used as a separator; and, one digit may be used for values between 0 and 9)

This sets RAM's time to 12:07:00 PM

TOPMEM Set or display the highest available address in system memory.

TOPMEM[ddddd|X'hhhh']<ENTER>
TOPMEM (H)<ENTER>

ddddd = a decimal number from 26879 (24831 if ESOTERIC) to 65535
hhhh = a hexadecimal number from 68FF (60FF if ESOTERIC) to FFFF.

TOPMEM sets the upper limit of memory available to the operating system. This is useful if you have some high memory drivers to protect, because all MULTIDOS and ESOTERIC programs check the value of TOPMEM and operate at that limit.

EXAMPLE:
TOPMEM X'BFFF'<ENTER>

This sets the upper memory usage to BFFFH.

EXAMPLE:
TOPMEM 61439<ENTER>

This sets the memory limit to 61439.

TOPMEM without a value displays the current TOPMEM value.

EXAMPLE:
TOPMEM<ENTER>
TOPMEM = X'EFFFH'

TYPE Type-ahead. (not MODEL I)

TYPE[switch]<ENTER>

TYPE enables you to key in characters prior to returning to the command mode. However, TYPE is disabled whenever access to a diskette is active. The LIBRARY commands DIR, CAT, and FMAP consume a type-ahead character for each line of file information displayed. Therefore, type-ahead is essentially off because these commands usually display several lines. BASIC captures one type-ahead character prior to an INKEY$ command (i.e., the INKEY$ buffer is scanned at the end of each statement and at the end of each logical line; therefore, if a key is pressed well before an INKEY$ command is encountered, the INKEY$ command receives this key press). For Model 4 MULTIDOS and ESOTERIC, press <CTRL·LEFT-SHIFT> to flush the type-ahead buffer.

VERIFY Reread a written sector.

VERIFY[switch]<ENTER>

VERIFY causes all disk writes to be reread for readability. All directory writes are automatically verified.

MAX-80 and Model 4
V64 Set video width to 64 characters.

V64<ENTER>

V64 sets the video width to 64 characters then clears the display.

MAX-80 and Model 4
V80 Set video width to 80 characters.

V80<ENTER>

V80 sets the video width to 80 characters then clears the display.

BACKUP/CMD Duplicate a diskette.

BACKUP[A][[:]d1]][to][[:]d2]<ENTER>

A = absolute format
d1 = source logical drive
d2 = destination logical drive

BACKUP/CMD duplicates all files from one diskette to another. The A option, if specified, causes an absolute format and does not check and/or warn you if the destination disk contains data. The source and destination logical drive numbers can be specified on the BACKUP command line. If the source and destination logical drives are the same, BACKUP/CMD prompts you to mount the destination or source disk as required (swapping). To prevent accidental rewriting on the source disk, when swapping is required, manually write protect the source disk.

BACKUP/CMD takes you through the easy procedure to duplicate a disk. After BACKUP/CMD is loaded and executes, the screen clears and

'Diskette Duplicator (c) 1999, V. B. Hester

Q1 Which logical drive contains the source diskette? _'

appears. BACKUP/CMD is waiting for a numerical response of 0 to 7, or 0' to 7' followed by <ENTER>. If the source drive number was specified in the BACKUP command line, then this query is bypassed.

Respond with the logical drive number for the source disk. Next:

Q2 'Which logical drive for the destination diskette? _'

Again BACKUP/CMD is waiting for a numerical response of 0 to 7, or 0' to 7' followed by <ENTER>. If the destination drive was specified in the BACKUP command line, then this query is bypassed.

Respond with the logical drive for the destination disk. Next:

Q3 '<ENTER> when the source disk is in logical drive X.'

Where X is the response to the first query (Q1). Mount the source disk, if it is not already there, into logical drive X then press <ENTER>.

BACKUP analyzes the source disk for cylinder count and density, then

'Source diskette has **YY** physical cylinders, in ZZZZZZ density.

Q4 Physical cylinderss for the destination diskette (WW to 96)? _'

is displayed. Where **YY** is the number of cylinders on the source disk and ZZZZZZ is the density of the source disk. The destination disk is formatted in ZZZZZZ density; however, you can change the cylinder count equal to or greater than WW for the destination disk. If <ENTER> is the response for the cylinder count, then BACKUP/CMD formats **YY** cylinders.

After the physical cylinder response is acceptable, BACKUP/CMD displays:

'<ENTER> when the destination disk is in logical drive **V**.'

where **V** is the response to the second query (Q2). If the destination drive is the same as the source drive, you must swap the disks. If the destination disk was previously formatted and the **A** option was not specified in the BACKUP command, then the message:

```
'Diskette contains data.
 The diskette name is DISKNAME, dated mm/dd/yy.
Q5   Reformat this diskette (Y/N/Q)? _'
```

appears. If you want to reformat this disk, press <ENTER> or <Y><ENTER>. If you want to bypass the formatting, key <N><ENTER>, BACKUP/CMD skips the formatting but verifies the destination diskette. If you want to abort, key <Q><ENTER> or press <BREAK>.

BACKUP/CMD does not check the destination diskette for a density match with the source disk. If the destination disk had been formatted differently, you must respond <Y> to the fifth query (Q5) or BACKUP/CMD aborts during the initial verification.

If the source and destination logical drives are the same, BACKUP/CMD prompts you to insert the source disk and destination disk as required. The swapping continues as necessary until all the files are copied to the destination diskette.

After BACKUP/CMD has copied all files:

'Completed.'

is displayed.

If either the source and/or destination disk was in logical drive 0, then

'Insert SYSTEM <ENTER>'

is also displayed. This prompts you to insert a MULTIDOS or ESOTERIC system disk into logical drive zero to return to the state prior to entering BACKUP/CMD.

The format pattern placed on the disks by BACKUP/CMD may be changed by either CUSTOM/CMD or by zapping one to four bytes on BACKUP/CMD's first sector. To zap BACKUP/CMD, key ZAP<ENTER> in the DOS command mode, press <F>. Enter BACKUP/CMD to the Filespec prompt. When ZAP/CMD requests "Relative sector in file BACKUP/CMD (0 to 00017/0011H).....", press <ENTER>. Relative sector zero of BACKUP/CMD is displayed at this point. The format pattern bytes for double density are located in bytes 8EH/8FH, and the format bytes for single density are located in bytes 9EH/9FH. Please see next page.

Relative sector zero of BACKUP/CMD

```
B Hex00 057A 2042 4143 4B55 5020 3739 3520 7A2E .z BACKUP 699 z.
A    10 2A20 2020 204E 4F54 4943 4520 2020 202A *    NOTICE    *
C Dr 20 2A20 2028 6329 2870 2931 3939 3520 202A *  (c)(p)1999  *
K 0  30 2A20 562E 2042 2E20 4865 7374 6572 202A * V. B. Hester *
U    40 2A20 2041 6C6C 2052 6967 6874 7320 202A *  All Rights  *
P Cy 50 2A20 2020 5265 7365 7276 6564 2020 202A *   Reserved   *
/ 10 60 2A20 2020 204E 4F54 4943 4520 2020 202A *    NOTICE    *
C 0A 70 2A2A 2A2A 2A2A 2A2A 2A2A 2A2A 0102 0030 ************...0
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 0000 0000 0000 CDBC 3321 3734 CD67 4411 0000 ......3!74.gD...
  00 B0 131A E607 D521 C140 0109 00ED B0D1 4F1A .....!.@......O.
     C0 E6B8 B112 3E00 F50E 00CD 2134 1100 00E1 ....>.....”4....
File D0 3E00 BC3E FF28 1BFD CB02 4E28 073C 32CB >..>.(....N(.<2.
0000 E0 3032 0F32 CD84 30D5 CDBC 3321 3734 CD67 02.2..0...3!74.g
000H F0 44D1 3258 31AF 3266 3132 8532 CD1F 34CB D.2X1.2f12.2..4.
```

00 00 = least sensitive

E5 E5 = mildly sensitive (IBM single density)

6C 6C = mildly sensitive (equivalent to E5 E5)

5B 5B = intermediate (TRSDOS® Model III)

6D B6 = most sensitive (MULTIDOS and ESOTERIC)

The more sensitive the byte pattern, the greater the probability a marginal diskette will fail format and the lesser the probability that having formatted successfully, the diskette will fail later. The less sensitive the byte pattern, the lesser the probability a marginal diskette will fail format and the greater the probability that having formatted successfully, the diskette will fail later.

Relative byte A0H contains the double density interleave flag. MULTIDOS and ESOTERIC are distributed with an interleave of three. If this byte is non-zero, then the interleave is two. Three was originally used for the Model I running at 1.77 MHz. An interleave of two can be used successfully if the CPU speed is at least 2.5 MHz. For the MODEL 4, an interleave of two can be used successfully, because the CPU speed of the Model 4 is 4 MHz. An interleave of two loads a long file quicker than an interleave of three. What this means is a full track takes 600 mS to read with an interleave of three and 400 mS to read with an interleave of two. Please see ZAP 014. The interleave flag can be changed by CUSTOM/CMD.

MODEL I and MODEL III
CAT/CMD Display a directory of a TRS-80® disk.

Please see CAT in the LIBRARY COMMANDS section.

MODEL I and MODEL III
CDIR/CMD Zero unused directory entries.

CDIR[[:]d][(CLEAR[,C=cc][,D=dc])]<ENTER> (MODEL I and MODEL III)

cc = cylinder count for floppies
dc = directory cylinder

CDIR/CMD zeros the entire 32-byte unassigned directory entry, and prevents the recovery of a REMOVEd file via RESTOR. CDIR (CLEAR) is used to remove all files except DIR/SYS and BOOT/SYS. This can be used instead of reformatting a known good diskette. Use this powerful command with extreme caution. If the target diskette is a floppy, then the number of cylinders is obtained from the GAT sector unless C=nn is specified.

CDIR is a LIBRARY command for the MAX-80, MODEL 4, and ESOTERIC versions.

MODEL III, MAX-80, MODEL 4, and ESOTERIC
CONVERT/CMD Change address marks on single density disks.

CONVERT[[:]d]<ENTER>

CONVERT/CMD changes the USER DEFINED address marks on a single density diskette's directory sectors to DELETED address marks. DELETED address marks is required on all diskette's directory sectors for the Model III, Model 4, and MAX-80 versions of MULTIDOS to read; and, ESOTERIC to read. When you attempt to display the directory of a single density diskette and the error message is:

Granule allocation table read error.
Directory read error.

the single density diskette probably needs the address marks converted.

EXAMPLE:

CONVERT :2<ENTER>

This changes the address marks on the disk in drive 2.

If no drive number or zero is specified,

'Press <ENTER> when the target diskette is in drive zero.'

is displayed. Insert the single density diskette in drive zero, then press <ENTER>. After CONVERT/CMD is complete or an error occurs, you are prompted to insert a MULTIDOS or ESOTERIC system disk.

COMP/CMD Compare two files. (This is an ESOTERIC LIBRARY command.)

```
COMP filespec1:d1 [to] [filespec2:]d2 [(P[,G])]
COMP [:]d filespec1 [to] filespec2 [(P[,G])]
COMP [:]d filespec [(P[,G])]
COMP $ filespec:d1 [to] [filespec:]d2 [(P[,G])] {d1 and/or d2 = 0}
COMP [:]0 $ filespec1 [to] filespec2 [(P[,G])]
COMP [:]0 $ filespec [(P[,G])]
COMP (H)<ENTER>
```

The last three syntaxes are used whenever an alien and/or data diskette will be used in logical drive zero.

```
H = Help
P = direct to printer also.
P,G = direct to printer also printing 80H to BFH as graphic characters.
```

COMP/CMD compares, byte for byte, two files, and totals the differences.

EXAMPLE:

```
COMP DOG/CMD:1 2 <ENTER>

'Mismatch: physical sector 05H relative byte 02H  00H . vs 01H .
 Mismatch: physical sector 05H relative byte 03H  26H & vs 19H .
 Mismatch: physical sector 05H relative byte 33H  08H . vs 00H .
 Mismatch: physical sector 05H relative byte 34H  24H $ vs 25H %
 Total differences =     4 bytes.'
```

The file DOG/CMD on logical drive 1 has a 26H in the third byte on physical sector 5 (the first physical sector is 0), and the file DOG/CMD on logical drive 2 has a 19H in the same byte.

When a mismatch is found, COMP/CMD pauses. You can single-step through the mismatches by pressing <SPACE>, abort with <BREAK>, or continue without pauses by pressing <ENTER>. If you press <ENTER>, you can press <SPACE> to pause.

COPY/CMD Duplicate a single file. (This is an ESOTERIC LIBRARY command.)

```
COPY [#]filespec1:d1 [to] [filespec2:]d2
COPY [:]d [#]filespec1 [to] filespec2
COPY [:]d [#]filespec
COPY $ [#]filespec:d1 [to] [filespec:]d2  {d1 and/or d2 = 0}
COPY [:]0 $ [#]filespec1 [to] filespec2
COPY [:]0 $ [#]filespec
COPY (H)<ENTER> (MAX-80, MODEL 4, and ESOTERIC)
```

```
H = Help
# = the destination file is to be dated to the current RAM date.
$ = whenever a MULTIDOS or ESOTERIC system diskette must be removed from
    drive zero.
```

COPY/CMD duplicates a file from one disk to another. The disk containing filespec1 is referred to as the source disk. The disk to receive filespec1 is referred to as the destination disk. If the file's name on the destination disk is to be different, then use filespec2.
If the filespec for the destination disk is the same as the filespec for the source disk, then the filespec need not be repeated.

```
COPY CHARGES/TXT:1 to :2<ENTER>
```

If the destination disk drivespec is the same as the source disk drivespec, COPY/CMD prompts you when to mount the source or destination disk.

```
COPY :3 SHIFT/TXT to MURK/ABC<ENTER>
COPY SHIFT/TXT:3 to MURK/ABC:3<ENTER>
```

Both of the above two commands duplicates the contents of SHIFT/TXT to MURK/ABC on the disk in logical drive number three, and COPY/CMD prompts you when to mount the source or destination disk.

When logical drive zero is a drivespec, and the source and/or destination disk is not a MULTIDOS or ESOTERIC system disk, a $ must precede the source filespec. A MULTIDOS or ESOTERIC system disk is a disk with at least, Allocate/DOL, Command/DOL, Open/DOL, Close/DOL, and Error/DOL.

```
COPY $WHENEVER/BAS:0 to :2
COPY :0 $HELPME/CIM to SHOWME/CIM
COPY $ THEM/OLD:0 to THEM/NEW:1
```

Please follow the prompting provided by COPY/CMD. Whenever the prompt:

```
'Insert SYSTEM <ENTER>'
```

is displayed, remove the source or destination disk from logical drive 0, insert a MULTIDOS or ESOTERIC system disk into logical drive 0, then press <ENTER>.

COPY/CMD normally assigns the source filespec's date to the destination file. However, if you want the destination file is to receive the date stored in RAM, place a # immediately in front of the source filespec.

```
COPY #CHECKING/BAS:1 to CHECKING/BAS:2
COPY #CHECKING/BAS:0 to :0
COPY :2 #CHECKING/BAS
COPY 2 #CHECKING/BAS to CHECKING/BAS
COPY :0 $ #CHECKING/BAS
```

COPY/CMD does not prompt you to mount a disk if the source and destination drivespecs are different, and neither is logical drive 0.

```
COPY FUNLOVER/TXT:2 to :3
```

Does not prompt you to mount any disks. And the colon, :, is optional for stand alone drivespecs.

```
COPY FILEDOS:1 to 2
```

CUSTOM/CMD System and Utility customizer (zapper).

CUSTOM<ENTER>

CUSTOM/CMD enables you to customize your system and incorporates all of the MAX-80, and MODEL 4 optional ZAPs; and, most of the MODEL I and MODEL III optional ZAPs. CUSTOM/CMD requires no parameters because CUSTOM/CMD is menu driven. When you have updated an option for a particular program, you may press <ENTER> to update the file, or press <BREAK> to update another option for the same file. You can return to the previous menu by pressing <BREAK>. Although the changes are immediate – upon pressing <ENTER>, changes to SYSRES/SYS and ESORES/SYS do not take effect until you re-boot your computer.

ZAPS in CUSTOM/CMD

MODEL I	MODEL III	MODEL 4
ZAP002	ZAP002	ZAP001
ZAP003	ZAP003	ZAP003
ZAP005	ZAP005	ZAP004
ZAP006	ZAP006	ZAP005
ZAP007	ZAP007	ZAP006
ZAP008	ZAP008	ZAP014
ZAP011	ZAP011	ZAP019
		ZAP021
		ZAP022
		ZAP023
		ZAP024
		ZAP025

DBLFIX/CMD Fix boot sector on DBLDOS diskettes.

DBLFIX [:]d<ENTER>

The drive number specified, d, must be other than zero. DBLFIX/CMD modifies DBLDOS data and system diskettes to be read and written to by MULTIDOS or ESOTERIC. The DBLDOS diskette will perform the same after DBLFIX/CMD has modified the diskette. The DBLDOS diskette to be modified must be write enabled.

EXAMPLE: To fix a DBLDOS diskette in drive 1.

DBLFIX :1<ENTER>

DDT/CMD 3½" and 5¼" floppy disk drive timer.

DDT<ENTER>

DDT/CMD is a diskette drive rotation speed timer, requiring a diskette, formatted or un-formatted, to be in the selected disk drive. Press any key to return to the drive select prompt. Press <BREAK> to exit DDT/CMD.

FIXDATE/CMD Change date format (except for floppy in logical drive zero)

FIXDATE [:]d [(param[,ABS][,CHECK][,G])]<ENTER>
FIXDATE (H)<ENTER>

d = drive number — the drive number is mandatory.

param = 1) ZT converts 1.0 through 2.0 to 3.xx format (this parameter is the default, i.e., if omitted "ZT" will be assumed).
2) TZ converts 3.xx to 1.0 through 2.0 format
3) OZ converts 2.1x to 1.0 through 2.0 format
4) ZO converts 1.0 through 2.0 to 2.1x format
5) **OT converts 2.1x to 3.xx format**
6) TO converts 3.xx to 2.1x format

ABS or A = Specifies update files that are probably in the previous format to the new format.
CHECK or C = Display the files to be changed without changing them.
GAT or G = Update GAT bytes CCH and CDH to represent the number of physical cylinders less 35 in CCH, and the diskette's configuration in CDH (see the discussion of these two bytes in the DIRECTORY STRUCTURE section).
H = Help

FIXDATE/CMD changes the date format among the three formats used by MULTIDOS. It is recommended that you convert your disks once. It is relatively easy to convert your disks from 2.0 to the format used by 3.xx; however, the conversion from 2.1x to 3.xx or 3.xx to 2.1x may require the ABS parameter. The ABS parameter is used to convert files that their date format cannot be interpreted properly. Normally, during initial conversions, you would not use this parameter. However, it may be required if you are going to convert back to the 2.0 format with the TZ and/or REV parameter. Version 2.1x uses the bytes previously assigned to the access password to store the hour, minute, and year. The previous bytes for the date remained. Version 3.xx uses the bytes previously assigned to the update and access passwords to store the year, month, day, hour, minute, and seconds divided by two. The previous bytes used for the date are now used for the access password. ESOTERIC only uses the 3.xx format.

Although FIXDATE/CMD automatically updates the bytes on the GAT table at locations CCH and CDH (for use with Open/DOL on Version 2.11 or later) whenever any file's date is converted, you may specify the **G** parameter to force this action if none of the dates are converted. Open/DOL, for versions 2.11 or later, has been changed to only access a diskette if the drive's configuration agrees with the diskette's configuration. i.e., if you put a single-sided diskette in a drive that is configured double-sided, Open/DOL does not open the file and returns "Drive Configuration Error." (DOS error 40/28H). However, [V]FU/CMD has built in error messages and returns "Unknown error."

FIXDATE/CMD will not change the dates on a floppy diskette in logical drive zero.

MODEL I and MODEL III
FMAP/CMD Disk allocation map.

Please see FMAP in the LIBRARY COMMANDS section.

FORMAT/CMD Prepare a disk for data storage.

FORMAT[[:]d] [A][!][-diskname]<ENTER>

d = drive number
A = absolute format
! = use current DCT parameter values, and the default track count {from a sector in FORMAT/CMD}
- = to signify a disk name to follow

FORMAT/CMD prepares a disk for file storage only.

FORMAT/CMD takes you through the easy procedure to format a disk. After FORMAT/CMD is loaded and executes, the screen clears and

'Formatter (c) 1998, V. B. Hester

Which logical drive contains the diskette to be formatted? _'

is displayed. Reply with the target diskette's logical drive number. If the drive number was specified in the FORMAT command line, then this query is bypassed.

'Name of diskette to be formatted (default "* DATA *")? _'

Reply with the desired disk name 1 to 8 characters in length, or press <ENTER> if * DATA * is acceptable.

(Q3) 'Physical cylinders to format (2 to 96 - default 40)? _'

Reply with the desired track count. FORMAT can only format up to the drive capacity. If you only have a 40 track drive, you cannot format 80 tracks.

'Date for diskette to be formatted (default mm/dd/yy)? _'

Reply with a date in the format mm/dd/yy, or press <ENTER> to use the default date.

'Master password for this diskette (default PASSWORD)? _'

Reply with the password 1 to 8 characters in length, or press <ENTER> to use PASSWORD.

'Single, Double, or "P" density (S,D, or P - default "x")? _'

Reply with S, D, or P, as desired, or press <ENTER> to use "x". If the DCT is locked, please use the default setting. Otherwise, FORMAT will not function.

'Logical cylinder for the DIRECTORY (1 to XX - default 17)? _'

XX = Is either one less than the track count entered in (Q3) or 63 – whichever is lower. Press <ENTER> to use 17, or enter the desired directory cylinder.

If the disk to be formatted contains data, and the A option was not specified in the FORMAT command line, then the message:

```
'Diskette contains data.
 Diskette name is DISKNAME, dated mm/dd/yy.
 Format this diskette (Y/N/Q)? _'
```

appears. If you want to reformat this disk, press <ENTER> or <Y><ENTER>. <N><ENTER>, <Q><ENTER>, or <BREAK> aborts FORMAT.

SPECIAL FORMATS

With knowledge of the DCT parameters and how they correlate with each other, you can create special sector/granule formats. Keep in mind, however, that the maximum number of granules per cylinder is eight. Suppose you want a double density diskette to maximize the number of granules per cylinder while minimizing the number of sectors per granule thus obtaining the most efficient storage of a lot of small files. This can be accomplished by manually adjusting (via ZAP/CMD or DEBUG) the DCT and setting bit 5 of DCT+1.

EXAMPLES: For ESOTERIC replace the 45xx with 09xx.

(1) DOUBLE DENSITY - 3 sectors/granule and 6 granules/cylinder, drive #3

```
      4530H  8023 0111 0000 0306 1210 0000 0000 0000
```

This can only be for single sided diskettes. If a diskette were double sided, then the number of granules per cylinder would have to be 12 (the maximum is 8).

(2) SINGLE DENSITY - 2 sectors/granule and 5 granules/cylinder, drive #2

```
      4520H  0022 0111 0000 0205 0A08 0000 0000 0000
```

(3) DOUBLE DENSITY - 4 sectors/granule and 4 granules/cylinder, drive #1

```
      4510H  8021 0111 0000 0404 100E 0000 0000 0000
```

Double density diskettes are formatted with 18 sectors per track. Obviously, this format pattern does not utilize all of the available diskette space. It uses only 16 sectors per track. The other two sectors are formatted; however, normal I/O to this diskette does not access the "hidden" sectors.

You cannot format a double density diskette with 2 sectors/granule and 9 granules/cylinder, because the maximum number of granules per cylinder is 8,

(4) DOUBLE DENSITY - 2 sectors/granule and 8 granules/cylinder, drive #2

```
      4520H  8022 0111 0000 0208 100E 0000 0000 0000
```

When FORMAT encounters a locked DCT (bit 5 of DCT+1 set), FORMAT creates a diskette with the DCT parameters and put this information in relative bytes 0F8 through 0FF on sectors 0 and 1 of the first track.

The format pattern placed on the disks by FORMAT/CMD may be changed by zapping one to four bytes on FORMAT/CMD's first sector. Key in ZAP<ENTER>, press <F>. Enter FORMAT/CMD to the Filespec prompt. When ZAP/CMD requests "Relative sector in file FORMAT/CMD (0 to 00014/000EH).....", press <ENTER>. Relative sector number zero of FORMAT/CMD is displayed at this point. The format pattern bytes for double density are located in bytes 8EH/8FH, and for single density in bytes 9EH/9FH. The default format track value, hexadecimal, is located in byte AFH.

Relative sector zero of FORMAT/CMD

```
F Hex00 057A 2046 4F52 4D41 5420 3739 3520 7A2E .z FORMAT1098 z.
O    10 2A20 2020 204E 4F54 4943 4520 2020 202A *    NOTICE    *
R Dr 20 2A20 2028 6329 2870 2931 3939 3520 202A *  (c)(p)1998  *
M 0  30 2A20 562E 2042 2E20 4865 7374 6572 202A * V. B. Hester *
A    40 2A20 2041 6C6C 2052 6967 6874 7320 202A *  All Rights  *
T Cy 50 2A20 2020 5265 7365 7276 6564 2020 202A *   Reserved   *
/ 10 60 2A20 2020 204E 4F54 4943 4520 2020 202A *    NOTICE    *
C 0A 70 2A2A 2A2A 2A2A 2A2A 2A2A 2A2A 0102 0030 ************...0
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 = ..
  18 A0 4446 4C54 2046 4D54 2054 524B 203D 2028 DFLT FMT TRK = (
  12 B0 0000 003E 00CD 9744 DBF0 0F38 FBFD CB00 ...>...D...8....
     C0 5628 0A3E 00CD 9744 DBF0 0F38 FB3A 4834 V(.>...D...8.:H4
File D0 CD8F 3322 D132 21C5 32CD 6744 1600 CD35 ..3".2!.2.gD...5
0000 E0 340E 00CD F932 C273 32FD CB02 4E28 233A 4....2.s2...N(#:
000H F0 5D30 57C5 3E01 3229 3432 4B34 CD35 34C1 ]0W.>.2)42K4.54.
```

00 00 = least sensitive

E5 E5 = mildly sensitive (IBM single density)

6C 6C = mildly sensitive (equivalent to E5 E5)

5B 5B = intermediate (TRSDOS® Model III)

6D B6 = most sensitive (MULTIDOS and ESOTERIC)

The more sensitive the byte pattern, the greater the probability a marginal diskette will fail format and the lesser the probability that having formatted successfully, the diskette will fail later. The less sensitive the byte pattern, the lesser the probability a marginal diskette will fail format and the greater the probability that having formatted successfully, the diskette will fail later.

Relative byte B0H contains the double density interleave flag. MULTIDOS and ESOTERIC is distributed with an interleave of three. If this byte is non-zero, then the interleave is two. Three was originally used for the Model I running at 1.77 MHz. An interleave of two can be used successfully if the CPU speed is at least 2.5 MHz. For the MODEL 4, an interleave of two can be used successfully, because the CPU speed of the Model 4 is 4 MHz. An interleave of two loads a long file quicker than an interleave of three. What this means is a full track takes 600 mS to read with an interleave of three and 400 mS to read with an interleave of two. Please see ZAP 014. The interleave flag can be changed by CUSTOM/CMD.

MODEL I and MODEL III

GR/CMD Keyboard graphic character filter (high memory).

GR<ENTER>

GR/CMD modifies the keyboard driver to produce graphic characters, values 129 through 191, directly from the keyboard. GR/CMD loads into high memory and modifies the TOPMEM address. To produce graphic characters press <[RIGHT]-SHIFT·CLEAR>. To return to normal characters press <[RIGHT]-SHIFT·CLEAR> again. The code values returned from the keyboard driver as modified by GR/CMD are:

Key	Code	Key	Code	Key	Code
0	129	L	150	f	171
1	130	M	151	g	172
2	131	N	152	h	173
3	132	O	153	i	174
4	133	P	154	j	175
5	134	Q	155	k	176
6	135	R	156	l	177
7	136	S	157	m	178
8	137	T	158	n	179
9	138	U	159	o	180
A	139	V	160	p	181
B	140	W	161	q	182
C	141	X	162	r	183
D	142	Y	163	s	184
E	143	Z	164	t	185
F	144	↑	165	u	186
G	145	a	166	v	187
H	146	b	167	w	188
I	147	c	168	x	189
J	148	d	169	y	190
K	149	e	170	z	191

Model I: If you do not have a lower case hard modification, the values returned by **a** through **z** can be achieved with <SHIFT><key>. GR/CMD is not executable from BASIC.

MODEL I and MODEL III

HELP/CMD Provide syntax and quick reference for MULTIDOS LIBRARY commands.

HELP[command]<ENTER>

HELP assists you in using the MULTIDOS LIBRARY commands along with the format required to execute each command. If command is omitted or incorrect, then HELP lists all of the available commands in the HELP file.

MODEL 4 MULTIDOS

INSTALL/CMD Update the file SYSRES/SYS on your hard disk.

INSTALL<ENTER>

Before you use this utility, please note the version you are using. This information is presented to you on the initialization banner. Your version is either 2.00, 2.10 or 2.11. It is important to know whether you version is 2.00 or 2.1x. You can recognize versions earlier than 2.1x because these versions only prompted you for the date. Starting with version 2.10, you were prompted for both the date and time.

The following 3.xx files are not on version 2.00: COMP/CMD, FIXDATE/CMD, INSTALL/CMD, UTIL2/BOL, and PCOPY/CMD.

The following 3.xx files are not on version 2.10: COMP/CMD, UTIL2/BOL, and PCOPY/CMD.

The following 3.xx files are not on version 2.11: UTIL2/BOL, and PCOPY/CMD.

If you have a file with one of these names, then it will be overwritten when INSTALL/CMD is executed. Either rename your file(s) or copy them to another disk.

INSTALL/CMD is used to update the file SYSRES/SYS and the other MULTIDOS files on your hard disk while maintaining the present DCT information (CONFIG configuration). INSTALL/CMD should be executed when the latest version of MULTIDOS is mounted in one of the floppy drives. After you execute INSTALL/CMD the following message appears:

'Update hard disk's SYSRES/SYS to latest version.

Which logical drive contains the new SYSRES/SYS file? _'

Key the logical drive number that the latest version of MULTIDOS is mounted in. If INSTALL/CMD ensures you are updating a hard disk from a floppy, then updates SYSRES/SYS and an interim version of Open/DOL and Close/DOL. Now INSTALL/CMD executes the new version of VFU/CMD to copy the balance of the MULTIDOS files to your hard disk.

MODIFY THE DATE FORMAT FOR VERSIONS 2.00 (only prompted for the date)
If you are upgrading from version 2.00 (different date format) to version 3.xx, then you must execute the following command:

FIXDATE :0<ENTER>

to modify the previous date format on non-MULTIDOS files to the new date format used by MULTIDOS 3.xx (*please* reference FIXDATE/CMD).

MODIFY THE DATE FORMAT FOR VERSIONS 2.1x (prompted for both the date and time)
If you are upgrading from version 2.1x (different date format) to version 3.xx, then you must execute the following command:

FIXDATE :0 (OT)<ENTER>

to modify the previous date format on non-MULTIDOS files to the new date format used by MULTIDOS 3.xx (*please* reference FIXDATE/CMD).

LO/CMD Object file offset utility.

LO[filespec]<ENTER>
LO (H)<ENTER>

LO/CMD can change where and how a file loads. LO/CMD defaults to the /CMD extension. If you want to access a file with a different extension, place the extension after the filename, or just a / for no extension. Follow me through the steps detailed below:

LO LO<ENTER>

The screen clears and

'Load Offsetter - Version 4.0 (c) 1999, V. B. Hester
Source Filespec? LO
Relative, direct, or fixed offset (R/D/F)? _'

is displayed. Press <ENTER> (default = the first choice, R - relative).

'Program loads:
5200 - 5243
5300 - 5DFA
Entry point = 5200
Offset appendage (N/Y)? _'

is displayed. The information on the display tells us that the file LO/CMD loads from 5200H to 5243H, skips over 5244H through 52FFH, loads 5300H to 5DFAH, and has an entry point of 5200H. An offset appendage moves the file to 5200H before the file executes. A <Y><ENTER> response tells LO/CMD to add an offset appendage to a file to be specified later. Respond <Y><ENTER>. If we had entered an N or just pressed <ENTER>, we would have bypassed this query:

'Disable or enable interrupts (D/E)? _'

Press <ENTER> (default = first choice, D, disabled). Then

'New base address in HEX? _'

is displayed. An <ENTER> here defaults to the lowest load address displayed (5200H). Key in 7000<ENTER>. The 7000 is where the destination filespec loads, and the entry point is to the offset appendage. Now

'Destination filespec? _ '

is displayed. Here is where we would enter a filename (auto /CMD appended). Instead, we will start over by entering an * character to this query.

*<ENTER> starts over at the "Source Filespec? _" prompt.

Key in LO<ENTER>. Now, lets respond D<ENTER>.

'Load base starting address HEX (5E00 minimum)? _'

is displayed. Press <ENTER> (default = 5E00H).

Next

```
'Load marks or direct dump (L/D)? _'
```

is displayed. The direct offset enables us to load a /CIM type file, e.g., BOOT/SYS.

Lets start over because LO/CMD is an inappropriate file for this action. Key *<ENTER>

Key in LO<ENTER>. Now, lets respond F<ENTER>.

```
'Load base starting address HEX (5E00 minimum)? _'
```

is displayed. Press <ENTER> (default = 5E00H).

```
'Program loads:
 5200 - 5243
 5300 - 5DFA
 Entry point = 5200
 Load marks or direct dump (L/D)? _'
```

is displayed. Press <ENTER> (default = the first choice, L, load marks).

Key <Y><ENTER> to "Offset appendage (N/Y)? _"

Key <ENTER> to "Disable or enable interrupts (D/E)? _"

Now

```
'Entry point HEX? _'
```

is displayed. An <ENTER> here defaults to the appendage entry point. Any other address is taken verbatim. Key in 402D<ENTER>. The file, we could name, would load and exit to MULTIDOS. Lets start over again once more. Enter *. Key in LO<ENTER>. Respond F<ENTER>, <ENTER>. Now D<ENTER> for the "Load marks or direct dump (L/D)? _" query.

```
'File or disk sectors (F/D)? _'
```

appears on the display. An <F> or <ENTER> response takes us to the "Destination filespec? _" prompt, but a <D><ENTER> asks us for drive, track and sector. A direct dump, <D>, file has no load marks. The BOOT/SYS files on other operating systems are direct dumped files. SYSRES/SYS on your MULTIDOS and ESORES/SYS on your ESOTERIC diskette has no load marks and was written to the diskette with this utility.

SUMMARY:

An R offset loads a file to "New base address". If we add an appendage, the file relocates to the original address before it executes. If we do not add an offset appendage, the file loads to "New base address" and exits. A D offset enables us to add load marks to a file so that we can disassemble the file and/or reverse engineer the file. An F offset enables us to change a file with load marks to a direct file. Also, whenever the file load is non-contiguous, the F offset restarts at a new page (xx00H) boundary.

MODEL I and MODEL III

MEM/CMD Random Access Memory test.

MEM<ENTER>

MEM/CMD tests random access memory from 4000H to TOPMEM leaving an FFH byte in many locations. If a memory byte fails, the byte and bit that failed is displayed and MEM/CMD exits without testing additional memory.

MAX-80, MODEL 4, and ESOTERIC

MEM/CMD Random Access Memory test.

MEM[(H)|(A)]<ENTER>

H = Help
A = test expansion RAM if allocated or not

MEM/CMD tests random access memory from 0 to TOPMEM. If the expansion RAM is available, then MEM tests the expansion RAM leaving an FFH pattern in many bytes. However, if MEMDISK and/or SPOOL is using the expansion RAM, then MEM/CMD prints:

'Expansion RAM occupied. Press <ENTER> to test or <CLEAR> to end.'

If you press the <CLEAR> key, MEM/CMD exits. If you press the <ENTER> key, then MEM/CMD tests the expansion RAM leaving an FFH pattern in every byte. This would overwrite anything currently in the SPOOL file and totally destroy a MEMDISK.

MAX-80 and MODEL 4: If a memory bit fails, the byte and bit that failed is displayed and MEM/CMD exits without testing the balance of RAM. MEM/DISK names the power-up bank of RAM as BANK A, with two 32K CHUNKS (CHUNK 1 [0 to 7FFFH] and CHUNK 2 [8000H to FFFFH], and names the expansion RAM as BANK B, with two 32K CHUNKS: CHUNK 1 and CHUNK 2. MULTIDOS only banks in expansion RAM from 8000H to FFFFH replacing BANK A, CHUNK 2.

If the failing bit is indicated as BIT 8, then the memory byte is unstable.

ESOTERIC: If a memory bit fails, the byte and bit that failed is displayed and MEM/CMD exits without testing the balance of RAM. MEM/DISK names the power-up banks of RAM as BANK 0 (0 to 7FFFH) and BANK 1 (8000H to FFFFH), and names the expansion RAM as BANK 2, and BANK 3. ESOTERIC only banks in expansion RAM from 8000H to FFFFH replacing BANK 1.

ESOTERIC indicates the BANK being tested.

If the failing bit is indicated as BIT 8, then the memory byte is unstable.

MODEL I and MODEL III

MEMDISK/CMD High memory pseudo drive.

MEMDISK[param]<ENTER>

param: tt = number of tracks (2 to 16, default 16)
or X = Remove MEMDISK

MEMDISK/CMD uses high memory as a data disk allocating 2K for each track. Each track is eight granules with one sector per granule. The logical drive assigned to MEMDISK is the first available logical drive slot (the lowest one that is 'Nil'). MEMDISK is removed by

MEMDISK X<ENTER>

MEMDISK is an excellent tool for testing; but remember, of you remove MMEDISK, re-boot, or turn off your computer, the data in MEMDISK is lost!

MAX-80, MODEL 4, and ESOTERIC

MEMDISK/CMD Random access memory pseudo drive.

MEMDISK[[:]d][(H)|(S][,R][,OFF)]<ENTER>

d = drive number (0 to 7)
S or SYS = DOS system files only
R or REC = Recover MEMDISK
OFF = Remove MEMDISK

MEMDISK/CMD uses the 64K of expansion RAM as a data or system disk. If the spooler is active, then MEMDISK only uses 32K of the expansion RAM. If you want to use the spooler and MEMDISK, then the spooler should be loaded first. If drive zero or no drive number is specified, then MEMDISK/CMD copies all of the system overlays, creating a system MEMDISK including BASIC/CMD and all of the /BOL files. If you want to create a system disk without BASIC/CMD and the /BOL files, then the S parameter must be specified.

MEMDISK (S)<ENTER>

This command creates a system disk without BASIC leaving more room for other programs.

If you re-boot and you want to recover the contents of MEMDISK, then

MEMDISK (R)<ENTER>

should be executed. This command recovers the MEMDISK provided the computer is not turned off. The R parameter also recovers MEMDISK after the MEMDISK is removed via

MEMDISK (OFF)<ENTER>

This removes MEMDISK from expansion RAM and the DCT. In addition, if MEMDISK was a system disk, then MEMDISK/CMD prompts you to insert a system disk into the prior logical drive 0.

MAX-80, MODEL 4, and ESOTERIC

MODULE/CMD High memory management utility.

MODULE[*module name*]<ENTER>

MODULE/CMD requires a specific memory header to report and/or remove any high-memory modules. The use this memory header enables you to reclaim high memory once the lowest module is removed. Here is the description of the MULTIDOS/ESOTERIC memory header:

```
MLOAD   JR  START          ;pseudo branch
LAST    DW  0              ;previous TOPMEM
        DB  namex-name     ;length of name
name    DM  'Module Name'  ;module's name
namex
prva1   DW  0              ;where value1 was
prvv1   DW  0              ;value1
prva2   DW  0              ;where value2 was
prvv2   DW  0              ;value2
prvan   DW  0              ;where valuen was
prvvn   DW  0              ;valuen
        DW  0              ;zero: end of values
prta1   DW  0              ;task1 address
prtn1   DB  0              ;task1 number
prta2   DW  0              ;task2 address
prtn2   DB  0              ;task2 number
prtan   DW  0              ;taskn address
prtnn   DB  0              ;taskn number
        DW  0              ;zero: end of tasks
START                      ;module code start
```

The first group, **PRVxx** word pairs, is described as follows: prvv***1*** holds the value that was in location prva***1***, prvv***2*** holds the value that was in location prva***2***, and so on until the last pair: prvv***n*** holds the value that was in location prva***n***. The pair of words is terminated with a zero word. The second group, **PRTxx** word and byte, is described as follows: prta***1*** is the previous task address for the task number stored in prtn***1***, prta***2*** is the previous task address for the task number stored in prtn***2***, and finally prta***n*** is the previous task address for the task number stored in prtn**n**. This is also terminated with a zero word. Even if your module does not modify an address and/or change the address of an interrupt task, you must always terminate each group with a zero word. This enables the utility MODULE/CMD to set the values and slot address to where they were before the module was invoked. Executing the utility MODULE/CMD with no arguments displays the locations of all high memory modules. Executing the utility MODULE/CMD with a particular module's name removes that module from high memory and restores all values and interrupt task addresses. This enables you to suspend a background task with a new one, and then restore the previous background task when the module is removed.

The module's name can be any character that is available from the keyboard. The first character cannot be a space, and the length must be from 1 to 72 characters. For simplicity, the module names of the utilities MONITOR/CMD, PRT/CMD, SHOW/CMD, and SSAVER/CMD are MONITOR, PRT, SHOW, and SSAVER respectively. The module's name does not have to be the same as the file's name. Familiarity with assembly language programming, better equips you in creating memory headers.

MAX-80, MODEL 4, and ESOTERIC

MONITOR/CMD Real time debugger (extended Debug/DOL).

MONITOR[(H)]<ENTER>

MONITOR/CMD is a high memory utility that supersedes DEBUG. And, can be used with MULTIDOS and ESOTERIC overlays and utilities that load into the overlay are of RAM. MONITOR/CMD is dormant until a DEBUG execute action is invoked. For the MAX-80 and MODEL 4, the display format must be in the 80 character mode. If the display is blank, press <O> to enable the display. MONITOR/CMD does not have DEBUG's commands **P**, **Q**, **S**, and **X**, because MONITOR/CMD only uses the last eleven screen rows. MONITOR/CMD has an **L** command that enables you to disassemble 1,138 Z80® instructions.

L[aaaa],bbbb<ENTER>

disassembles from aaaa (default PC register) to bbbb. Use <SPACE> to pause/single-step, <ENTER> to continue, and <BREAK> to stop and return to MONITOR's display. MONITOR's display format is:

```
AF FFBB  S  H  NC           AF'0342   Z    N           I 00      EI         0000
BC 0271  ED43 2340 2220 40C9 79D8 CDDE 41CD EA41 .C#A@ @.y...A..A  BC'0006  0000
DE 0180  2D36 803E 58B7 C8DD CB05 66C8 57AF C50E -6 >X.....f.W...  DE'0019  0000
HL 4009  0900 0000 0000 0000 0000 0000 010A 0100 ................  HL'4123  0000
IX 4015  010A 0100 0020 8F00 074D 0370 3F00 0000 .........M.p?...            0000
IY 50C2  6540 E521 AF51 060C CD9F 50CD 9550 E37E e@.!.Q....P..P.~            209B
PC 0971  E1        POP     HL                                    SP F368 56FB
   401D  074D 0307 3F00 000B 06FB 4A3C 00FF 0000 .M.p?.....J<....
   402D  C300 44C3 A307 F637 3EAF C3F2 0101 9706 ..D....7>.......
   403D  02B0 06F7 55EF 000C 4C9C 4B9C 0175 4C20 ....U...L.K..uL
   404D  0023 7EFE 2028 FAFE 3AD0 FE30 3D3C 3FC9 .#~. (..:..0=<?.
```

NOTE: There are seven values displayed above the stack pointer that are displayed in an MSB·LSB format. In this example, the stack pointer (SP) is at address F368H, the next instruction is POP HL; and, when this instruction is executed, the HL register pair will be valued with 56FBH. All of MONITOR's commands behave exactly like the same command in DEBUG.

COMMAND	TERMINATOR REQUIRED	OPERATION PERFORMED
Baaaa	<ENTER>	Set minimum single-stepping address for CALLs.
C	none	Single-steps next instruction, with CALLs executed in full.
Dqqqq	<SPACE>	Sets memory display starting address to qqqq.
E	none	Produces continuous C commands until <SHIFT·SPACE> is pressed.
F	none	Disables display (screen contents is unaffected).
G[,kkkk]	<ENTER>	Executes with optional breakpoint at kkkk.
I	none	Single-steps next instruction.
M[cccc]	<SPACE>	Sets the current modification address to cccc. Modification information is displayed in the lower left of the screen. If cccc is currently in the display, it is overlaid by a block cursor. If cccc is omitted, the last modification address is used for cccc.
N	none	Produces continuous I commands until <SHIFT·SPACE> is pressed.
O	none	Enables display.
Rrp aaaa	<SPACE>	Loads register pair rp with the value aaaa.

PCOPY/CMD Backup large files onto floppy diskettes.

PCOPY<ENTER>

A large file is partitioned (segmented) into smaller pieces that fit onto floppy diskettes. PCOPY removes the file's extension and replaces it with its own recognizable extension: #??, where ?? is a number. e.g., MAIL/DAT becomes MAIL/#01, MAIL/#02, etc. Each partition diskette has a control sector enabling you to restore the original file in any sequence. And PCOPY keeps track of the diskettes disallowing you to recopy the same diskette twice.

Good operating practices for PCOPY:
All of your files to backup should have a unique filename with any extension.
e.g., CONTROL/DAT, MESSAGE/ASC, CUSTOMER/DAT, FILE/TXT, etc.

It is recommended that the extension should not be used to distinguish files.
i.e., CONTROL/DAT and CONTROL/TXT would appear the same when partitioned on backup diskettes. i.e., both would appear as CONTROL/#??, where ?? is a number.

Filenames without an extension could be inadvertently deleted when REMOVE is used to remove a backup partition. To put it another way, the /#01 extension is not recognized as part of the filename under normal operating conditions —— the # becomes a delimiter. Although REMOVE sends the complete filespec - **FILENAME/#??:d** (d is the drive number) - to Open/DOL, Open/DOL looks for **FILENAME/** on all mounted disks according to the normal DOS search. You can set bit 5 of SM0 (see ZAP 001 last paragraph, change the byte from 00 to 20) to disable REMOVE from removing a file without an extension. Obviously, PCOPY sets bit 5 of SM0.

Please use extensions.

General Operating Characteristics for PCOPY/CMD:

1. The destination drive cannot be the same as the source drive.

2. A system disk must always be in drive zero with Allocate/DOL, Close/DOL, Command/DOL, Error/DOL, and Open/DOL.

3. All disks must be initially mounted and all partition (destination) diskettes must be formatted.

4. Each backup partition must be on separate diskettes.

MAX-80, MODEL 4, and ESOTERIC

5. Cannot be executed from SUPERBASIC and return to SUPERBASIC. If PCOPY is executed from SUPERBASIC via CMD"PCOPY", then PCOPY exits to DOS when complete.

BACKUP —— making partitions of a file

PCOPY is menu driven and keeps you posted on the backup and/or restore process. If the destination diskette has enough free space to hold the backup file, then PCOPY merely copies the file. When PCOPY is executed the following screen appears:

File Backup/Restore - Version 2.01, (c) 2004 V. B. Hester

Reading record 00000
Records written 00000

Backup or Restore (B/R)?

The prompt message "Backup or Restore (B/R)?" is waiting for you to press either <B> or <R> followed by <ENTER>. This prompt defaults to <B> if just <ENTER> is pressed. You can press <BREAK> at this point to exit to DOS.

If <B><ENTER> is pressed, then the following additional prompt appears:

Filespec to Backup:

Enter the filespec to backup without a password. e.g., the file CONTROL/DAT on logical drive 3 with the password UNCLE would be entered as:

Filespec to Backup: *CONTROL/DAT:3*

At this point, PCOPY searches for the file on logical drive 3.

If you response to this prompt is CONTROL/DAT (without a drivespec), then PCOPY performs the normal DOS search for the filespec.

After you enter the filespec and DOS finds it, a message and prompt similar to the following appears:

Filespec to Backup: *CONTROL/DAT:3*
The complete file has 01500 records.

Destination drive?

The 01500 in the message is the size in physical records (sectors) of the source file that will be partitioned onto floppy diskettes. It is possible at this time to calculate the number of diskettes required to hold all partitions. Your configuration is probably different than others; therefore, PCOPY does not indicate a finite number of diskettes required (also PCOPY does not require that all of the diskette space be available). Nevertheless, here is a table of the number of free sectors that you can use to calculate the required number of floppies.

	40 cyls	80 cyls
Single-Sided/Single Density	384	784
Double-Sided/Single Density	774	1574
Single-Sided/Double Density	695	1415
Double-Sided/Double Density	1397	2837

The actual number of free sectors is one more than this table indicates, because PCOPY uses one sector on each partition diskette as a control sector.

You can press <BREAK> at the "Destination drive? " prompt to return to DOS. Or you can enter the logical drive number for mounting all floppies to receive the partitions. PCOPY does not allow you to use multiple logical drives to mount partition diskettes on for either backup or restore.

NOTE: PCOPY expects all disks to be initially mounted and all partition (destination) diskettes to be formatted.

If a diskette is in the logical drive entered, PCOPY proceeds to start the backup. If the file to backup fits on the mounted diskette, PCOPY simply copies it to the diskette in the destination logical drive (does not add the control sector either). However, you are probably using PCOPY to make a backup of a file that is too large for the floppy drives you have. In this case, PCOPY uses all the space on the mounted floppy (it doesn't have to be blank —— it may have other files on it) and prompts you to insert another diskette with the following message:

Mount next disk in logical drive d
PRESS ANY KEY WHEN READY.

Where **d** is the logical drive number you previously entered. If type-ahead is active, a simple key touch is all that is required. If type-ahead is not active, then you will have to hold a key down until the message disappears. If you do not mount a different diskette, then PCOPY responds with the following message:

There is NO space on that disk!
Mount next disk in logical drive d
PRESS ANY KEY WHEN READY.

If you remove the previously mounted diskette and press a key before you mount a different diskette, then PCOPY responds with this message (with ZAP 002 applied):

Drive not available.
<KEY> when corrected, <BREAK> to abort.

without ZAP 002 applied (and a longer time):

Hash index table read error.
Drive not available.
<KEY> when corrected, <BREAK> to abort.

You will be continually prompted to insert different floppies until the backup is complete, indicated with this message (and tone):

BACKUP COMPLETE.

If the destination diskette has enough space to hold the original file, then the message:

COPY COMPLETE

appears.

Status messages for backup

The numbers on the upper message area keep you posted on the backup.

Reading record 00000

indicates the records read from the original file. This number continue to increase until it matches the number in the message

The complete file has 01500 records.

Keep in mind that these are physical records not logical records. e.g., if your file has 24,000 logical records of 16 bytes each, then 1,500 physical records (24000 ÷ 16) are required.

The message:

Records written 00000

indicates the number of physical records written to a mounted floppy diskette. This number starts out at one for each floppy diskette. Also, you will notice for the first partition that the number of records written is one more than the number of records read. Remember, the floppy diskette has an additional sector used for restore control. After a floppy diskette is full, the status number appears in a manner similar to the following:

File Backup/Restore - Version 2.01, (c) 2004 V. B. Hester

Reading record 01397
Records written 01398

Backup or Restore (B/R)? *B*

Filespec to Backup: *MAIL/DAT*

The complete file has 01500 records.

Destination drive? *6*
Mount next disk in logical drive 6
PRESS ANY KEY WHEN READY.

After you have completed a backup, you will probably examine the partitioned directories. You will note the following:

All except the last partition has its logical record length (LRL) set to 256 (if the original file's LRL was 256, then the last partition has a LRL of 256 also). However, this information is in all of the control sectors, enabling you to restore in any order. The LRL information is only indicated on the last partition, because this is where the end of file (EOF) is for restoring the original file. If the original file does not end on a sector boundary (indicated by a EOF of **ssss**/**bbb** where **bbb** is not zero), then only the last partition has an EOF not equal to zero.

RESTORE —— merge partitions into a complete file
If you respond <R><ENTER> to "Backup or Restore (B/R)? ", PCOPY changes the display to:

File Backup/Restore - Version 2.01, (c) 2004 V. B. Hester

Reading record 00000
Records written 00000
Restored record written 00000

Backup or Restore (B/R)? *R*

Filespec to Restore:

PCOPY added the status line "Restored record written 00000".

At the "Filespec to Restore: " prompt, only enter the filename portion of the partitions. It is highly recommended that you include the drivespec, because PCOPY searches for FILENAME/#?? without using the hash index table (HIT). This is a very slow process without a drive number.

Example of the better response to the "Filespec to Restore: " prompt:

For the original file, MAIL/DAT, that was backed up into partitions, and is about to be restored from a diskette in logical drive 6, enter ***MAIL:6***

If you enter the partition filespec without a drivespec, PCOPY (via the normal DOS search) searches (long time) for the file and, if found, establishes the source logical drivespec.

After you enter the filespec and DOS finds it, a message and prompt similar to the following appears:

Filespec to Restore: *MAIL:6*
The complete file has 01500 records.

Destination Drive?

The 01500 in the message is the size of the original file not the size of the partition diskette mounted (each partition diskette has this information in a control sector). Also the status message "Reading record 00000" was changed to "Reading record 00001", indicating only the control sector was read. The "Destination Drive? " prompt is for the disk to have the complete original file. You can press <BREAK> here to return to DOS or enter the logical drive number to hold the complete file.

After PCOPY reads the rest of the partition file and writes out the data to the disk in the destination drive, you are prompted with the following:

Mount next disk in logical drive d
PRESS ANY KEY WHEN READY.

Where **d** is the logical drive the first partition diskette is mounted in.

If you press a key without mounting a different partition diskette, then the following appears:

Mounted disk was read before.
Mount next disk in logical drive 6
PRESS ANY KEY WHEN READY.

PCOPY maintains a table of the starting position for each partition diskette. In this case the same starting position was found and PCOPY refused to rewrite this information (the table limit for starting positions is 110).

If you remove that partition diskette and press a key before you mount a different partition diskette, then PCOPY responds with this message.

Drive not available.
<KEY> when corrected, <BREAK> to abort.

If you inadvertently mount a diskette without a partition of the file to restore, then PCOPY responds with this message:

File not in directory.
<KEY> when corrected, <BREAK> to abort.

If you mount a diskette with a different version than the first partition read of the original file, then PCOPY responds with:

The version on mounted disk is different.
Mount next disk in logical drive 6
PRESS ANY KEY WHEN READY.

You will be continually prompted to insert different partition diskettes until the restore is complete, indicated with this message and a tone:

RESTORE COMPLETE.

The maximum file size that PCOPY can handle is 65534 physical records/16,776,960 bytes. This is one physical record (256 bytes) less than the DOS can handle.

Status messages for restore

The numbers in the upper message area keep you posted on the restore.

Reading record 00000

indicates the records read from each partition diskette. This number changes to 00001 when the control sector is read (it also stops at 00001 for each subsequent partition; however, if everything is apparently correct, it immediately increases to the number of records read from a partition).

The message:

Records written 00000

indicates the number of physical records restored. This number continues to increase until it matches the number in the message "The complete file has 01500 records"

The message:

Restored record written 00000

is the relative physical record restored. Depending on the sequence of mounting partition diskettes, this may or may not track the number in the message "Records written 00000". If the partition diskettes are mounted in the sequence /#01, /#02, etc, then this tracks the numbers in the message "Records written 00000".

Also, you will that notice the number in "Reading record 00000" is one more than the number in "Records written 00000" for restoring the first partition diskette. Remember, the partition diskettes have an additional sector for restore control. You will receive a display similar to the following after restoring the first partition:

File Backup/Restore - Version 2.01, (c) 2004 V. B. Hester

Reading record 01398
Records written 01397
Restored record written 01397

Backup or Restore (B/R)? *R*

Filespec to Restore: *MAIL:6*
The complete file has 01500 records.

Destination Drive? *2*
Mount next disk in logical drive 6
PRESS ANY KEY WHEN READY.

Error messages

An error message you can encounter with PCOPY is:

That file is empty!

PCOPY does not attempt to backup or copy an empty file.

Insufficient space on that volume!

This is presented just before PCOPY exits when the destination disk is perceived to be non-removable. i.e., the destination is MEMDISK or a hard disk volume and there is insufficient space to hold the original file.

Of course you can receive the DOS error:

Write protected media.

With this error, PCOPY exits. Be sure all disks to receive partitions are write enabled. If this message appears because CONFIG has the destination drive write protected, then this error occurs when you attempt to write on the first partition. You have to exit anyway to write enable the drive.

Whenever the prompt:

<KEY> when corrected, <BREAK> to abort.

appears, you can press <BREAK> to exit PCOPY.

Another error message you might encounter with PCOPY during restore is:

File partitions were incorrect.
RESTORED FILE IS NOT CLOSED AND IS INCORRECT!

This occurs if you made more than one set of backup partition diskettes with different amount of free space and mixed them during the restore operation. Although the table on page 74 indicates the amount of free space on a newly formatted diskette, you can partition files on diskettes with other data. If you want additional sets of backup partitions, I recommend using **VFU** or **FU** to copy the partitions onto other diskettes. PCOPY will not mix different partition diskettes if the original file was modified before you made another set of backup diskettes.

Of course you can receive the DOS error:

Write protected media.

With this error, PCOPY exits. Be sure the disk to restore the original file to is write enabled. If this message appears because CONFIG has the destination drive write protected, then this error occurs when you attempt to restore the first partition. You have to exit anyway to write enable the drive.

MODEL I and MODEL III

PRT/CMD Forms and Printer Filter (high-memory)

PRT<ENTER>

PRT/CMD is a high memory, menu driven forms filter and EPSON™ graphics character converter. PRT/CMD is required to enable the LIBRARY command FORMS. After PRT<ENTER> is executed, the prompt:

'FORMS FILTER 3.0 – Copyright (c) 2000, Vernon B. Hester
Enter decimal code to be ignored. _'

is displayed. Enter a decimal number (1 to 255), one at a time. PRT/CND allows up to ten values to be filtered, i.e., do not get to the printer. If you do not want to filter a value or have entered all values, press <ENTER>. Next, the prompt:

'Graphics conversion for EPSON (tm) printer (N/Y)? '

Enter <Y> if you have and EPSON™ printer (does not work with GRAFTRAX PLUS) capable of printing TRS-80® video graphics and require an adjustment of 20H; otherwise, press <N> or <ENTER>. The LIBRARY command RESET or RESET (T) removes PRT/CMD from high memory.

MAX-80, MODEL 4, and ESOTERIC

PRT/CMD Printer Character Trap (high-memory)

PRT b1[,b2][,b3][...b33]<ENTER>
PRT<ENTER> to display the values trapped.

b1, b2, … b33 = values to trap

PRT/CMD accepts the first 33 values entered on the command line and inhibits these values from going to the printer (traps these values). If additional values to trap are desired, re-issue the PRT b1,b1,b2... command until you have loaded all the desired values to trap. The LIBRARY command RESET or RESET (T); or, the utility MODULE/CMD removes PRT/CMD from high-memory.

RA/CMD Reverse Assembler (disassembler)

RA[(H)|(HD)|(Z)]<ENTER>

H = help
HD = HD64180 microprocessor (XLR8er™ modification for the MODEL 4)
Z = Z80® microprocessor (MODEL I/III, MAX-80, approximately with emulators)
If neither HD nor Z is specified, RA/CMD defaults to the present microprocessor.

In the Z80® mode, RA/CMD disassembles 1,136 Z80® instructions. The same 1,136 instructions ZEUS/CMD assembles. In the HD64180 mode, RA/CMD disassembles 730 instructions. The 730 instructions are 697 common Z80® instructions and the 33 additional instructions available with the HD64180 (the HD64180 mnemonics are abbreviated). RA/CMD disassembles object code from either memory or a standard TRS-80® load module. You can direct the disassembly to the display (default) or device assigned to the printer DCB – see options. In addition, you can create source code in ASCII format that can be read into ZEUS with the # filespec modifier.
e.g., **ZEUS #*filespec***; or, once ZEUS is invoked: **L#*filespec***.

Upon executing RA/CMD, the screen clears and the following prompt appears:

Memory or disk object (M/D)? _

<ENTER> or <M><ENTER> for object code in memory.
All addresses in RA/CMD are one to four hexadecimal digits without the H suffix, and default to 0000H.

The options available for object code in memory are **NP** for no pauses and **PRT** to direct the output to the device in the printer DCB.

<D><ENTER> for object code from a disk file.
Disk files default to a /CMD extension. i.e., to disassemble DDT/CMD, enter DDT and to disassemble DOGPATCH enter DOGPATCH/

The options available for object code from a disk file are:
NP = **N**o **P**auses
PRT = Output to **PR**in**T**er (sets **NP**, i.e., does not pause)
NIP = **N**o **I**nstructions **P**rinted (the cross reference printed)
NLR = **N**o **L**ocation **R**eference (resets **NIP**, i.e., disables **NIP**)
STF = **S**ource **T**o **F**ile – default extension /TXT (sets **NP**, resets **NLR** and **NIP**)
F&&& = **F**irst label name. **AAA** through **ZZZ** (wraps around) default **TAA**
REA = **R**eference **E**nable **A**ll default except for reference type **y** - DEFB
RDA = **R**eference **D**isable **A**ll
RE% = **R**eference **E**nable **%** type
RD% = **R**eference **D**isable **%** type
The reference types are:

c = CALL mn	m = LD (mn),A	r = JR mn
i = IN A,(n) or IN0 r,(n)	n = LD r,n	s = LD rr,mn
j = JP mn	o = OUT (n),A or OUT0 (n),r	t = LD rr,(mn)
k = LD (mn),rr	p = op [A,]n	w = LD A,(mn)
y = DEFB n		

Where: n = byte 0 to FFH, mn = word 0 to FFFFH, r = 8 bit register, and rr = 16 bit register pair

To pause disassembly, press <SPACE>. To single-step after pausing, press <SPACE>. To continue, press <ENTER>. To stop, press <BREAK>. Pressing <BREAK> causes 'Another disassembly (Y/N)? _' to appear. <ENTER> or <Y><ENTER> re-starts RA/CMD. <N><ENTER> or <BREAK> exits RA/CMD.

For object code from a disk file, once the file is successfully opened, RA/CMD asks you 'Offset addresses by? _'. This adds the offset (modulo FFFFH) to the load addresses in the disk file. And, the next prompt 'Starting address? _', asks you for the address (added to offset, if specified) you want from any valid location in the disk file. <ENTER> starts you at the beginning of the file – and not necessarily the lowest address. Pressing <ENTER> for both queries directs RA/CMD to show the file as is.

For object code from a disk file, on a disassembled instruction output line, the number of references to any address in this line is displayed in the first column – maximum 0FH; and, if any reference is not to the leftmost byte (starting address), then the offset to the leftmost byte is displayed as a displacement to any reference to the addresses on this disassembled line.

EXAMPLE:

```
  2C 8905  LD       A,0
     8907  INC      A
     8908  LD       (8906H),A
```

The bit pattern for the offset is 1100 (the C in 2C); therefore, there are two references to either addresses 8905H and 8906H (you can see one of them at 8908H). The leftmost bit (the first 1 in 1100) is set because there is a reference to 8905H. This reference most likely is a JP, JR, or CALL. If there were no references to 8905H, then the output lines would be:

```
  14 8905  LD       A,0
     8907  INC      A
     8908  LD       (8906H),A
```

Here we have an offset of 0100 (the 4 in the 14) that means there is no reference to 8905H and a reference to 8906H.

For object code from a disk file, the location reference table (if not inhibited) displays in ascending order all addresses referenced and the type of reference. If you want to see references to a byte in a DEFB instruction, then you must enable reference type **y** with **REY** (case not significant) as a response to 'Option:'

EXAMPLE:

```
4411  FFBFk
44D2  FFC8j   FFD9c
```

Location 4411H is referenced at FFBFH by an instruction similar to LD (4411H),rr where rr is BC, DE, HL, IX, IY, or SP (most likely not SP).

Location 44D2H is referenced at FFC8H with a JP 44D2H; and, 44D2H is referenced at FFD9H with a CALL 44D2H

RS/CMD Memory Scanner.

RS<ENTER>

RS/CMD scans memory and locates an 8 bit byte or 16 bit word specified. Enter the *from* and *to* addresses in one to four hexadecimal digits. Press <ENTER> if less than four hexadecimal digits. To reposition to the previous prompt, press <←>.

'Word, or byte search (W/B)? _'

For a word search, press <W> or <ENTER>. For a byte search press <B>. Next RS/CMD asks you to enter the search target. Enter up to four hexadecimal digits for word search or up to two hexadecimal digits for byte search. To reposition to the previous prompt, press <←>.

Next, for word scans only, the prompt:

'Auxiliary mnemonic? _'

is displayed. You can, optionally, inquire about calls, jumps, and loads to a selected word at this time. Enter one or more of the following:

```
 C = call/carry
 J = jump
 L = load
 M = sign minus
NC = non carry
NZ = non zero
 P = sign positive
PE = parity even
PO = parity odd
 Z = zero
```

The above terms may be combined if needed (e.g., CNZ for CALL NZ,word). If no auxiliary mnemonic is desired press <ENTER>. For the L (LOAD) command, the following question

'Immediate, or Extended Addressing (I/E) _'

is displayed. If the response is E, then:

'From, or to the register (F/T) .'

is displayed. Answer as desired.

Next the query:

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

is displayed. Enter A or the desired register pair. RS/CMD displays the hexadecimal locations where the specified byte or word is found.

Please note that the start of the overlay area is used as scratchpad and the search byte or word is 'mirrored' in this area. i.e., false positive. Also, some finds may be part of two different instructions; therefore, use RA/CMD to verify the search results.

MODEL 4 and ESOTERIC

SETCOM/CMD Set or display RS-232-C parameter values

Please see SETCOM in the LIBRARY COMMANDS section.

SHOW/CMD Real time display of selected byte(s) or word(s) (not MAX-80)

SHOW[W]addr1,[W]addr2<ENTER>

addr1 = first byte or word (hexadecimal without an H suffix)
addr2 = second byte or word (hexadecimal without an H suffix)

SHOW/CMD displays the contents of on word, or one byte, or two words, or two bytes, or one word and one byte in the upper right corner of the display. If two bytes, words, or combinations are specified, then SHOW/CMD displays the second value to the left of the first value. Show reassigns background task slot six (two if Model I or Model III) to display the first value and, if specified, background task slot seven (three if Model I or Model III) to display the second value. A byte value is the default for the address entered .If you want a word value displayed, precede the address with a W.

EXAMPLE:
To display the cursor refresh RAM location and the ROW/COLUMN in the upper right using Model 4 MULTIDOS:

SHOW W4023,W4020

SHOW/CMD is removed from high memory and background task slots six and seven restored to previous assignments by executing MODULE SHOW<ENTER>.

MODEL I and MODEL III

SPOOL/CMD RAM printer buffer (SPOOLER)

SPOOL<ENTER>

SPOOL/CMD is a high memory printer buffer that uses background task slot one. SPOOL/CMD requires PRT/CMD (printer filter) to be activated. Upon execution of SPOOL<ENTER>, the following prompt appears:

How many 256 byte BLOCKS for SPOOLING (default 9, 1-99)? _

Enter the BLOCK size for the spooler. <ENTER> or an invalid input defaults to 9 BLOCKS.

You have some control of spooling while the spooler is active. To activate this control, press <RIGHT-SHIFT·BREAK>. The spooling is suspended and an **S** appears in the upper right of the display. Pressing <N> removes the spooler from high memory and resets background task slot one. Pressing <Y> changes the **S** to a **B**. With a **B** in the upper right of the display, pressing <Y> resumes spooling and <N> empties the spooler and sends a carriage return to the printer. Also, the LIBRARY command RESET removes the spooler from high memory and resets background task slot one.

MODEL I, MODEL III, and MAX-80

SSAVER/CMD Screen Saver

SSAVER[(M=m[,T]<ENTER>
SSAVER (H)<ENTER>

m = minutes to wait before SSAVER/CMD blanks the screen; default = 3
T = do not suspend background tasks when screen blanks

SSAVER/CMD protects the screen from burned-in characters or images by saving the screen contents and blanking the screen when there is no activity for a selected time period. No activity is defined as: no key press while an application is scanning the keyboard; or, the screen is not being updated in more than one position.

The maximum time period depends on the hardware interrupt rate: MODEL I at 40 per second is 27 minutes, MAX-80 at ≈61.04 per second is 17 minutes, and the MODEL III at 30 per second is 36 minutes.

The screen is restored by any key that returns a value when pressed, i.e., not <SHIFT>. This key press only restores the screen and is absorbed by the screen saver.

SSAVER/CMD loads into high memory and can be removed by the LIBRARY command RESET and for the MAX-80 removed by the utility MODULE/CMD.

MODEL 4 and ESOTERIC

SSAVER/CMD Screen Saver

SSAVER[(M=m[,T][,E][,U][,S]<ENTER>
SSAVER (H)<ENTER>

m = minutes to wait before SSAVER/CMD blanks the screen; 1 to 18, default = 3
T = do not suspend background tasks when screen blanks
E = directs SSAVER/CMD to use bank 2 of expanded memory bypassing MegaMEM'
U = directs SSAVER/CMD to use bank 3 of expanded memory bypassing MegaMEM and bank 2 of expanded memory
S = directs SSAVER/CMD to use standard memory (in high memory) bypassing MegaMEM, bank 2 of expanded memory, and bank 3 of expanded memory.

With no memory directs, SSAVER/CMD uses this sequences for the screen buffer: tests if MegaMEM is available, if no MegaMEM tests if bank 2 of expanded memory is available, if bank 2 is not available, tests if bank 3 of expanded memory is available, if bank 3 of expanded memory is not available, uses high memory as the screen buffer.

SSAVER/CMD protects the screen from burned-in characters or images by saving the screen contents and blanking the screen when there is no activity for a selected time period. No activity is defined as: no key press while an application is scanning the keyboard; or, the screen is not being updated in more than one position.

The screen is restored by any key that returns a value when pressed, i.e., not <SHIFT>. This key press only restores the screen and is absorbed by the screen saver.

SSAVER/CMD can be removed by the LIBRARY command RESET or the utility MODULE/CMD.

SYSGEN/CMD System disk generator. (not ESOTERIC)

SYSGEN<ENTER>

SYSGEN/CMD creates a MULTIDOS system diskette on a formatted disk that usually has a configuration different than the MULTIDOS master diskette.

MULTIDOS supports the following types of one-volume system diskettes:

	Model I	Model III	Model 4
1. Single-sided, single-density	Y	N	N
2. Single-sided, double-density	Y	Y	Y
3. Single-sided, pseudo-double-density	Y	Y	Y[4]
4. Double-sided, single-density	Y	N	N
5. Double-sided, double-density	Y	Y	Y

In addition, for two-volume formats, the following types of system diskettes are supported:

	Model I	Model III	Model 4
6. Single-density	Y	N	N
7. Double-density	Y	Y	Y
8. Pseudo-double-density	Y	Y	Y[4]

Note 4: Types 3 & 8 are not recommended for Model III/4.

MULTIDOS has the capability to communicate with, in addition to the eight supported system formats, the following one-volume format.

9. Double-sided, pseudo-double-density.

SYSGEN/CMD is designed to create a different type of one-volume system diskette (1 through 5) from any of the eight supported system formats.

When the initial query

'Drive of the source SYSTEM disk or * for setup (0-7 or *)? _'

appears, press the logical drive number that has a complete MULTIDOS system disk mounted. At this point you receive the DCT setup menu. The DCT setup menu only can be modified with the arrow keys, and requires non-conflicting parameters when setting up a drive. To describe the non-conflicting parameters, please read the following carefully:

Legend used for describing floppy diskettes and drives:

SS = single-sided
DS = double-sided
SD = single-density
DD = double-density
PD = pseudo-double-density
SS/SD = single-sided single-density
SS/DD = single-sided double-density
SS/PD = single-sided pseudo-double-density
DS/SD = double-sided single-density
DS/DD = double-sided double-density
DS/PD = double-sided pseudo-double-density

The number of sectors per cylinder must match the following table for 3½" or 5¼" floppy diskettes.

SD	DD	PD
10 SS, 20 DS	18 SS, 36 DS	10 SS, 20 or 40 DS

The directory file size must be 12 or greater.

Normal characteristics for a DS diskette are:

```
Media = 5" FLOPPY,
Type = SINGLE DENSITY,
Number of sides = 2,
Step code = 0,              {0 = 6ms, 1 = 12ms, 2 = 20ms, and 3 = 30ms}
Directory cylinder = 17,
Sectors per granule =  5,
Granules per cylinder = 4,
Directory file size = 18.

Media = 5" FLOPPY,
Type = DOUBLE DENSITY,
Number of sides = 2,
Step code = 0,            {0 = 6ms, 1 = 12ms, 2 = 20ms, and 3 = 30ms}
Directory cylinder = 17,
Sectors per granule =  6,
Granules per cylinder = 6,
Directory file size = 32.
```

Before we continue with SYSGEN/CMD, let's understand more about floppy diskettes:

The minimum unit in reading from and/or writing to a floppy diskette is one sector. A sector is 256 contiguous bytes of data on the same track.

The minimum diskette space allocated for a file is one granule. A granule is 1 to 64 contiguous sectors. A granule may span more than one track and/or cylinder.

For **SS** diskettes, the track size is the same as the track capacity of the disk drive. For **DS** diskettes, the track size is twice the disk drive's track capacity. The disk drive's track capacity is the physical cylinder size of the floppy diskette. **DS** diskettes have twice as many tracks as it has cylinders. We can conclude from this that a cylinder is one read/write position of the disk drive's head, and a track is one recording surface. (This definition is used for rigid drives also, e.g., a two-platter, four-head rigid drive, has four tracks per cylinder). The number of tracks per cylinder is the number of heads the operating system is using per cylinder.

What about PD?

PD uses logical tracks. NEWDOS/80 defines a logical track as a lump. A logical track or lump consists of a multiple of whole granules. And, a logical track can span more than one track and/or cylinder. The operating system handles the physical conversion to properly locate the read/write head. Model I system diskettes require track zero, sector zero to be formatted in single-density. Model III system diskettes require track zero, sector one to be formatted in double-density. MULTIDOS formats the entire

track zero in single-density for Model I **PD** diskettes. However MULTIDOS can read **PD** diskettes with track zero formatted in either single or double-density. NEWDOS/80 can create **PD** diskettes with track zero formatted in either single-density or double-density (Model III NEWDOS/80). Nevertheless, a **PD** diskette with track zero formatted in a different density must render track zero inaccessible during normal operation (track zero is only recognized during boot).

Why PD?

DBLDOS, the original double-density operating system for the TRS-80®, published by Percom, was apparently designed to minimize the tear-up of NEWDOS/21 (NEWDOS/21 is a modified version of TRSDOS® 2.1) and maximize compatibility with existing software. Actually DBLDOS only modified SYS0/SYS and BASIC/CMD (BASIC/CMD was modified to disable several of the NEWDOS/21 enhancements to BASIC). The major modification to SYS0/SYS was switching between the single-density and double-density controllers and enabling the system to have 192 logical tracks. Effectively, the only required change was to the file SYS0/SYS. You can see why this method was used —— go double-density with ONE file changed. Also, **PD** yields a smaller granule (five sectors) and is much more efficient (even faster) for a system diskette. The most efficient MULTIDOS Model I **SS/DD** configuration is a **PD** system with **DD** data disks.

To make a Model I **PD** system diskette (single-sided only), format a diskette in **PD** and use SYSGEN/CMD to copy the required files to the newly formatted diskette. When SYSGEN/CMD asks for the PHYSICAL CYLINDERS, key in the track capacity of your drive —— this is probably 40 or 80 (the actual number of logical tracks is the number of physical tracks [less 1 if cylinder zero is single density] multiplied by 1.8, e.g., 35 track drive has 61 logical tracks, and a 40 track drive has 70 logical tracks).

If you want to make a Model I one-volume, double-density, double-sided system diskette, track zero must be formatted in single density in order for the Model I ROM to boot the system. The fastest way to accomplished this with MULTIDOS is to set the configuration to "(SI=2)", and format the diskette in double density. After a successful format, set the configuration to "(SI=1)" and format track zero single density using ZAP/CMD's format ("T" option).

i.e., (Assumes destination diskette is in drive 1)

```
CONFIG :1 (SI=2)<ENTER>
FORMAT :1<ENTER>          {you must input additional data for FORMAT}
CONFIG :1 (SI=1)<ENTER>
ZAP<ENTER>
<T><1><ENTER>
<0><ENTER>
<D><F3>
```

Now execute SYSGEN/CMD to copy the system files to the properly formatted diskette.

Now Back to SYSGEN/CMD:
After you have setup the DCT correctly, you will receive the query:

'How many PHYSICAL cylinders for the destination disk? ..'

Enter the number of TRACKS the drive is capable of handling.

The next query is:

'The destination disk is in PHYSICAL drive (0-7)? .'

Floppies are in physical drives 0 through 3 (this is asking you what position on the drive cable is the floppy to receive the system files).

The next query is:

'Do you want the DCT locked (N/Y)?

Press <ENTER>. Be certain a properly formatted diskette is in the destination logical drive when you press <ENTER> to the prompt:

'Press <ENTER> when the destination disk is in drive x'

SYSGEN/CMD copies all of the /DOL, /EXT, and /BOL files to the destination disk. The rest of the files should be copied via VFU in order to maintain the protection level and directory attributes.

If you want to create a system disk on a rigid drive, please follow the instructions that come with the rigid disk driver diskette.

SYSGEN/CMD enables you to create a NEWDOS/80 type of system diskette.

EXAMPLE: (double sided diskette for a 80 track drive)

```
Media = 5" FLOPPY,
Type = PSEUDO-DOUBLE DENSITY,
Number of sides = 2,
Step code = 0,
Directory cylinder = 35,
Sectors per granule =  5,
Granules per cylinder = 8,
Directory file size = 28.
```

This is a PDRIVE of: TI=A, TD=G, TC=80, SPT=36, GPL=8, DDSL=35, DDGA=6

MODEL 4

T4/CMD and TWO/IDO Double-sided, one-volume system diskette creator.

DO TWO<ENTER>

TWO/IDO is a do file for making a double-sided one-volume system diskette. This requires drive one to be capable of double-sided operations (i.e., you cannot make a double-sided diskette if drive one is single-sided). **Before you execute TWO/IDO, make sure the file T4/CMD is present on the system diskette.** To use TWO/IDO, place the master system diskette into drive zero, and another (doesn't necessarily have to be unformatted) diskette into drive one.

Key DO TWO<ENTER>. TWO/IDO executes and creates a double-sided one-volume system diskette in drive one (only the essential files are copied over to drive one, e.g., TWO/IDO and T4/CMD are not copied). At this point you can take your new double-sided one-volume system disk out of drive one and reboot it in drive zero (provided, of course, that drive zero is capable of double-sided operations).

You can BACKUP this new system diskette to make more, but remember: you have to CONFIG the drive that you're BACKUPing to so that it is a double-sided one-volume drive. CONFIG 1(SI=2) would do that.

MODEL I and MODEL III

TAPE/CMD Tape to disk transfer utility

TAPE<ENTER>

TAPE/CMD transfers a 500 baud contiguous system program from tape to disk. When executed,

'Enter "T" to read TAPE, or "D" to return to DOS? '

is displayed. Ready the tape deck, position the tape to the start of the program, and press <T>. If the program does not load properly or is not contiguous, an error message appears and the tape load terminates.

When the program loads successfully, indicated by:

'Is the PROGRAM "xxxxxx" to be modified? (Y or N)? _'

If you want the program to initially load at another address (e.g., to avoid clobbering DOS), then enter <Y> and

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

is displayed. Enter <E> if the interrupts are to remain enabled, or enter <D> if interrupts must be disabled to execute the program. Next

'Enter new base address in HEX? _'

is displayed. Enter an address greater than the START address. The next prompt:

'Initialize LEVEL II type DCB and RST vectors (Y or N)? _'

Respond <Y> if the program requires a LEVEL II environment. If the start address is below 5200H, I recommend a LEVEL II environment.

If you responded N to 'Is the PROGRAM "xxxxxx" to be modified? (Y or N)? _', or have completed the modification response, the following query appears:

'Is a new filename required (Y or N)? '

Enter <N> if you want to use the name of the program on tape with a /CMD extension. Or enter <Y> to receive this query:

'Filename please? __'

Enter the filename. Press <BREAK> at the nest prompt to exit TAPE/CMD.

VFU/CMD Versatile File Utility (MULTIDOS) or **FU/CMD** (File Utility for ESOTERIC).

[V]FU<ENTER>
[V]FU %<ENTER> {To access /DOL and /EXT files.}

VFU/CMD enables you to select files from a directory format to:
copy files from one disk to another,
move files from one disk to another, and
purge files.

VFU/CMD calculates the amount of free memory upon entry. If sufficient memory is available, VFU/CMD loads Open/DOL, Close/DOL, and Allocate/DOL into high memory, enabling you to copy, move, and/or purge without a system disk in drive zero. If insufficient memory is available, then the message:

'Insufficient MEMORY without a SYSTEM disk in drive zero.
<KEY> to continue.'

appears, telling you to keep a system disk in drive zero.

VFU/CMD is ready for your command when:

'Versatile File Utility 12.0 (c) 2004, V. B. Hester

Press	Action
<C>	Copy
<E>	Execute
<H>	Hard copy
<M>	Move
<P>	Purge

Choice _'

is displayed. Key your choice. <ENTER> defaults to the first choice, and <BREAK> exits VFU.

VFU - COPY COMMAND - <C> and/or VFU - MOVE COMMAND - <M>

The MOVE command performs the same as the COPY command except, for MOVE, the files on the source drive are purged after the files are copied to the destination drive.

The copy command can be used to copy files from TRSDOS® Model III double density diskettes to a non-TRSDOS® disk.

To copy files press <C> or <ENTER>. To move files press <M>. VFU/CMD responds with:

```
' Versatile File Utility 12.0 (c) 2004, V. B. Hester
  Copy

       Press     Condition
        <A>      Any condition.
        <C>      Create file on destination disk.
        <D>      Overwrite destination if dates are different.
        <O>      Overwrite destination file.
        <S>      Overwrite destination if sizes are different.
        <T>      Update destination to later version.

  Choice _'
```

This screen enables you to select the conditions to meet before file copying can take place. MAX-80, MODEL 4, and ESOTERIC: Pressing <←>, <↓>, or <→> returns you to main menu.

A = Any. You may also press <ENTER> for this condition.
C = Create only. Only copy/move file(s) if the filename does not exist on the destination disk. Used to avoid overwriting a file with the same name.
D = Different date. Only copy/move file(s) if the filename exists on the destination disk and has a different date. This can be used to undo an update, provided you kept an earlier version.
O = Overwrite. Only copy/move file(s) if the filename exists on the destination disk.
S = size difference. Only copy/move file(s) if the filename exists on the destination disk and has a different length (different EOF).
T = Time and date difference. This is used when you want to update a file that was created or updated earlier – either the same day or a prior date.

When the source diskette is Model III TRSDOS®, only conditions **A**, **C**, and **O** are valid.

The next prompt is:

```
'Wildcardmask >_'
```

Enter the wild card mask as defined for the LIBRARY command DIR. VFU/CMD is limited to five logical OR masks. Or press <ENTER> to avoid excluding any files. MAX-80, MODEL 4, and ESOTERIC: Pressing <←> in the first position returns you to main menu.

The next prompt is:

```
'<S> for selective, or <T> for total. _'
```

To select files to copy, press <S> or <ENTER>. For all files, press <T>.

If a wildcard mask was not specified then the next two prompts are:

'Include "INVISIBLE" files (N/Y)? _'

Press <N> or <Y> as appropriate (<ENTER> = <N>).

'Include "SYSTEM" files (N/Y)? _'

Press <N> or <Y> as appropriate (<ENTER> = <N>). Finally, VFU/CMD requests the source and destination drives with the following prompt:

'Source drive? _ Destination drive? _'

The source and destination drive cannot be the same. Press the number for the logical source drive, and press the number for the logical destination drive. ***[If you mis-key the source drive, key in the same number as the destination drive to receive the source drive prompt!]***

MAX-80, MODEL 4, and ESOTERIC: Pressing <←>, <↓>, or <→> returns you to main menu.

If the selective option was chosen, then the directory is displayed with a winking cursor next to the first filename. The arrow keys move the winking cursor in the arrow direction. If you want to copy the filename with the winking cursor, press <Y>. A marker appears in front of the filename, and the winking cursor moves to the next file. If you do not want to copy a file, press <N>, <SPACE>, or <→> and the cursor moves to the next filename. During the file selection process, <SHIFT·←> positions the cursor to the first filename removing all markers. To remove a particular marker, use the arrow keys to position the cursor over the marker and press <N> or <SPACE>. If you move beyond the last filename, press <←> to continue file selection.

If the total option was selected, then the directory is displayed with a marker in front of every filename. Use <←> to position to a file that may be un-selected with <SPACE>, or <SHIFT·←> to select all files.

The filenames with a marker next to them are the ones selected for copying.

After you have selected the files for copying, press <ENTER> to receive the prompt:

'<ENTER> to proceed, <A> to abort, or <R> to repeat. _'

Press <ENTER> to start the copy function. To terminate the copy function, before all files are copied, hold down one of the <SHIFT> keys until VFU/CMD completes copying the current file.

After the copy process is complete the prompt:

'<R> to redo, <I> to restart or <KEY> for resulting directory.'

is displayed. Press <R> to repeat the copying (usually to another destination diskette), <I> to go to the main menu or any key except <BREAK> to see the directory of the destination disk.

VFU - EXECUTE COMMAND - <E>

To execute a file, press <E>. The next two prompts are:

'Include "INVISIBLE" files (N/Y)? _'

Press <N> or <Y> as appropriate (<ENTER> = <N>)

'Include "SYSTEM" files (N/Y)? _'

Press <N> or <Y> as appropriate (<ENTER> = <N>). Finally, VFU/CMD requests the drive with the following prompt:

'Drive number? _'

MAX-80, MODEL 4, and ESOTERIC: Pressing <←>, <↓>, or <→> returns you to main menu.

Use the arrow keys to position the winking cursor in front of the file to be executed and press <Y>. The EXECUTE command commences when the <Y> is pressed. If the selected filespec does not have a /CMD extension, the EXECUTE command loads and executes BASIC/CMD, and attempts to run the filespec.

VFU - PRINT DIRECTORY COMMAND - <H>

To print a directory, press <H>. The next two prompts are:

'Include "INVISIBLE" files (N/Y)? _'

Press <N> or <Y> as appropriate.

'Include "SYSTEM" files (N/Y)? _'

Press <N> or <Y> as appropriate. Finally, VFU/CMD requests the drive with the following prompt:

'Drive number? _'

MAX-80, MODEL 4, and ESOTERIC: Pressing <←>, <↓>, or <→> returns you to main menu.

After the logical drive number is pressed, the prompt:

'Identification: _'

is displayed. Input your choice (8 characters maximum) to identify this disk. After you press <ENTER>, the directory is displayed on the screen. Finally, VFU/CMD prompts you with the following message:

'Press <A> to abort, <ENTER> to execute, or <R> to repeat. _'

A reply of <ENTER> prints the directory. If a 10 character per inch printer is used, the printout width fits inside a 5¼" diskette jacket.

VFU - PURGE COMMAND - <P>

To purge files from a disk, press <P>. The next prompt is:

```
'Versatile File Utility 12.0 (c) 2004, V. B. Hester
 Remove

 Wildcardmask >_'
```

Enter the wildcard mask as defined under the LIBRARY command DIR, or press <ENTER> to avoid excluding any files.

The next prompt is:

```
'<S> for selective, or <T> for total. _'
```

To select files to purge, press <S> or <ENTER>. For all files, press <T>. If a wild card mask was not specified, then the next two prompts are:

```
'Include "INVISIBLE" files (N/Y)? _'
```

Press <N> or <Y> as appropriate (<ENTER> = <N>).

```
'Include "SYSTEM" files (N/Y)? _'
```

Press <N> or <Y> as appropriate (<ENTER> = <N>). Finally, VFU/CMD requests the drive with the following prompt:

```
'Drive number? _'
```

Press the desired logical drive for file purging.

MAX-80, MODEL 4, and ESOTERIC: Pressing <←>, <↓>, or <→> returns you to main menu.

Files to be purged are selected similar to files to be copied. After you have selected the files for purging, press <ENTER> to receive the prompt:

```
'<A> to abort, <ENTER> to execute, or <R> to repeat. _'
```

Press <ENTER> to start the purge function. To terminate the purge function, before all files are purged, hold down one of the <SHIFT> keys until VFU/CMD completes purging the current file.

After the purge process is complete the prompt:

```
'<R> to redo, <S> to restart or <KEY> for resulting directory.'
```

is displayed. Press <R> to purge the same files (obviously from another diskette), <S> to go to the selection menu or any key except <BREAK> to see the directory of the destination disk.

ZAP/CMD Disk sector/memory review/modification utility.

ZAP<ENTER>

ZAP/CMD is a quick, simple way to copy disk sectors, read and modify disk sectors, read and modify file sectors, read and modify RAM, format a single cylinder, verify sectors, and fix MULTIDOS and ESOTERIC directories.

General ZAP/CMD information

DEFAULTS
When ZAP/CMD is waiting for your response, ZAP/CMD defaults to the first choice or lowest value if <ENTER> is pressed. Whenever an input field is complete, ZAP/CMD inserts an <ENTER> for you, i.e., if the query is "DENSITY (D/S) ." (one key press is required), pressing <ENTER> or <D> terminates the query and continues.

NUMBERS
Numbers are interpreted as decimal, unless an H is appended to the number. If a number is suffixed with an H, then the number is considered hexadecimal.

FILENAMES
Filenames are character case significant. i.e., zap/cmd is not the same as ZAP/CMD. However, you can force the filespec to upper case without using the <CAPS> key, by pressing the <↓> key at the Filespec prompt. Once <ENTER> is pressed, the case of keys reverts back to the keyboard setting.

BREAK KEY
Pressing <BREAK> returns you to the previous query. When the "Choice" prompt is displayed, pressing <BREAK> exits ZAP/CMD.

MODIFICATION
Pressing <M> after a displaying a page of MEMORY or a diskette SECTOR, puts you in the modification mode. This is noted by two transparent blinking cursors - one in the HEX area and the other in the corresponding ASCII area (right 16 bytes). The arrow keys moves both cursors in the arrow direction. The <@> key toggles you between the HEX and ASCII modification mode, as indicated by the presence of **Hex** or **Asc** in the upper left hand corner. Key the new data for the bytes to be changed. Two keystrokes are required for hexadecimal changes, and one keystroke for ASCII changes. If you are in the HEX modification mode, pressing <Z> zeroes all bytes from the cursor position to the end of the page. Memory modification is effective immediately. However, sector modification is not effective until <ENTER> is pressed to terminate modification and <ENTER> is pressed a second time to update the sector.

MAX-80, MODEL 4, and ESOTERIC
RETURN TO "Choice" PROMPT
Pressing <F1> returns you to the "Choice" prompt from anywhere in ZAP/CMD.

EXIT
In addition to pressing <BREAK> at the "Choice" prompt to exit, pressing <F3> from anywhere in ZAP/CMD also exits ZAP/CMD.

Copy sectors <C>

The ***Copy sectors*** option is obtained by pressing <C>. This option copies sectors from any disk MULTIDOS and/or ESOTERIC can read to any other disk MULTIDOS and/or ESOTERIC can read. However, you are cautioned to make sure the configuration byte for the source and destination disks have been set properly. These bytes are set correctly after a successful CAT[(T)] on both disks. The copy function automatically resolves overlaps when the same disk is used for the source and destination (in the same drive). ***Copy sectors*** cannot copy sectors between two different disks to be mounted in the same drive.

Disk sectors <D>

The ***Disk sectors*** option is obtained by pressing <D>. This option steps through a single density, double density, or "P" density diskette, provided the target diskette has had its respective configuration byte updated (this byte is correct after a successful CAT[(T)]. This includes TRSDOS® double density diskettes (Model I or Model III) and NEWDOS 80 diskettes. The selection of the first sector is menu driven; additional sectors are displayed by pressing:

Key pressed	Action
<↑>	Increase cylinder by one
<↓>	Decrease cylinder by one
[<SHIFT>]<→>	Increase sector by one
[<SHIFT>]<←>	Decrease sector by one
<C>	Re-select cylinder/sector
<T>	Re-select cylinder(track)/sector
<S>	Re-select sector

Form the ***Disk sectors*** option, you can direct ZAP to copy a sector by pressing <CTRL·C>

Directory entry <E>

The ***Directory entry*** option enables you to easily locate the directory sector that contains the target filename. The filename to be entered exactly as the file name appears in the directory (character case significant). Enter the target file name at the

'Filespec ....'

prompt. If ZAP/CMD finds the directory sector of the file, ZAP/CMD is in the ***Disk sectors*** option.

File sectors <F>

The ***File sectors*** option is obtained by pressing <F>, and accesses files, regardless of protection, density, and/or the operating system (except TRSDOS® Model I double density). The filename to be entered exactly as the file name appears in the directory (character case significant). Enter the target file name at the

'Filespec ....'

prompt. ZAP/CMD finds the file, then

'Relative sector in xxxxxxxx/xxx:x (0 to ddddd/hhhhH) ....'

is displayed. The xxxxxxxx/xxx:x is the target file name, and ddddd/hhhhH is the highest relative sector in the target filename in decimal (ddddd) or hexadecimal (hhhhH). A number greater than ddddd/hhhhH displays the highest relative sector. Enter the desired relative sector. A file's first sector is relative sector zero; therefore, just <ENTER> displays the first sector of the file. You can access other sectors in the target file with the arrow keys.

Key pressed	Action
<↑> or [<SHIFT>]<→>	Increase file sector by one
<↓> or [<SHIFT>]<←>	Decrease file sector by one

You can convert from ***File sectors*** to ***Disk sectors*** by pressing <CTRL·C>.

Memory <M>

The ***Memory*** option is obtained by pressing <M>. When the "Address ....." prompt appears, enter the desired address in decimal or hexadecimal with an H suffix. The memory address is changed by pressing the arrow keys.

Key pressed	Action
<↑>	Increase address by 256/100H
<↓>	Decrease address by 256/100H
<→>	Increase address by one
<SHIFT·→>	Increase address by 16/10H
<←>	Decrease address by one
<SHIFT·←>	Decrease address by 16/10H

Format Cylinder <T>

The ***Format Cylinder*** option is obtained by pressing <T>. This option formats and verifies a single cylinder on a floppy diskette. Don't be quick on the draw. The format density requires only one keystroke! Formatting a cylinder wipes out all data on that cylinder.

Verify <V>

The ***Verify*** option is obtained by pressing <V>. This option determines if the target cylinders, sector by sector, are readable. If the target diskette has mixed density tracks, ***Verify*** switches densities automatically. ***Verify*** uses logical cylinders (LUMPS) when verifying a double density and/or double sided NEWDOS/80 diskette.

Check/Repair directory <X>

The ***Check/Repair directory*** option is obtained by pressing <X>. This option is designed to fix the directories for MULTIDOS and ESOTERIC diskettes. ***Check/Repair directory*** also fixes the directory for alien diskettes; however, the data placed on the GAT sector may not be compatible with the alien diskette's operating system. If you fix the directory for an alien diskette, be sure you have the necessary information to modify the data on the GAT sector if the data was changed by ***Check/Repair directory***.

DIRECTORY STRUCTURE

Directory structure

The directory, usually named DIR/SYS, has information that details the disks usage, filenames, location, length, protection level, logical record length, and password hash values for all files on the disk. Each directory has three separate but integral sections:

1. The first sector, Granule Allocation Table (GAT), has the data on free and allocated space on the disk.
2. The second sector, Hash Index Table (HIT), has the hash codes for each active file on the disk.
3. The balance of the sectors contains the directory records (DIREC) for each file. Each DIREC is 32 bytes in length. The number of directory records varies with the density and number of sides for each disk.

GAT Organization

The granule allocation table is located on the first sector of the directory, containing information on disk space assignment. Disk space assignment is performed in a unit called a granule. A granule is a subdivision of a LOGICAL track that consists of a contiguous group of whole sectors. A PHYSICAL track is one of many concentric circular recording surfaces on a side of a diskette and/or a platter in a rigid drive. A cylinder is the head position for all identical numbered tracks in a given drive. A cylinder may have one track for single sided floppy diskettes, eight tracks for a four platter rigid disk, six tracks for a three platter rigid disk, two tracks for a double sided floppy diskette, etc. The number of tracks for a cylinder varies with the hardware configuration of the disk drive. However, the number of granules per LOGICAL track is fixed in accordance to the table below.

Diskette	Granules/Track	Granules/Cylinder	Sectors/Granule
SS/SD	2	2	5
DS/SD	2	4	5
SS/DD	3	3	6
DS/DD	3	6	6

SS = single-sided, **DS** = double-sided, **SD** = single-density, **DD** = double-density

TABLE 1

The GAT is divided into two tables. Bytes X'00' through X'5F' are the assignment/free table, that corresponds to an individual cylinder on the disk. Each bit within the byte is used to indicate which relative granules within the cylinder is assigned or free. A reset bit indicates a free granule, and a set bit indicates an assigned granule.

```
    Hex  00 FFFF FFFF FFFF FFF8 F8FF FFFF FBFF FFFF ................
         10 FFFF FFFF FFFF FFF8 F8F8 F8F8 F8F8 FAF8 ................
         20 F8F8 F8F8 F8F8 F8F8 FFFF FFFF FFFF FFFF ................
    Drv  30 FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
     2   40 FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
         50 FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
         60 F8F8 F8F8 F8F8 F8F8 F8F8 F8F8 F8F8 F8F8 ................
    Cyl  75 F8F8 F8F8 F8F8 F8F8 F8F8 F8F8 F8F8 F8F8 ................
     17  80 F8F8 F8F8 F8F8 F8F8 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 ................
     00  C0 FFFF FFFF FFFF FFFF FFFF FFFF 0542 E042 ..............B.B
    00H  D0 4D55 4C54 4944 4F53 3130 2F30 312F 3835 MULTIDOS04/01/89
         E0 0D20 2020 2020 2020 2020 2020 2020 2020 .
    DM   F0 2020 2020 2020 2020 2020 2020 2020 2020
```

Figure 1

Refer to the GAT table in figure 1.

Relative byte X'0C' has the value X'FB' (11111011). The bits set are 7,6,5,4,3,1,0. Therefore, granule 2 is free.

Relative byte X'1E' has the value X'FA' (11111010). The bits set are 7,6,5,4,3,1. Therefore, granules 2 and 0 are free.

Bytes X'60' through X'BF' are the available/locked out table, that corresponds to a relative cylinder. The available/locked out table indicates which granules are locked out during formatting. A reset bit indicates an available granule, and a set bit indicates a locked out granule. All granules locked out during formatting are also assigned in the assignment/free table.

Hard disks and "P" density diskettes do not have an available/locked out table. If you use NEWDOS/80 diskettes (a form of "P" density) some have this table and others do not. If the number of LUMPS is greater than 96, then the balance of the available/locked out table should be set to FF.

Bytes X'C0' through X'CB' extends the assignment/free table from X'00' through X'CB' for hard disks, and are set to X'FF' for floppy diskettes.

Byte X'CC' contains the number of logical cylinders in excess of 35.

Byte X'CD' contains information about the diskette. (floppies)

```
BIT 7   1 = DATA diskette.    0 = SYSTEM diskette.
BIT 6   1 = Double density.   0 = Single density.
BIT 5   1 = Double sided.     0 = single sided.
BIT 0-2 is one less than the number of granules per cylinder.
```

For "P" density diskettes and/or NEWDOS/80 diskettes updated with CAT :d(X), byte X'CC' contains the number of logical cylinders (LUMPS), and byte X'CD' contains the number of physical cylinders for the disk.

Bytes X'CE' and X'CF' contain the disk's master password hash code.

Bytes X'D8' through X'D7' contain the disk name, and bytes X'D8' through X'DF' contain the disk date.

Bytes X'E0' through X'FE' stores a system's disk AUTO command, if any. If byte X'E0' is X'0D', then there is no AUTO command.

HIT Organization

The Hash Index Table (HIT) is found on the second sector of the directory. The HIT is used to store, in a position that corresponds to the DIREC, the hash code for each active file in the directory. The hash code is computed by creating an 11 byte work space, eight for the filename (left justified and padded with spaces) and three for the extension (left justified and padded with spaces). This work space is processed through a hashing routine that produces a one byte value of X'01' through X'FF'. The HIT reduces the time required to access a file, through the operating system, by eliminating the need to check byte for byte each filename in the directory. After the system matches the hash code, the corresponding DIREC is examined for a byte for byte match of the filename/ext.

DIREC Organization

The third through the last directory sector contain 32 byte directory records (DIREC). The maximum number of directory records is 256, regardless of the number of sectors allocated to the directory. (The HIT storage is limited to 256.)

A file's DIREC sector can be displayed with ZAP/CMD. This option is obtained by pressing <E>, and requires the filename to be entered exactly as the file name appears in the directory (character case significant). Enter the target file name at the

```
'Filespec ..............'
```

prompt. If ZAP/CMD finds the file, then the sector with the filespec's DIREC is displayed and you are in the ***Disk sectors*** option.

DIREC+00 This byte contains several attributes of a file.

Bit 7 If this bit is zero, then this DIREC is the file's primary directory entry (FPDE). If this bit is one, then this DIREC is the file's extended directory entry (FXDE). Extended entries are required when a file has more than four segments. The maximum granules per segment is 32; therefore, any file with more than 128 granules will have n extended directory entry.

Bit 6 Overlay files must have this bit set.

Bit 5 If this bit is set, then VFU requires the % parameter to access.

Bit 4 This is the bit to indicate if a file is active or not, regardless of the other bits in DIREC+00, and/or bytes in the DIREC. If this bit is set, then the file is active. REMOVE resets this bit and removes the corresponding HIT byte. RESTOR resets this bit and recreates the corresponding HIT byte.

Bit 3 If the bit is one, then the filename is hidden to directory display and/or print unless the invisible parameter, (I), is specified.

Bits 2 through 0 - Contain the access protection level of the file.

111 = NONE
110 = EXEC
101 = READ
100 = WRITE
011 = not used
010 = RENAME
001 = REMOVE
000 = full

DIREC+01 and DIREC+02

These two bytes contain the hash code for the access password.
In a FXDE, DIREC+01 is the reverse pointer to the previous FXDE or FPDE.

DIREC+03

The value of this byte indicates how many bytes the file extends into its last record.

DIREC+04

This byte contains the file's logical record length.

DIREC+05 through DIREC+12

These eight bytes contain the filename. The filename is left justified and padded with spaces.

DIREC+13 through DIREC+15

These three bytes contain the file's extension, if any. The extension is left justified and padded with spaces.

DIREC+16 and DIREC+17

These two bytes contain the date of the last update to the file. DIREC+16 bits 6 through 1 are the year; DIREC+16 bit 0 and DIREC+17 bits 7 through 5 are the month; DIREC+17 bits 4 through 0 are the day.

DIREC+18 and DIREC+19

These two bytes contain the time of the last update to the file. DIREC+18 bits 7 through 3 are the hour; DIREC+18 bits 2 through 0 and DIREC+19 bits 7 through 5 are the minutes; DIREC+19 bits 4 through 0 are the seconds divided by two.

DIREC+20 and DIREC+21

These two bytes contain the number of physical records in the file in LSB/MSB order.

DIREC+22 through DIREC+29

These bytes, used as pairs, form four segment fields. If the first byte is an X'FF', then the file is contained in the previous segment fields. (If DIREC+22 = X'FF', then the file uses no disk space.) If the first byte is X'00' through X'FD' (X'FE' is not used), then it represents the cylinder this file segment starts on. The second byte has the starting granule (on the cylinder in the first byte) in bits 7-5, and the number of contiguous granules less one in bits 4-0.

DIREC+30 and DIREC+31

These bytes point to a FXDE if DIREC+30 is a X'FE', otherwise DIREC+30 must be a X'FF'. The pointer, DIREC+31, has the relative file entry offset in bits 7 through 5, and the relative directory file sector in bits 4 through 0. Directories normally start on physical sector 0; therefore, physical sector 2 is relative directory file sector 0.

NOTE: DIREC+02 through DIREC+21 are not used for FXDE's.

DOS ERRORS

HEX	DECIMAL	ERROR MESSAGE
00	0	No error.
01	1	undefined error in MULTDOS and ESOTERIC
02	2	undefined error in MULTDOS and ESOTERIC
03	3	Lost data during read.
04	4	Parity error during read.
05	5	Data record not found during read
06	6	Attempted to read system data record.
07	7	HARD DISK ERROR.
08	8	Drive not available.
09	9	undefined error in MULTDOS and ESOTERIC
0A	10	Due to memory usage, file can't be extended. (MODEL I/III)
0B	11	Lost data during write.
0C	12	Parity error during write.
0D	13	Data record not found during write.
0E	14	Write fault on disk drive.
0F	15	Write protected media.
10	16	Drive not available.
11	17	Directory read error.
12	18	Directory write error.
13	19	Improper file name.
14	20	Granule allocation table read error.
15	21	Granule allocation table write error.
16	22	Hash index table read error.
17	23	Hash index table write error.
18	24	File not in specified directory.
19	25	File access denied.
1A	26	Directory space full.
1B	27	Disk space full.
1C	28	End of file encountered.
1D	29	No record found.
1E	30	Full directory. File can't be extended.
1F	31	File not found.
20	32	Improper drive number specified.
21	33	undefined error in MULTDOS and ESOTERIC
22	34	Load file format error.
23	35	Memory fault.
24	36	Attempted to load ROM memory. (MODEL I/III)
25	37	Access attempted to protected file.
26	38	File has not been opened.
27	39	Drive configuration one volume.
28	40	Drive configuration error.
29	41	Corrupt DIREC structure.

When an error occurs involving the directory, MULTIDOS and ESOTERIC generate two error messages. For example, if the disk in drive zero is write protected and an AUTO command is issued, MULTIDOS and/or ESOTERIC responds with:

```
Granule allocation table write error.
Write protected media.
```

Typically, you would use the second error message to resolve the problem.

Possible causes/Descriptions of errors:

Access attempted to protected file. - ACCESS password given, but UPDATE password required.

Attempted to read system data record. - Disk read of a directory cylinder without going through the MULTIDOS or ESOTERIC directory read routine.

Corrupt DIREC structure. - Link to an extended directory record is wrong. Or, primary directory entry has a link to an extended directory record, and it doesn't require one.

Data record not found during read/write. - The track, side, or sector number is not found. A flawed diskette causes this error, or a diskette formatted with different tracks per inch value than the drive it is in.

Directory read/write error. - A read/write operation, involving the directory, failed. Refer to the second error message.

Directory space full. - All DIREC slots are used. See ZAP/CMD for DIREC detail.

Disk space full. - All granules have been assigned.

Drive configuration error. - The DCT's configuration disagrees with the disk's.

Drive configuration one volume. - Side one selected for a logical drive with a single volume configuration.

Drive not available. - Diskette not spinning in the selected drive.

End of file encountered. - An attempt is made to read 1 unit past the EOF.

File access denied. - Incorrect password given.

File has not been opened. - An I/O attempt was made to an unopened file.

File not in specified directory. - Filespec is not in the specified directory.

Full directory. File can't be extended. - A file is being expanded and requires an extended entry; however, there are no DIREC slots available.

Granule allocation table read/write error. - A read/write operation, involving the GAT, failed. Refer to the second error message.

Hash index table read/write error. - A read/write operation, involving the HIT, failed. Refer to the second error message.

Improper drive number specified. - The drivespec is not in the range of 0 through 7.

Improper file name. - Filespec syntax incorrect or no filespec given.

Load file format error. - Load routine didn't know where to put the file.

Lost data during read/write. - Disk I/O loop too slow, or hardware problem.

Memory fault. - Load routine detected a bad RAM byte or no RAM.

No record found. - An attempt is made to read > one unit past the EOF.

Parity error during read/write. - A read/write failed (flawed disk).

File not found. - Filespec is not on any mounted disk.

Write fault on disk drive. - Definitely a hardware problem.

Write protected media. - A write attempt to a write protected disk.

BASIC High level language interpreter.

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

ff = number of file buffer areas allocated (0-15) default 3
V = user defined random I/O flag
mm = memory protect address.
command = Any valid SUPERBASIC command.

If any of the optional parameters are specified a mandatory space is required after BASIC. A comma is interpreted as a separator for a multiple command.

EXAMPLES:

BASIC<ENTER>

SUPERBASIC loads with 3 file buffer areas allocated (289 bytes for each file buffer area) and RAM available up to TOPMEM.

BASIC 4<ENTER>

SUPERBASIC loads with 4 file buffer areas allocated and RAM available up to TOPMEM.

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

SUPERBASIC loads with 2 expanded file buffer areas allocated (545 bytes each), capable of handling user defined record lengths (1 to 256), RAM from 60000 up is not available to SUPERBASIC, and the program PROG1/BAS is loaded by SUPERBASIC and executes starting with the first program line. If you are expecting to use direct-access files with a logical record length other than 256, then you must specify the V parameter.

The MODEL I and MODEL III versions have an enhanced version of BASIC - BBASIC - for program development. BBASIC is SUPERBASIC with BOSS. All the features of BASIC are included in BBASIC.

The MODEL 4 and ESOTERIC versions have a high resolution (Hi-RES) version of BASIC - BASICH - to use with either the Radio Shack and/or Micro-Labs high resolution board. The enhanced features are only available with the additional hardware. All the features of BASIC are included in BASICH.

BASIC * Recover SUPERBASIC 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 arguments intact. CAUTION: You cannot go from SUPERBASIC to DOS, execute commands such as DIR and then use BASIC * to return. In such cases you should use the CMD"fffff" command from within SUPERBASIC.

MODEL I and MODEL III: If the return to SUPERBASIC is successful, then 'Continue?' appears. Enter 'Y' if you want the program to continue.

MODEL I and MODEL III
BASIC ! Capture a BASIC program from RAM.

BASIC !<ENTER>

This unique command transfers a non-SUPERBASIC program in RAM to SUPERBASIC, sets file buffer area to zero, and maintains protected memory. This command usually follows a reset with MULTIDOS in drive zero. I recommend that you hold down the <ENTER> key to suppress any AUTO command. NOTE: You should use a MULTIDOS system diskette that does not have an invincible AUTO command.

MODEL I and MODEL III
BASIC # Recover a LEVEL II or MODEL III BASIC program from RAM.

BASIC #<ENTER>

MODEL I/III MULTIDOS uses CMD"W" from within SUPERBASIC instead of BASIC2 in DOS to enter LEVEL II/MODEL III BASIC. And, if a BASIC program is in SUPERBASIC, then the SUPERBASIC programs is transferred to LEVEL II/MODEL III BASIC via CMD"W". Once in LEVEL II/MODEL III BASIC (to run the BASIC program), you key SYSTEM<ENTER> and answer the *? Prompt with /16480<ENTER> to re-boot a MODEL I/III MULTIDOS system diskette in drive zero. I recommend that you hold down the <ENTER> key to suppress any AUTO command. NOTE: You should use a MULTIDOS system diskette that does not have an invincible AUTO command. At the DOS command prompt key BASIC #<ENTER>. Now the LEVEL II/MODEL III BASIC program is in SUPERBASIC.

Overlay location of BASIC commands and statements that are not in BASIC/CMD:

Command or Statement	Model I and Model III	MAX-80, Model 4, and ESOTERIC
; or @	CREF/BOL	CREF/BOL
B	n/a	RESOLVE/BOL
F	UTIL/BOL	UTIL/BOL
G	EDIT/BOL	EDIT/BOL
M	UTIL/BOL	UTIL/BOL
N	UTIL/BOL	UTIL/BOL, RENUM/BOL
O	RENUM/BOL	RENUM/BOL
CMD"C"	UTIL/BOL	UTIL/BOL
CMD"D"	Debug/DOL	Debug/DOL
CMD"O"	UTILA/BOL	n/a
CMD"P"	PACK/BOL	PACK/BOL
CMD"Q"	UTIL/BOL or UTILA/BOL	SORT/BOL
CMD"U"	UNPACK/BOL	UNPACK/BOL
CMD"V"	UTIL/BOL	UTIL/BOL
CMD"W"	PACK/BOL	n/a
CMD"X"	UTIL/BOL, EDIT/BOL, RENUM/BOL	UTIL/BOL, EDIT/BOL, RENUM/BOL
CMD"Y"	n/a	UTIL2/BOL
CMD"fffff"	UTIL/BOL or UTILA/BOL	UTIL/BOL
SORT	n/a	SORT/BOL

The following manual text applies to all models unless specifically restricted by indicating the applicable model(s) (usually preceding the manual text). ESOTERIC BASIC is named ESOBASIC; and, ESOBASIC is similar to MODEL 4 SUPERBASIC.

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·↑>            = list first program line
    MODEL I and MODEL III
    <SHIFT·→>            = list last program line
    MAX-80, MODEL 4, and ESOTERIC
    <SHIFT·↓>            = list last program line
```

Single keystroke commands are recognized only in the first position after the command prompt, >, appears. MODEL I and MODEL III: If you have pressed any other key(s), you must press <BREAK> instead of <←> to position the cursor to the first position. MAX-80, MODEL 4, and ESOTERIC: You may use <←> to position the cursor to the first position.

SINGLE CHARACTER COMMANDS

```
         ; = cross-reference (MODEL 1 and MODEL III)
         @ = cross-reference (MAX-80, MODEL 4, and ESOTERIC - CUSTOM/CMD can make it ;)
         A = AUTO                {the MODEL I and MODEL III have reserved word AUTO}
         B = resolve laBels
         C = CONT
         D = DELETE              {the MODEL I and MODEL III have reserved word DELETE}
         E = EDIT                {the MODEL I and MODEL III have reserved word EDIT}
         F = Find ASCII string
         G = Global editor
         I = Insert              {= A[UTO] current line +1, 1}
K"filespec = KILL"filespec"      {see KILL}
         L = LIST
L"filespec = LOAD"filespec"      {see LOAD}
         M = Move a single line
         N = duplicate a line. i.e., make a New line like an existing line
         O = renumber (re-Order line sequence)
         P = list Page           {= CLS:LIST current line - current line +x}
         R = Run program
R"filespec = RUN"filespec"       {see RUN}
S"filespec = SAVE"filespec"      {see SAVE}
```

The above commands are terminated with <ENTER> and are not case sensitive. i.e., either **g**<ENTER> or **G**<ENTER> invokes the global editor.

@ (MAX-80, MODEL 4, and ESOTERIC) Cross reference arguments and integers.
; (MODEL 1 and MODEL III)

@[p][[#]xxx]<ENTER>

p = listing directive. If p = * the reference listing is to the display. If p = $ then the reference listing is to the printer and the display.
xxx = Reference target:
(1) A one and/or two character variable name without a type suffix.
(2) An integer number which may be a line number and/or a value used in the program.
(3) A key word if preceded by a # symbol.

Option (1) lists all line numbers which contain the variable specified. Option (2) lists all line numbers and other references to the integer specified. Option (3) lists all line numbers which contain the key word. After a reference target has been established, the reference lines are displayed sequentially with each additional press of **@**<ENTER>.

If the **p** option is used with an **xxx** target, then a reference listing is produced starting with the target and proceeding in ascending order. Use of the **p** option without an **xxx** target results in a reference listing of all integer numbers and arguments. If the reference listing is requested in the form **@p#**<ENTER>, a listing of key words is produced. The key word listing is produced in token value order. This is not alphabetical order.

To pause during a reference listing, press <SHIFT>@. To resume the listing, press any key except <SHIFT>@. To stop the listing, press <BREAK>.

When the line number containing the reference is displayed, it may be followed by one or more of the following modifiers:

/n where n = the number of references to the target in the 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 #.
(n the variable is used as an array variable in this line.

If n is has a value of one it is not displayed.

EXAMPLES:

@*<ENTER>
All integer and variable references are displayed on the screen.

@K<ENTER>
All line numbers that contain the variable K are displayed on the screen. The total occurrences of the variable K is printed on a new screen line, after the line numbers, following a >.

```
@K
      K   11/3 36/5 133/6 135 136/5 138/2 181/2 202/4 203
              >29     {K occurs 29 times}
```

@$G<ENTER>
All arguments starting with G and continuing through ZZ will have their references directed to the video and line printer. The printer output is suppressed if the printer is routed to a file.

@#PRINT<ENTER>
All line numbers which contain the key word PRINT are displayed on the screen. The total occurrences of the key word PRINT is printed on a new screen line, after the line numbers, following a >.

@$#<ENTER>
All key words will have their references directed to the video and the line printer. The printer output is suppressed if the printer is routed to a file.

A

Automatic line numbering. {MODEL I/III also have reserved word AUTO.}

A[start][,increment]<ENTER>

start = starting line number, default 10
increment = the number between each line, default 10

The **A** command prints a line numbers waiting for you to provide the BASIC line data terminated with <ENTER>. If no data is provided for a line:
MODEL I and MODEL III:
SUPERBASIC does not inset the line into BASIC text area.
MAX-80, MODEL 4, and ESOTERIC:
SUPERBASIC leaves the blank line number in the BASIC text area.

After each line is entered, this command adds the increment to the previous line and waits for your input. If you have no additional data to input, press <BREAK> to exit the A command.

MODEL I and MODEL III:
If a line exists for the new line number, then a ***** appears immediately behind the line number. e.g., 120***** To avoid overwriting this line number, press <BREAK>.
MAX-80, MODEL 4, and ESOTERIC:
If a line exists for the new line number, then a **!** appears immediately behind the line number. e.g., 120**!** To avoid overwriting this line number, press <BREAK>.

The A command changes a program.

MAX-80, MODEL 4, and ESOTERIC

B RESOLVER - Resolve labels {changes program, and CLEARs 50}

B[*]<ENTER>

B<ENTER> is for error checking. **B***<ENTER> resolves the labels.

RESOLVER resolves, changes to line numbers, all labels in the resident BASIC program. In addition, all label definitions are removed. If a reference error is in the resident BASIC program, RESOLVER displays all of the errors it finds and does not resolve any labels. Error types are:

llll/nnnnn/U	Line number llll has line nnnnn undefined.
llll/"label"/U	Line number llll has label "label" undefined.
llll/S	Line number llll contains a syntax error.
llll/O	Line number llll contains a line number > 65535.

EXAMPLE:

```
20/"JILL"/U
```

Line 20 has a reference to the label "JILL" and there is no LABEL"JILL" in the resident BASIC program. The reference may be in the form of: GOTO "JILL", GOSUB "JILL", ...THEN "JILL", ...ELSE "JILL", RESUME "JILL", ON n GOTO "JACK","JILL", IF ERL = "JILL", etc.

C Continue program execution after STOP {MODEL I/III also have reserved word CONT.}

C<ENTER>

C resumes program execution if the program was stopped by pressing <BREAK> or the program encountered a STOP or END statement. Program resumption is not valid if the program is changed. Commands and statements that change a program are:

1. **A**<ENTER>
2. **B**[*]<ENTER>
3. a successful **D**[ln1]-[ln2]<ENTER>
4. **E** and <ENTER>. To exit Edit and return to command mode without changes press <Q>
5. **G**<ENTER>
6. **I**[ln0]<ENTER>
7. a successful **M**ln1,ln2<ENTER>
8. a successful **N**ln1,ln2<ENTER>
9. **O**[new][,inc][,start][,end]<ENTER>
10. **CMD"C"**<ENTER>
11. processed **CMD"P"**<ENTER>
12. **CMD"U"**[,C]<ENTER>
13. **CMD"X"**<ENTER>
14. **CHAIN** var$ (without the B option)
15. **CLEAR** {see CLEAR for differences between MODELS I/III and MAX/MODEL4/ESOTERIC}
16. **LOAD** var$ (without the B option)
17. **MERGE** var$
18. **NEW**
19. **RUN** var$ (without the B option)

D

Delete program line(s) {MODEL I/III also have reserved word DELETE.}

INPUT	LINES DELETED
D<ENTER>	the current line
D ln1 - ln2<ENTER>	lines from ln1 to ln2
D - ln2<ENTER>	the first line to ln2

MAX-80, MODEL 4, and ESOTERIC:

D ln1 -<ENTER>	lines from ln1 to the end of the program
D -<ENTER>	the entire program

If **ln1** does not exist, then the first line deleted is the next highest line greater than **ln1**. MAX-80, MODEL 4, and ESOTERIC: If **ln2** does not exist, then the last line deleted is the highest line less than **ln2**. At least one line must exist between **ln1** and **ln2**; otherwise, an "Improper parameter used." error results. MODEL I and MODEL III: ln2 must exist; otherwise, an "Improper parameter used." error results.

Note: In SUPERBASIC, entering a line number **LISTS** the line and does not delete the line.

The D command changes a program.

E

Edit a single program line {the MODEL I and MODEL III have reserved word EDIT}

E[ln]<ENTER>

ln = program line to modify - default the current line

NOTE: **E**<ENTER> is the same as **E.**<ENTER>.

The E command changes a program.

F

FIND - Find ASCII characters including ones in remark statements and quoted text

F*tar*<ENTER>

The **F** command prints the line numbers for all occurrences of ***tar*** in the resident BASIC program. If more than one occurrence is in a line, then a / is printed followed by the number of occurrences. Both leading and trailing spaces are recognized.

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 line 90 three times.

G Global editing (GEDIT) the resident BASIC program. {changes program}

G<ENTER>

GEDIT makes mass changes to the resident BASIC program, by performing the following:

Change all or part of argument.
Change all or part of constants, data list items, or strings.
Change CHR$(xxx) into packed strings.
Append adjacent line numbers into one long line.
Split one long line into two shorter ones.
Change key words.

GEDIT - General changes

G<ENTER>

The screen clears and the following prompt is displayed:

```
'L            T

 T = _'
```

This is the target entry mode. The rules for the target are discussed after the replacement is defined. After a target is entered, you are prompted to enter Line A, which is the first line to search and has a default value of the first program line. The next prompt is 'Line B', which is the last line to be searched and has a default value of the last line in the program. Line A and Line B limit changes to a specific range of program lines. The next prompt 'R =' is the replacement entry mode for the target specified. To re-enter an erroneous target, press <ENTER> for the replacement, and GEDIT returns to the target entry mode. To delete all occurrences of a target, use <SHIFT><@> as the replacement.

```
'L        T

 T = A
 Line A , Line B
 R = H
 Use it?'
```

Finally, GEDIT prompts 'Use it?'. This is the last chance to correct an error in the entries. An <N> response puts the GEDIT in the replacement mode, and a <Y> response starts GEDIT to search the program for the target and make the changes. The screen shows the line number being searched under the L and the last line number where a target was found under the T. When GEDIT has made all of the changes, the total changes are displayed. To exit from GEDIT, at this point, press <BREAK>; otherwise, any other key restarts GEDIT.

If the replacement is larger than the target, the program increases in size by that difference for each occurrence. If you run out of memory, an “Insufficient MEMORY in nnnnn” error message is displayed, and all changes are made up to the point (somewhere in line nnnnn) where memory was insufficient.

Rules for target:

In the following examples, T is the target, R is the replacement and ==> means is changed to.

EXAMPLE: Change all occurrences of the variable B to F

T = B and R = F

B==>F, AB==>AF, BA==>FA, BB==>FF, B$==>F$, AB$==>AF$, CA is not changed.

To change only a single character variable, while leaving multi-character arguments with the same letter unchanged, enclose the target character in single quotes, e.g., 'X'.

EXAMPLE: 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) —— note that (AA) was not changed.

To change only the first character of a multi-character variable, enter the target character followed by a single quote, e.g., X'.

EXAMPLE: Change all occurrences of the first A to H

T = A' and R = H

AA==>HA, AB==>HB, AC==>HC, QD is not changed, AE==>HE, AA$==>HA$, BA$ is not changed.

To change the second or greater character of a multi-character variable, while leaving the first character unchanged, precede the target character with a single quote, e.g., 'X.

EXAMPLE: Change all occurrences of the second A to I

T = 'A and R = I

AA!==>AI!, BA==>BI, AA$==>AI$, A$(AA)==>A$(AI), A is not changed, AX is not changed.

To change a target which is enclosed in quotation marks, precede the target with the $ sign.

EXAMPLE: Change all E in strings to Z

T = $E, and R = Z.

"ABCDE"==>"ABCDZ", TE is not changed, "EASY"==>"ZASY"

MAX-80, MODEL 4, and ESOTERIC

The <WC> character is used in any target position if you don't care what's in that target position. For the replacement, the <WC> character leaves that position unchanged. e.g., T = <WC>T, and R = <WC>Q means change all arguments ending with T to now end with Q. **NOTE: <WC> is not available with HBASIC.**

GEDIT - Key word changes

GEDIT can change key words, such as PRINT to LPRINT. You must be careful, however, as lowercase is not acceptable. Under SUPERBASIC as each line is entered and/or edited, it is processed through a buffer. This buffer looks for key words and changes them to a one byte code. It also converts all arguments to uppercase. Do not use lowercase for key words and/or arguments because GEDIT does not process changes through this buffer. A syntax error will occur at run time. If the target is a key word and/or the arithmetic operators +, -, *, /, [, >, =, or <, then bracket the target with the < and > characters, e.g., <**TAB(**>, <**-**>, <**PEEK**>, <**/**>, <**[**>, <**PRINT**>, <**=**>, <**<**>, etc.

GEDIT - Building compressed strings

Programs containing many CHR$(xxx)+CHR$(xxx) statements, can be shorten by building compressed strings. To build a compressed graphics string in place of a CHR$(xxx) type lines, enter the + character for the target. To build a compressed string with space compression codes, enter the * character for the target. This results in faster execution and more free memory. As an example, the line:

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

requires 32 bytes of memory. After GEDIT builds 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. The CHR$(xxx) cannot contain blanks within the parentheses, i.e., CHR$(191) is not changed.

EXAMPLES:

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

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

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

For those who want to change any CHR$(xxx) to a one byte value, key an **=** after the **+** or ***** to adjust the xxx value down by 128. i.e., ***=** compresses CHR$(064) through CHR$(127). Note the required '0' before the '64'. GEDIT requires three-digit values. And do not compress a CHR$(000) [leave it as CHR$(0) and GEDIT will not change it].

GEDIT - Appending line numbers

GEDIT can append a program line to the preceding line in a program, disregarding references to the line. As an example, if a program originally contains lines 1,2,3,4, and 5, and you append line 3 to line 2, an error would result if line 5 contains GOTO 3 and you ran the program which executes line 5. Also do not append to a line which contains an open quote (10 A$="THIS IS). Close the quote first, then append (10 A$="THIS IS").

GEDIT's append can create lines greater than 255 bytes in length, that execute properly, but cannot be completely listed on the screen and/or edited. Use CMD"U" to separate the lines. To append, respond with the / character and the line number to be appended as the target.

EXAMPLE:

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

To append the above two lines, the target is /20. The result is one line:

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

GEDIT - Splitting lines

GEDIT can 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. However, if you assign a new line number greater than line number of the following line, run time errors will occur if the program has a references to the in between line. As an example, if a program contains lines 10, 20, 30, 40, 50, and 60, and you split line 20 into lines 20 and 45, lines 30 and 40 are not found by a reference instruction such as GOTO 30, GOSUB 40, etc. However, if the program flow is such that SUPERBASIC would normally process the next statement after the new lines 20 and 45, lines 30 and 40 will be processed. SUPERBASIC 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 split a line, enter the target in the form -ttt, where ttt is the target for the split. If the target is a key word such as PRINT, enclose the target with < and > signs. If no target is specified, then the line is split at the first occurrence of a colon. Line A now represents the line number to be split and line B represents the new line number.

EXAMPLE:

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

To split the above line, T = -, Line A = 10, Line B=15. The result is:

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

EXAMPLE:

```
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$="FGH"
35 PRINT B$
```

NOTE: Use of 'manual' appends and splits may change program's logic.

I

Automatic line numbering {changes program}

I<ENTER>
I[ln0]<ENTER> {MAX-80, MODEL 4, and ESOTERIC}

The **I** command presets the **A** command with the current line +1 as the starting line, and the increment to 1. MAX-80, MODEL 4, and ESOTERIC: You can put a line number after the **I** and this line number supersedes the current line. i.e., **I 23** is the same as **A 24,1**

L

List program line(s) onto display.

INPUT	LINES LISTED
L<ENTER>	the current line
L ln1 - ln2<ENTER>	lines from ln1 to ln2
L - ln2<ENTER>	the first line to ln2
L ln1 -<ENTER>	lines from ln1 to the end of the program
L -<ENTER>	the entire program

If **ln1** does not exist, then the first line listed is the next highest line greater than **ln1**. If **ln2** does not exist, then the last line listed is the highest line less than **ln2**. Note: In SUPERBASIC, entering a line number **LISTS** the line and does not delete the line.

M

Move a single program line {changes program}

M ln1,ln2<ENTER>

ln1 = line number to be moved
ln2 = location ln1 is moved

The **M** command relocates **ln1** to **ln2**, deleting **ln1**, and inserting it as **ln2**. If **ln2** existed prior to this command it is replaced by **ln1**. The **.** (period) may be used for **ln1** and/or **ln2** to refer to the current line. The references to the moved line are not updated. If you want to move a group of lines and/or update references to a single line, use the renumber command, **O**.

N

Duplicate program line(s) {changes program}

N ln1,ln2<ENTER>

ln1 = line number to be duplicated
ln2 = location where ln1 is duplicated

The **N** command duplicates **ln1** as **ln2** while leaving **ln1** intact. If **ln2** existed prior to this command, it is replaced by **ln1**. The **.** (period) may be used for **ln1** and/or **ln2** to refer to the current line. References remain with the original line - **ln1**.

MAX-80, MODEL 4, and ESOTERIC
The N command can duplicate a group of lines, provided there is no overlap.

N ln1-ln2,ln3<ENTER>

This command duplicates lines from **ln1** through **ln2** starting with **ln3** and incrementing by one. i.e., the first line of duplicate text is ln3, the second line is ln3+1, the third line is ln3+2, etc. In this construct, **ln3** cannot exist. References are not updated because the original lines remain. This construct uses the renumber overlay.

O

RENUMBER - Renumber the resident BASIC program. {changes program, and CLEARs 73}

O[nnn][,iii][,sss][,eee]<ENTER>

nnn = The first line number to be assigned to a renumbered line. **nnn** has a default value of 10.
iii = The increment to be used in renumbering. **iii** must be greater than 0, and has a default value of 10.
sss = The starting line number in the original program where renumbering is to start. **sss** has a default value of 0. This is the first line renumbered.
eee = The ending line number in the original program where line renumbering is to stop. This is the last line renumbered. **eee** must be => **sss** and has a default value of 65529 for the MODEL I and MODEL III. For the MAX-80, MODEL 4, and ESOTERIC the default value is 65534; and, RENUMBER does not renumber line 65535.

RENUMBER checks for missing and/or invalid line numbers in a program, recovers a NEWed program, and renumbers all or part of a program. RENUMBER displays single letters to indicate the status of renumbering: C = checking, R = renumbering, and M = moving. An E indicates error checking only. There is no status indicated when recovering a program immediately after a NEW.

Error types are:

lllll/nnnnn/U	Line number lllll has line nnnnn **U**ndefined.
lllll/S	Line number lllll contains a **S**yntax error.
lllll/O	Line number lllll contains an **O**verflow line number.

For error checking only or to recover a program immediately after a NEW, key o<ENTER>.

EXAMPLE:

```
20/9/U
```

Line 20 has a reference to line 9 and there is no line number 9 in the resident BASIC program. The reference may be in the form of: GOTO 9, GOSUB 9, ...THEN 9, ...ELSE 9, RESUME 9, ON n GOTO 8,9, IF ERL = 9, etc.

RENUMBER checks your program for errors before renumbering. If any errors are found they are displayed and the program is unchanged.

EXAMPLE:

```
        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
```

Any attempt to renumber this program, produces:

```
      20/290/U   40/S    80/O
```

This tells us line 20 has a reference to line 290 which is **U**ndefined, line 40 has a **S**yntax error (the double comma between 20 and 60), and line 80 has an **O**verflow line number (70000). After changing 290 to 90, 70000 to 10 and removing the extra comma, the command o50,1,50,69<ENTER> would generate the following 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
```

This program needs a little help to do anything meaningful. Let's renumber this program so that the INPUT statement in line 30 gets processed. This is accomplished by MOVING lines 30 to 70 inclusive to less than 20. Lets key in o11,1,30,70<ENTER>, and the results is:

```
      10 PRINT "TEST"
      11 INPUT A
      12 ON A GOTO 10,20,14,15
      13 GOTO 10
      14 PRINT A
      15 PRINT A*2
      20 GOTO 90
      80 GOTO 10
      90 END
```

As long as newly generated line numbers do not overlap the original text lines, the new lines may be placed anywhere in text. When renumber moves text, an 'M' is displayed to indicate a move is taking place. If the amount of the program to move is greater than the available memory, then renumber takes as many moves as necessary to complete the move - indicated by multiple 'M's.

P List a page of program lines

P[ln]<ENTER>

The **P** command lists lines, starting with the line equal to or higher than **ln**, until two screen lines are available. Subsequent **P**<ENTER>'s continue to list the balance of the program starting with the last line previously listed until the last line is listed. After the last line is listed, **P**<ENTER> clears the display and lists the last line.

MODEL I and MODEL III DISK BASIC KEY WORDS/MAX-80, MODEL 4 and ESOTERIC

MAX-80, MODEL 4, and ESOTERIC BASIC reassigned token values for key words that most likely would not be used in a program. e.g., DELETE. And MAX-80, MODEL 4, and ESOTERIC BASIC assigned key words to the four unassigned values of 0FBH through 0FFH; otherwise, the token values are the same for the five versions of BASIC.

DEC	HEX	WORD	DEC	HEX	WORD	DEC	HEX	WORD	DEC	HEX	WORD
128	80H	END	160	A0H	OUT	192	C0H	VARPTR	224	E0H	EXP
129	81H	FOR	161	A1H	ON	193	C1H	USR	225	E1H	COS
130	82H	RESET	162	A2H	OPEN	194	C2H	ERL	226	E2H	SIN
131	83H	SET	163	A3H	FIELD	195	C3H	ERR	227	E3H	TAN
132	84H	CLS	164	A4H	GET	196	C4H	STRING$	228	E4H	ATN
133	85H	CMD	165	A5H	PUT	197	C5H	INSTR	229	E5H	PEEK
134	86H	RANDOM	166	A6H	CLOSE	198	C6H	POINT	230	E6H	CVI
135	87H	NEXT	167	A7H	LOAD	199	C7H	TIME$	231	E7H	CVS
136	88H	DATA	168	A8H	MERGE	200	C8H	MEM	232	E8H	CVD
137	89H	INPUT	**169**	**A9H**	**NAME/CHAIN**	201	C9H	INKEY$	233	E9H	EOF
138	8AH	DIM	170	AAH	KILL	202	CAH	THEN	234	EAH	LOC
139	8BH	READ	171	ABH	LSET	203	CBH	NOT	235	EBH	LOF
140	**8CH**	**LET/SOUND**	172	ACH	RSET	204	CCH	STEP	236	ECH	MKI$
141	8DH	GOTO	173	ADH	SAVE	205	CDH	+	237	EDH	MKS$
142	8EH	RUN	**174**	**AEH**	**SYSTEM/CALL**	206	CEH	-	238	EEH	MKD$
143	8FH	IF	175	AFH	LPRINT	207	CFH	*	239	EFH	CINT
144	90H	RESTORE	176	B0H	DEF	208	D0H	/	240	F0H	CSNG
145	91H	GOSUB	177	B1H	POKE	209	D1H	[	241	F1H	CDBL
146	92H	RETURN	178	B2H	PRINT	210	D2H	AND	242	F2H	FIX
147	93H	REM	**179**	**B3H**	**CONT/EXIT**	211	D3H	OR	243	F3H	LEN
148	94H	STOP	180	B4H	LIST	212	D4H	>	244	F4H	STR$
149	95H	ELSE	181	B5H	LLIST	213	D5H	=	245	F5H	VAL
150	96H	TRON	**182**	**B6H**	**DELETE/ERASE**	214	D6H	<	246	F6H	ASC
151	97H	TROFF	**183**	**B7H**	**AUTO/ZERO**	215	D7H	SGN	247	F7H	CHR$
152	98H	DEFSTR	184	B8H	CLEAR	216	D8H	INT	248	F8H	LEFT$
153	99H	DEFINT	**185**	**B9H**	**CLOAD/WC, HR**	217	D9H	ABS	249	F9H	RIGHT$
154	9AH	DEFSNG	**186**	**BAH**	**CSAVE/LABEL**	218	DAH	FRE	250	FAH	MID$
155	9BH	DEFDBL	187	BBH	NEW	219	DBH	INP	251	FBH	'
156	9CH	LINE	188	BCH	TAB(	220	DCH	POS	**252**	**FCH**	**HEX$**
157	**9DH**	**EDIT/SORT**	189	BDH	TO	221	DDH	SQR	**253**	**FDH**	**WPEEK**
158	9EH	ERROR	190	BEH	FN	222	DEH	RND	**254**	**FEH**	**BIN$**
159	9FH	RESUME	191	BFH	USING	223	DFH	LOG	**255**	**FFH**	**ROW**

Arguments in a program cannot contain key words. Example: DIS**TAN**CE, D**ON**E, and B**LOC**K cannot be used as arguments because they contain TAN, ON, and LOC respectively.

SUPERBASIC

A SUPERBASIC program consists of one or more program lines. The lines may be numbered from 0 to 65529 for the Model I and Model III and 0 to 65535 for the MAX-80, Model 4, and ESOTERIC. The lines and are created with 1 to 240 characters for the Model I and Model III and 0 to 245 characters for the MAX-80, Model 4, and ESOTERIC. The characters in a line make one or more BASIC statements. A statement is one complete instruction. A statement can be one word and/or a group of words. When you are in the command mode, you can use most statements as commands – you cannot complete INPUT statements in the command mode. And, the commands previously described cannot be used as statements in a BASIC program. Some of the following SUPERBASIC instructions may be used as statements; however, their execution terminates the program and clears all variables. Examples of SUPERBASIC instructions are in the sample program found in APPENDIX B. This is indicated with *'See line 00000 in sample program.'* where 00000 is the line number in the sample program.

The statements that follow are either not covered in the MODEL III/4 BASIC Language Reference Manual (Radio Shack CAT. NO. 26-2112) and/or are different in SUPERBASIC.

&H, and &X Hexadecimal and Binary constants.

&[**H**]hhhh hhhh is a hexadecimal number 1 to 4 characters long.
&X xxxxx xxxxx is a binary number 1 to 16 characters long.

The ampersand instruction is an integer function that enables you to use hexadecimal (base 16) or binary (base 2) constants in your program. The H is optional in SUPERBASIC.

EXAMPLES:

```
X = &4000           assigns the decimal value of 16384 to X.
Y = &6000 - &5200   assigns 3584 (24576-20992) to the variable Y
POKE &FFFD, 4       places 4 in memory location FFFDH.
Z = &X1111          assigns 15 to the variable Z.
```

The ampersand functions can be used to assign values to variables; or, can be used as operands for numeric operators, relational operators, or logical operators.

See lines 10500 and 10600 in sample program.

MAX-80, MODEL 4, and ESOTERIC:
The VAL string function can process an ampersand constant in a string.

EXAMPLE:

```
X = VAL("&4000")    assigns the decimal value of 16384 to X
```

A useful application using this capability is converting an input statement that asks for a hexadecimal number into a constant that can be processed with other functions.

EXAMPLE: Get value contained in a word.

```
1250 INPUT "Location of word (HEX)";A$
1300 PRINT WPEEK(VAL("&" + A$)
```

MAX-80, MODEL 4, and ESOTERIC

ASC ASCII to decimal

ASC(str$)

str$ = string.

ASC returns the ASCII code of the first character in **str$**. The string argument must be enclosed in parentheses. If the string argument is null, ASC returns 0.

```
EXAMPLE:  A$="GEORGE", B$="HARRY", C$="01010", D$="" (null)
        15600 PRINT ASC(A$),ASC(B$),ASC(C$),ASC(D$),ASC("E")
              prints: 71, 72, 48, 0, and 69 respectively.
```

MAX-80, MODEL 4, and ESOTERIC

BIN$ Integer to binary.

BIN$(exp)

exp = expression in the range of -32768 to 32767 or 0 to 65535.

BIN$ converts a numeric expression or constant into a 16 character string. The numeric expression or constant must be enclosed in parentheses.

```
EXAMPLE:
        10 PRINT BIN$(173)    prints 0000000010101101
        20 PRINT BIN$(-74)    prints 1111111110110110
        30 PRINT BIN$(65461)  prints 1111111110110110
```

MAX-80, MODEL 4, and ESOTERIC

CALL Execute machine language routine.

CALL(exp)

exp = expression in the range of -32768 to 32767, or 0 to 65535.

CALL executes the machine language routine at **exp**. If the machine language routine executes without moving the Stack Pointer, and terminates with a RET instruction, your SUPERBASIC program continues with the next statement. CALL does not pass and/or receive any arguments to/from the machine language routine. Use the USR function to pass and/or receive one argument.

```
EXAMPLE:
        1020 CALL &4461:CLS
```

When this line is encountered, SUPERBASIC calls the machine language routine at 4461H (screen dump for MAX-80 and MODEL 4) then returns to the next statement, CLS.

See line 19200 in sample program.

MAX-80, MODEL 4, and ESOTERIC

CLEAR n Change the amount of string space. {does not change program}

CLEAR exp

exp = expression in the range of -32768 to 32767 or 0 to 65535.

CLEAR exp changes the amount of string space currently allocated for string storage without clearing variable values. However, all FOR-NEXT loops and GOSUBs are nullified.

EXAMPLE:

```
100 CL=0
120 ON ERROR GOTO "ERROR"
130 LABEL "CLEAR"
140 CLEAR CL
150 INPUT"Your State";ST$
... more program lines
780 LABEL"ERROR":IF ERR = 26 THEN CL=CL+5: RESUME "CLEAR"
```

This short program keeps asking for the name of your State until it clears enough string space (by adding 5 to the amount of free string space) to handle the number of characters in you state's name.

See line 10200 in sample program.

MAX-80, MODEL 4, and ESOTERIC

CLEAR Clears variable values, resets USR definitions, collapses FOR-NEXT loops, resets GOSUB return pointers, and changes program.

CLEAR n:CLEAR is a near equivalent of **CLEAR n** on the MODEL I and MODEL III.

MAX-80, MODEL 4, and ESOTERIC

CLS n Paint the screen.

CLS(exp)

exp = an expression between 0 and 255.

CLS exp places the video poke value of **exp** in all video RAM positions.

EXAMPLE:

```
10 CLS 191
```

This whites out the display and does not reset WIDE display [produced by CHR$(23)]. CLS without and expression clears the screen, resets reverse video, resets WIDE mode, and homes the cursor. CLS32 clears the screen and does not reset WIDE display.

NOTE: The <CLEAR> key does not clear the screen. <CLEAR> clears from the cursor to the end of display.

MAX-80, MODEL 4, and ESOTERIC

CMD"B" Disable normal responses to the <BREAK> key.

```
55 CMD"B":INPUT"Amount";AM#
```

CMD"B" disables the <BREAK> key until a statement or command that change a program is executed. The <BREAK> key is temporarily enabled during a CMD"fffff" and disabled again when SUPERBASIC is resumed. See the **C** command for statements and commands that change a program.

CMD"C" Space compression. {changes program}

CMD"C"<ENTER>

This command eliminates unnecessary spaces and linefeeds from the resident BASIC program, resulting in more available memory and faster execution.

CMD"D" Load and execute DEBUG.

CMD"D"

This command loads and executes DEBUG. (See the LIBRARY command DEBUG.)

CMD"E" Disk I/O error.

CMD"E"

This command displays the last DOS error that occurred during this invocation of BASIC.

MODEL I and MODEL III

CMD"K" Zero array.

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

vv = any valid array variable name.
0 = dummy argument (one 0 required for each dimension).

CMD"K" sets all string arrays to null and sets every element of a numeric array to zero. ZERO does not remove the array from RAM. Use **CMD"L"** to remove an array.

EXAMPLE:

```
10 DIM A(3,4), B$(12)
. . .
500 CMD"K", A(0,0),B$(0) {all elements (20) of array A are now zero, and all 13
                          elements of array B$ are now null}
```

MODEL I and MODEL III

CMD"L" Remove array from memory

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

vv = any valid array variable name.
0 = dummy argument (one 0 required for each dimension).

EXAMPLE:

```
10 DIM A(3,4), B(12)
. . .
500 CMD"L", A(0,0),B(0) {array A and array B$ are removed from memory}
510 DIM A(5,6),B(7)     {removing an array enables you to re-dimension arrays}
```

MODEL I and MODEL III

CMD"O" Multi-key numeric and string sort.

CMD"O", exp,arv1(se)[POS bt][,[>]arv2(se)][POS bt][,...] {DIRECT}
CMD"O", exp,*iarv(se),arv1(se)[POS bt][,[>]arv2(se)][POS bt][,...] {INDIRECT}

See SORT for a complete description of CMD"O".

CMD"O" requires installation of UTILA/BOL. The BASIC program OPT/BAS enables you to switch between UTILA/BOL and UTIL/BOL. If you switch in UTILA/BOL, you will not have the following commands:

F single letter command	M single letter command
N single letter command	CMD"C"
CMD"V"	CMD"X"

You can always switch back UTIL/BOL. Or, have two different option diskettes.

CMD"P" PACKER - Pack program lines. {changes program}

CMD"P"<ENTER>

This command packs BASIC program lines together. When CMD"P" is entered:

'Maximum line length _'

appears. Enter the maximum length for program lines. A **#** following the maximum length instructs PACKER to pack program lines without removing spaces. MAX-80, MODEL 4, and ESOTERIC: A ***** following the maximum length instructs PACKER not to renumber 0, 1, 2, etc. Both **#** and ***** may be specified in any order. Enter any value between 0 and 65535. A 0 packs lines to a maximum of 65536 characters. The default value is 240 for the MODEL I and MODEL III and 245 for the MAX-80, MODEL 4, and ESOTERIC. If no errors occur, the program is packed and for the MAX-80, MODEL 4, and ESOTERIC renumbered 0, 1, 2, etc. Lines referenced by other program lines are not packed to a previous line. Lines with a REM or an IF statement are not packed to the following line. PACKER does not translate IF… THEN… expressions to IF… THEN… ELSE expressions; therefore, properly maintaining present program logic.

CMD"Q" String sort.

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

vv$ = Any valid string array variable name
n1 = An integer or integer variable representing the absolute highest element to be sorted.
n2 = A positive integer or integer variable representing the column number to sort in a two dimensional array (relative zero).

This command sorts a string array in accordance with the sign of n1. If n1 is positive, then the sort is in ascending and/or alphabetical order. If n1 is negative, then the sort is in descending and/or reverse alphabetical order. Option (1) is used to sort a single dimension array. The array is sorted up to the $n1^{th}$ element, including the 0^{th} element. Option (2) is used to sort a two dimensional array with n2 indicating which column of the array to use as the sort key.

EXAMPLE:

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

The printout is:

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

NOTE the following key points.

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" does not cause an error if the value of n1 is greater than the first dimension of the array.
3. If line 60 were changed to: CMD"Q",-I,A$(0) the printout would be:

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

What happened to WYOMING? A$(0)="WYOMING" and A$(10)="".

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

```
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.

There is another sort statement available. See SORT for MAX-80, MODEL 4, and ESOTERIC; and see CMD"O" for MODEL I and MODEL III.

MODEL I
CMD"R" Enable interrupts

CMD"R"

This command enables the interrupts.

CMD"S" Exit SUPERBASIC.

CMD["S"]<ENTER>

This command exits the present SUPERBASIC invocation. The S is optional.

MODEL I
CMD"T" Disable interrupts

CMD"T"

This command disables the interrupts. This command is invoked automatically when CLOAD, CSAVE, and/or SYSTEM in entered in the command mode.

CMD"U" UNPACKER - Unpack program lines. {changes program, and CLEARs 73}

CMD"U"[,C]<ENTER>

The UNPACKER unpacks a program into as many lines as possible, inserts spaces around each key word, and renumbers the program 10, 20, etc. The **C** option instructs CMD"U" to make as many lines as possible without adding spaces around each key word. UNPACKER displays single letters to let you know the progress:

C = Checking for errors (see renumber, O command, for error types)
S = Spreading (separating multiple statement lines to individual lines)
U = Unpacking (adding spaces around key words)
R = Renumbering

CMD"V" Defined variables.

CMD"V"<ENTER>

This command displays all assigned scalar variables and string equivalents in the order they were created, and all array arguments with the left most index incrementing to the dimension limit, then the elements to the right incrementing to the dimension limit, etc. The screen clears and up to one page, less two lines are displayed. Press any key to display an additional variable or <ENTER> to display up to one page more.

MODEL I and MODEL III

CMD"W" Transfer BASIC program to Level II BASIC or Model III BASIC

CMD"W"<ENTER>

This command transfers the resident BASIC program to ROM BASIC. cf **BASIC #** on page 102.

CMD"X" Remove REMark statements. {changes program}

CMD"X"<ENTER>

This command removes REMark statements from the resident BASIC program. If a line consists of a remark statement only, then the entire line is removed. After the remark statements are removed, error checking is performed by RENUM/BOL to check for undefined lines. CMD"X" requires UTIL/BOL, EDIT/BOL, and RENUM/BOL to be available. If a program consists of only REMark statements, then the last REMark statement is not removed.

MAX-80, MODEL 4, and ESOTERIC

CMD"Y" Utility for counting, and listing.

CMD"Y",[T]<ENTER>	Prints the line lengths in ascending line number order in the format **nnnnn/llllll**, where **nnnnn** is the line number and **llllll** is the length. If **T** is specified, then the lengths reflect the tokenized length.
CMD"Y",C<ENTER>	Prints the number of lines in the program.
CMD"Y",L[G]<ENTER>	LISTs the program, indenting lines wider than the video. If **G** is specified, then do not convert graphic characters to periods.
CMD"Y",LP[G]<ENTER>	LLISTs the program, indenting lines wider than the FORMS setting for width. If **G** is specified, then do not convert graphic characters to periods.
CMD"Y",Q<ENTER>	Prints the highest 64 line lengths in ascending length order in the format **nnnnn/llllll**, where **nnnnn** is the line number and **llllll** is the length.
CMD"Y",A<ENTER>	Prints the line numbers that should not be saved using the ",A" option (ASCII) in the CMD"Y",Q format. May not print anything.

CMD"fffff" Execute a MULTIDOS LIBRARY command or UTILITY from SUPERBASIC.

CMD var$
or **CMD "fffff"**

var$ = string argument valued with a MULTIDOS LIBRARY command or UTILITY
fffff = any valid MULTIDOS LIBRARY command or UTILITY.

This command executes any MULTIDOS command that does not change TOPMEM[5], including BASIC, from within the SUPERBASIC environment. The command may be used in the direct mode or as a statement within a BASIC program. CMD"fffff" temporarily sets TOPMEM to a

value to protect the BASIC program, and requires 6278 free bytes to execute for MODEL I and MODEL III or 6048 free bytes to execute for MAX-80, MODEL 4, and ESOTERIC.

EXAMPLES:

```
CMD"DIR"
The directory contents is displayed.

CMD"BACKUP":GOTO 230
BACKUP/CMD is loaded and executed. Upon completion, the BASIC program executes
the next instruction - GOTO 230.
```

NOTE 5: The LIBRARY commands DO, RESET, TOPMEM, and ROUTE change the value of TOPMEM. A DO file completes before it returns to BASIC, RESET simply exits BASIC, TOPMEM changes are ignored, and a ROUTE to a filespec is not allowed. You may execute a command such as CMD"ROUTE PR to DO" with no problem.

Any non-MULTIDOS /CMD program that uses memory above 68FFH (60FFH for ESOTERIC), causes a crash upon a return to SUPERBASIC if just the minimum (6278 or 6048) bytes are free. You can compute the free memory required:

FREE MEMORY REQUIRED = HIGHEST BYTE USED - 26879 + 6048. {MODEL 4 MULTIDOS}

See line 10600 in sample program.

DEF FN Define function.

DEF Fnxxx(uuu[,uuu...]) = www

xxx = Name of the function (any valid variable name).
uuu = Arguments to be used to define the function.
www = An expression or formula involving the arguments uuu.

This statement creates an implicit function. After a function has been defined, the function can be used as any of the other intrinsic functions, e.g., ATN, COS, ASC, etc. The type of value returned is the same as the type of variable used to name the function. In addition, the arguments used in the DEF FN statement are local arguments and do not affect the arguments used elsewhere in the program. A space is optional between DEF and FN, i.e., DEF FN or DEFFN.

EXAMPLE:

```
10 DEFFN Q(K,L) = K/10 + L/20
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, and the arguments in line 20 and 30 are N and D. K and L can be used elsewhere in the program without affecting FN Q. And FN Q, using K and L, has no effect on the arguments K and L used elsewhere in the program.

DEF USR Define entry address of USR routine.

DEF USRn = 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, arguments, and/or functions.

EXAMPLE:

```
10 CLEAR 500: DEFINT A-Z
20 N = 20000
30 DEF USR 5 = N * 3
.. more program lines
780 G=USR5(E)
```

Defines 60000 decimal to as the entry point to a USR 5 call. A space is optional between DEF and USR, i.e., DEF USR or DEFUSR.

See line 17500 in sample program.

MAX-80, MODEL 4, and ESOTERIC

ERASE Remove an array from memory.

ERASEvv[,vv[,...]]

vv = any valid array variable name.

ERASE removes arrays. The memory used by the array is made available without clearing any other variable. In addition, this enables you to dimension the same array again.

EXAMPLE:

```
10 DIM A$(20,2)
... more program lines
1020 ERASE A$
1030 DIM A$(30,3), B(22), C(88)
... more program lines
1020 ERASE B,C
```

The MODEL I and MODEL III functional equivalent to ERASE is CMD"L",

See line 22000 in sample program.

ERROR Simulate an error condition.

ERROR exp

exp = expression between 1 and 255

ERROR simulates the error in exp. The MODEL I and MODEL III (ROM) cannot simulate errors above 23. The ERR codes, ERROR codes, and definitions are detailed on pages 154 through 159.

MAX-80, MODEL 4, and ESOTERIC

EXIT Prematurely exit a FOR-NEXT loop.

EXIT eeeee

eeeee = a line number from 0 to 65535 or a "label"

EXIT enables you to exit a FOR-NEXT loops before the loop has completed the specified number of cycles. EXIT transfers the program control to the line number and/or label and cancels the current loop and all immediate outer FOR-NEXT loops. The value of the loop counters when EXIT is encountered is available for use.

EXAMPLE:

```
10 FOR I = 1 TO 999: IF INKEY$<>"" EXIT "KEY"
20 NEXT I
30 PRINT "Sorry, timed out."
40 GOTO "SLEEP"
50 LABEL "KEY"
```

This example provides a means to time a response. If the response takes longer than the allocated time (approximately 3 seconds), then the program branches to the line with LABEL"SLEEP". However, if you press a key, then the program branches to the line with LABEL"KEY" and the time of the response is in the variable I.

See lines 20100 and 40200 in sample program.

MAX-80, MODEL 4, and ESOTERIC

HEX$ Integer to hexadecimal.

HEX$(exp)

exp = expression in the range of -32768 to 32767 or 0 to 65535.

HEX$ converts a numeric expression or constant into a four character hexadecimal string. The numeric expression must be enclosed in parentheses.

EXAMPLE:

```
10 PRINT HEX$(173)    prints "00AD"
20 PRINT HEX$(-74)    prints "FFB6"
30 PRINT HEX$(65461)  prints "FFB6"
```

INPUT Assign data to variable(s).

(1) **INPUT**["prompt";]var1[,var2][,...
MAX-80, MODEL 4, and ESOTERIC
(2) **INPUT**!col,row,[%rvar,]|[#]len,char,[USING str$],["prompt";]var1[,var2][,...
(3) **INPUT**@pos,[%rvar,]|[#]len,char,[USING str$],["prompt";]var1[,var2][,...

"prompt"; = An optional prompt message.
var1, var2, etc. = numeric and/or string arguments.
col = the video column (X coordinate) horizontal axis. (0 to 79 in 80X24 video mode, or 0 to 63 in 64X16 video mode)
row = the video row (Y coordinate) vertical axis. (0 to 23 in 80X24 video mode, or 0 to 15 in 64X16 video mode)
% = precedes rvar to hold the terminator value
rvar = the variable to receive the terminator value (use of %rvar is exclusive with the automatic <ENTER> flag [#] because it also specifies automatic <ENTER> when the input field is full. A left arrow <←> in position one exits with **rvar** equal to 8. An <ENTER> before the field is full returns 13 into **rvar**. A full input field returns 0 in **rvar**.
# = specifies automatic <ENTER> when the input field is full.
len = the length of the input field(s) (1 to 255).
char = video poke value to delineate the input field (1 to 255).
USING str$= str$ = string expression representing the mask characters. **USING str$** means only accept characters in **str$**.
pos = video display position. (0 to 1919 in 80X24 video mode, or 0 to 1023 in 64X16 video mode)

Option 1 displays a space and question mark before the input field and prints a carriage return and places the cursor on the next line when <ENTER> is pressed. Options 2 and 3 do not print a space and question mark and do not print a carriage return when <ENTER> is pressed. If you want the cursor on the next line, you will have to issue a PRINT statement.

EXAMPLE:

```
10 CLEAR 5000:CLS
20 A$="1234"
... more program lines
300 PRINT"Please select one category"
310 PRINT"(1) List the file."
320 PRINT"(2) Save the file."
330 PRINT"(3) Display directory."
340 PRINT"(4) Return to DOS."
350 INPUT @ 720, 1, 95, USING A$, "Selection ";S%
... program continues
```

The only characters the INPUT statement accepts in line 350 are 1, 2, 3, or 4: the characters in A$ (line 20).

Options 2 and 3 enable you to have an INPUT prompt on the last line of the video display without scrolling the display when you enter the data.

See lines 12300 and 18000 in sample program.

INSTR String search.

INSTR([p,]string,substring)

p = An optional search starting position in string (default 1).
string = The name of the string to be searched.
substring = (1) The name of the substring you are searching for,
or (2) The actual substring you are searching for.

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$="SUPERBASIC", W$="" {null}, X$="SUPER")

Expression	Result
INSTR (Z$, "PER")	3
INSTR (2, Z$, W$)	0 {W$ is null}
INSTR (3, Z$, "U")	0
INSTR (2, Z$, X$)	0
INSTR (4, Z$, "BASIC")	6
INSTR (3, "ABCDABCDABCDABCD", "ABC")	5

MAX-80, MODEL 4, and ESOTERIC

LABEL Identifier for a symbolic expression for a line.

LABEL"label"

label = a string (1 to 255 characters)

LABEL enables you to construct a SUPERBASIC program with labels to identify key points in your program without having to remember the line number of the routine. For example, you have developed a routine to calculate the difference between two dates, and have placed this routine somewhere near the end of the program. While you are still developing the front end, as you add lines you must continually renumber the program to fit the new lines in the appropriate positions in the program text. After each renumber, you must find the line number of this routine and refer to it correctly. However, if you labeled the routine "DIFFDAYS", you can renumber as many times as you wish and simply refer to "DIFFDAYS" to perform the required calculation.

EXAMPLE:

```
10 CLEAR5000:CLS
20 INPUT!0,0,6,95,USING"0123456789","The promise date in YYMMDD ";PD!
30 INPUT!0,1,6,95,USING"0123456789","The critical date in YYMMDD ";CD!
40 GOSUB"DIFFDAYS":IF DD!>0 THEN PRINT!0,2,"The promise supports the program":
GOTO"NEXT" ELSE PRINT!0,2,"You must improve the promise date":GOTO"EXPEDITE"
... more program lines
1030 LABEL"DIFFDAYS" '(PD! - CD!) in days, is returned in DD!
... date calculator routine
2090 RETURN
```

See lines 11800, 12600, 15400, 16400, … in sample program.

LINE INPUT Input a string from keyboard.

(1) **LINE INPUT**["prompt";]var$
MAX-80, MODEL 4, and ESOTERIC
(2) **LINE INPUT**!col,row,[%rvar,]|[#]len,char,[[NOT]USING str$],["prompt";]var$
(3) **LINE INPUT**@pos,[%rvar,]|[#]len,char,[[NOT]USING str$],["prompt";]var$

Note: The vertical bar, |, denotes exclusive options. i.e., you may use the option before the vertical or the option after the vertical bar but not both in the same statement.

"prompt"; = An optional prompt message.
var$ = a string variable.
col = the video column (X coordinate) horizontal axis. (0 to 79 in 80X24 video mode, or 0 to 63 in 64X16 video mode)
row = the video row (Y coordinate) vertical axis. (0 to 23 in 80X24 video mode, or 0 to 15 in 64X16 video mode)
% = precedes rvar to hold the terminator value
rvar = the variable to receive the terminator value (use of %rvar is exclusive with the automatic <ENTER> flag [#] because it also specifies automatic <ENTER> when the input field is full. A left arrow <←> in position one exits with **rvar** equal to 8. An <ENTER> before the field is full returns 13 into **rvar**. A full input field returns 0 in **rvar**.
# = specifies automatic <ENTER> when the input field is full.
len = the length of the input field (1 to 255).
char = video poke value to delineate the input field (1 to 255).
USING str$= str$ = string expression representing the mask characters. **USING str$** means only accept characters in **str$**.
pos = video display position. (0 to 1919 in 80X24 video mode, or 0 to 1023 in 64X16 video mode)

Option 1 issues a linefeed and carriage return and places the cursor on the next line when <ENTER> is pressed. Options 2 and 3 do not issue a linefeed and carriage return when <ENTER> is pressed. If you want the cursor on the next line, you will have to issue a PRINT statement. The input field begins immediately after the optional prompt; therefore, for most programs your prompt would have a trailing space. e.g., "Amount ";

Options 2 and 3 enable you to have a LINE INPUT prompt on the last line of the video display without scrolling the display when you enter the data.

This statement is the method to input a complete line from the keyboard, including leading spaces, punctuation, and linefeeds. This statement sets the variable to null, and does not print a question mark. A space is optional between LINE and INPUT, i.e., LINE INPUT or LINEINPUT.

See lines 15800, 16600, 18500, 26100, and 26500 in sample program.

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 statement changes part of a string. The length of the target string, **vv$**, is not changed by the MID$ = statement. Excess characters to the right of **rr$** are ignored if **rr$** is longer than **vv$**.

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

Expression	Resultant C$
MID$(C$,3,4)="ABCDE"	12ABCD78
MID$(C$,1,2)=D$	BA345678
MID$(C$,8)="YZ"	1234567Y

See lines 19100 and 19300 in sample program.

MAX-80, MODEL 4, and ESOTERIC

NEW Resets program pointers.

NEW

NEW resets all program pointers and CLEARs 50. You can un-NEW with o<ENTER>.

MAX-80, MODEL 4, and ESOTERIC

ON STOP GOTO Trap a <BREAK> press during program execution.

ON STOP GOTO eeeee

eeeee = a line number from 0 to 65535 or a "label"

ON STOP GOTO transfers to the specified line or label if the <BREAK> key is pressed during program execution. After an ON STOP GOTO **eeeee** is encountered in the program, the <BREAK> key is disabled during input: the <BREAK> key is not recognized in a LINE INPUT and/or an INPUT statement. During program execution a transfer to **eeeee** occurs if the <BREAK> key is pressed.

To enable <BREAK>, set **eeeee** to zero. i.e., ON STOP GOTO 0. This should be performed in a common exit routine to ensure the <BREAK> key is available for the next program.

See lines 12200, 17600, 19400, 21300, and 40400 in sample program.

PRINT Display data.

(1) **PRINT**outv
(2) **PRINT**@pos,outv
MAX-80, MODEL 4, and ESOTERIC
(3) **PRINT**!col,row,outv

outv = expression with optional USING, TAB(, or TAB(# modifiers.
@pos = video display position. (0 to 1919 in 80X24, or 0 to 1023 in 64X16)
col = the video column (X coordinate) horizontal axis. (0 to 79 in 80X24 video mode, or 0 to 63 in 64X16 video mode)
row = the video row (Y coordinate) vertical axis. (0 to 23 in 80X24 video mode, or 0 to 15 in 64X16 video mode)

See lines 14700, 15100, 15700, 16500, 24700, 25100, and 25400 in sample program.

MAX-80, MODEL 4, and ESOTERIC
RANDOM Seed random number generator.

RANDOM[RESET]

RANDOM enables you to have a different sequence of pseudorandom numbers. RANDOM RESET enables you to have the same sequence of pseudorandom numbers.

MAX-80, MODEL 4, and ESOTERIC
RESTORE Reset DATA pointer.

RESTORE[eeeee]

eeeee = a line number from 0 to 65535 or a "label"

RESTORE resets the DATA pointer to the beginning of the BASIC program, or to the line or label represented by eeeee.

MAX-80, MODEL 4, and ESOTERIC
RND Produce pseudorandom numbers.

RND(exp)

exp = expression in the range of -16777215 and 16777215

RND uses the integer part of **exp** and:
returns a pseudorandom number between 1 and INT(**exp**) if INT(**exp**) is greater than 1.
returns 1 if INT(**exp**) = 1.
returns a pseudorandom number between 0 and less than 1 if INT(**exp**) = 0.
returns -1 if INT(**exp**) = -1.
returns a pseudorandom number between INT(**exp**) and -1 if INT(**exp**) is less than -1.

MAX-80, MODEL 4, and ESOTERIC (MODEL I and MODEL III exact equivalent is **CMD"O"**)

SORT Multi-key numeric and string sort.

```
        SORT exp,arv1(se)[POS bt][,[>]arv2(se)][POS bt][,...]     {DIRECT}
        SORT exp,*iarv(se),arv1(se)[POS bt][,[>]arv2(se)][POS bt][,...] {INDIRECT}

           exp =  absolute number of elements in each of the arrays participating in
                  the sort (0 to 32767).
    arv1, etc. =  the arrays to participate in the sort.
            se =  starting element of the corresponding array.
            bt =  starting point for string array (bt = 0 to 255).
          iarv =  an integer array used as a pointer for indirect sorts.
```

Both **exp** and **arv1** are required for either sort. In addition, **iarv** is required for the indirect sort.

The SORT statement is used to sort integer, single, or double precision numeric arrays, and string arrays. If **exp** is positive, then the sorted arrays is in ascending alphabetical order for string arrays or most negative to most positive for numeric arrays. Each array can have a plus (+) or minus (-) sign preceding the array name to determine the order in which that particular array controls the sort. However, if **exp** is evaluated to be negative, then the signs preceding the arrays are ignored, and the sort is in reverse alphabetical order (descending) for string arrays or most positive to most negative for numeric arrays.

Direct Sort

The DIRECT sort statement swaps all of the corresponding elements in each array in the sort statement.

Although all corresponding elements in the sort statement are swapped, you can specify how many arrays are to be considered in determining the sort by placing a > character in front of an array. The > character directs the sort statement not to consider arrays that follow.

EXAMPLE:

```
        SORT E, J(1), K(1), -L(1), >M(1), N(1)
```

If E is evaluated positive, then the elements in array J are sorted in ascending order. If two elements, being compared, in array J are equal, then the corresponding two elements are compared in array K to determine the sort order. If the corresponding two elements in array K are equal, then the corresponding two elements in array L are compared. If the corresponding two elements in array L do not match, then the sort order is descending for all of these corresponding elements when array L is used to determine the sort. If the corresponding two elements in array L are equal, arrays M and N are not considered in determining the sort; therefore, the corresponding elements are not swapped. Although arrays M and N do not determine the sort order, the corresponding elements in arrays M and N are swapped along with the corresponding elements in arrays J, K, and L.

If E is evaluated negative, then all of the arrays are sorted in descending order.

The use of the sign in evaluating the E expression enables you to programmatically control the sort order without having two sort statements.

```
770 'The variable K holds the number of elements to be sorted
780 INPUT!0,ROW(0),#1,95,USING"ADad","<A>scending or <D>escending ";Z$
790 PRINT
800 IF Z$ = "a" OR Z$ = "A" THEN E = K ELSE E = -K
```

You can place a negative sign in front of the E expression instead of each array to make the sort order descending.

```
SORT -22, A$(1), B#(1), A%(1), B!(1)
```

This statement sorts elements 1 through 22 of arrays A$, B#, A%, and B! in descending order. This is the same as:

```
SORT 22, -A$(1), -B#(1), -A%(1), -B!(1)
```

Optional Plus Sign

Although the sort defaults to ascending order, a plus sign is optional to signal ascending order.

```
SORT (J/2+K), +A(1),-B(1),+C(1)
```

This statement sorts array A in ascending order. If two elements, being compared, in array A are equal, then the corresponding two elements in array B are used to determine the sort order. If the corresponding two elements in array B are not equal, then the sort is descending for these two elements. This is the same as:

```
SORT (J/2+K), A(1),-B(1),C(1)
```

When a string array is used in a SORT statement, you can specify the starting point for comparison in each string array.

```
SORT V, A$(0) POS J, B$(0), C!(0),-D$(0) POS K, E#(0)
```

This statement sorts array A$ using the J^{th} character as the first character in determining the sort. If the length of a string in array A$ is less than J, then the elements are not swapped during the sort. Only the elements whose string's length is J or more characters long participate in the sort.

Indirect Sort

The indirect sort only swaps the elements in array **iarv**. The other arrays in the sort statement are not changed. The indirect sort enables you to sort pointers for the other arrays. The following example demonstrates the result of an indirect sort.

EXAMPLE:

GIVEN	
B(1) = 20	C(1) = 9
B(2) = 18	C(2) = 10
B(3) = 20	C(3) = 10
B(4) = 15	C(4) = 16
B(5) = 7	C(5) = 17
B(6) = 2	C(6) = 8

The initial values in array A% are not relevant for the indirect sort statement. Any prior values for the elements in A% are ignored and changed.

```
SORT 6, *A%(1), B(1), C(1)
```

The contents of the B and C arrays are not changed. However the contents of array A% is as follows:

```
A%(1) = 6
A%(2) = 5
A%(3) = 4
A%(4) = 2
A%(5) = 1
A%(6) = 3
```

A sorted list of arrays B and C can be obtained by using array A% as an index to the B and C arrays, i.e.,

```
240 FOR I = 1 TO 6
250 PRINT B(A%(I)), C(A%(I))
260 NEXT I
```

The indexing array (A% in the preceding example) must be an integer array. However, the other arrays in the indirect sort statement can be integer, single precision, double precision, or string.

```
SORT SQR(M!), *D%(1), G#(1), B$(1) POS 2, - H%(1), > J#(2)
```

The global descending control for indirect sorts can be either for the elements to participate in the sort or the index array.

```
SORT -12, *N%(1), H$(1)
SORT 12, *-N%(1), H$(1)
```

These two sort statements are equivalent in determining the sort order.

See line 20600 in sample program.

MODEL 4, and ESOTERIC

SOUND Generate sounds.

SOUND texp,dexp

texp = tone, expression in the range of 0 to 31.
dexp = duration, expression in the range of 0 to 255.

Texp is modulo 32. If texp evaluates to 56, then 24 is used for the tone. The lowest tone is zero. The lowest duration is zero and the longest duration is 255 which is about 11 seconds for a CPU speed of 4.05504 MHz. SOUND can't be stopped with <BREAK>.

MAX-80, MODEL 4, and ESOTERIC

TAB(# Indirect TAB.

TAB(#exp)

exp = expression in the range of 0 to 255.

TAB(#exp) uses the current cursor position as the reference point. Whereas a TAB(exp) statement uses the starting left margin as the reference point.

TROFF Disable TRON.

See TRON which follows.

MAX-80, MODEL 4, and ESOTERIC

TRON Enable trace.

There are nine trace commands to provide debugging aids in developing programs. TROFF and all eight TRON commands can be used as statements.

TRON or TRON 0 = Trace program flow in upper right corner of display.
TRON 1 = Trace program flow to printer.
TRON 2 = Print each BASIC statement in lower left corner of display prior to execution.
TRON 3 = Single step each BASIC statement or program line with delay.
TRON 4 = Full line single step.
TRON 5 = Single statement step.
TRON 6 = Disable single step and BASIC statement printing.
TRON 7 = Print erroneous statement when an error occurs.

When a BASIC program is executing, and TRON is enabled, the following control keys can be used to invoke other TRON commands:

<CTRL·P> = TRON 0	<CTRL·Q> = TRON 1
<CTRL·R> = TRON 2	<CTRL·S> = TRON 3
<CTRL·T> = TRON 4	<CTRL·U> = TRON 5
<CTRL·V> = TRON 6	<CTRL·W> = TRON 7

TRON Trace on: to VIDEO.

TRON enables printing each line number, prior to execution, in a controlled trace area - the first four video lines in the upper right corner. The first line number entered, after TRON, prints on the first video line in the upper right corner of the display prefixed by the # sign. As each new line is entered the new line number prints below the previous line number until four video lines are used. When the fifth line number is entered, the new line number overprints the first line number. When the sixth line number is entered the sixth line number overprints the second line number. This process continues, displaying the last four line numbers entered with the # sign prefixing the last entered line number.

TRON 1 Trace on: to PRINTER.

TRON 1 enables printing each line number, prior to execution, onto the printer. The printout is similar to:

```
LPRINT USING "######";LINE;
```

where LINE is the line number. This format, using six print positions per line number, enables you to control the printout by adjusting the FORMS width prior to using TRON 1.

TRON 2 Trace on VIDEO with print of BASIC statements.

TRON 2 prints each BASIC statement in the lower left corner using the last two video lines. The first statement in a program line is displayed without a preceding colon. A multiple statement line has the second and/or greater statement printed prefixed by a colon. A BASIC statement larger than two video lines scrolls the display.

TRON 3 Trace on VIDEO with delay.

TRON 3 delays the BASIC interpreter prior to each BASIC statement if TRON 5 is active; otherwise, TRON 3 delays prior to the execution of each BASIC line, displaying the line number in the upper right corner prior to the line being executed. You can switch between delays at the beginning of each line to delays at the beginning of each statement by pressing <CTRL·U> to delay at the beginning of each statement or <CTRL·T> to delay at the beginning of each line. The amount of delay is controlled by <SHIFT·↓> and <SHIFT·↑> while the program is executing. <SHIFT·↓> doubles the delay until the delay is approximately one second. And <SHIFT·↑> halves the delay until the delay is approximately three milliseconds. Both estimates are based on a CPU speed of 4.05504 MHz.

TRON 4 Trace on with full line pauses.

TRON 4 pauses execution of the BASIC program at the beginning of each line entered until any key is pressed except <SHIFT·@>.

TRON 5 Trace on with statement pauses.

TRON 5 pauses execution of the BASIC program at the beginning of each statement until any key is pressed except <SHIFT·@>.

TRON 6 Single step trace off.
TRON 6 disables TRON 2, TRON 3, and TRON 4 and TRON 5. TRON 6 executes a TROFF if TRON 2, TRON 3, TRON 4, or TRON 5 are the only trace commands active when TRON 6 is executed.

TRON 7 Print erroneous statement.
TRON 7 prints the erroneous BASIC statement when an error occurs. This command assists you in finding the error in a multiple statement line.

TRON 8 through TRON 65535 does not error because the TRON n invoked is the remainder of the 8 through 65535 divided by 8 (modulo 8).

TRON or TRON 0 and TRON 1 are mutually exclusive. TRON 4 and TRON 5 are obviously mutually exclusive. You can have TRON or TRON 1, TRON 2, TRON 3, TRON 4 or TRON 5, and TRON 7 active at the same time.

STRING COLLECTION

String collection for the MAX-80, MODEL 4, and ESOTERIC has been enhanced to significantly reduce the time it takes to collect active strings. To achieve this fast collection, two bytes of free memory are required for each active string in a temporary work area. When the string collection is complete, this temporary work area is returned to free memory.

Timing in seconds for a CPU speed of 4.05504 MHz (Yes, MODEL III at 4.05504 MHZ):

Strings	Fast Time	Slow Time	MODEL III Time
100	0.13	0.57	0.95
500	0.53	11.10	19.9
1000	1.11	43.20	77.8
2000	2.38	170.55	307.6

The previous method used to reduce string collection was obtained by clearing as much string space as possible. Now you should leave, if possible, at least to bytes free for each active string. Execution of a LOAD "filespec", RUN "filespec", and/or NEW command CLEARs 50 in addition to performing the LOAD, RUN, or NEW.

When any of the trace functions are active, BASIC prints a **$** in the upper right of the screen during the string collection routine. If there is insufficient RAM available for the fast collection, a **G** is printed and the slow process of string collection is invoked.

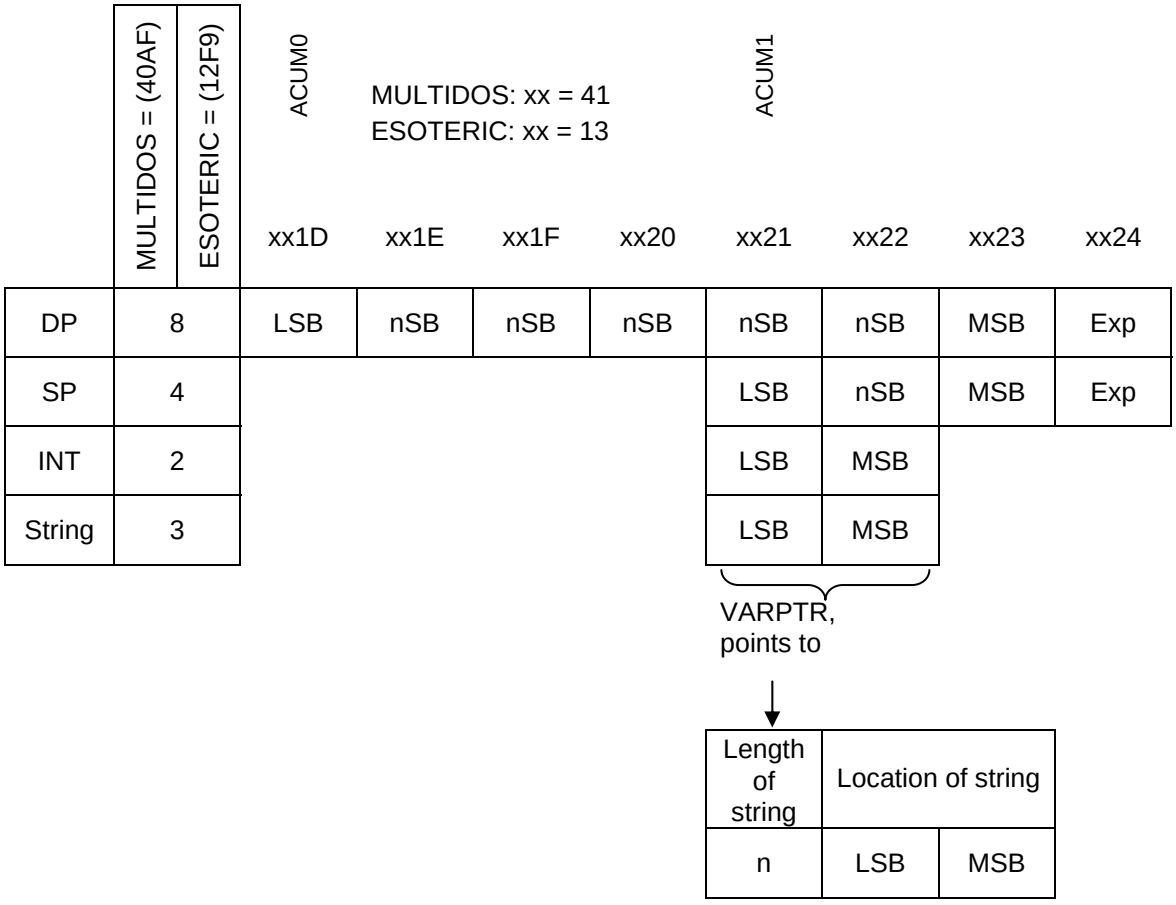
USR Execute a machine code routine.

```
var = USR [n](exp)

  var = variable to receive the value returned from the USR call.
    n = the digits 0 - 9. If n is omitted 0 is used.
  exp = any expression, any precision.
```

This function transfers control to a machine language subroutine that has been previously defined by the DEFUSR statement and returns to next statement following the USR call. MAX-80, MODEL 4, and ESOTERIC: All previous definitions are reset by a CLEAR statement.

Upon entry to the USR routine, the HL register pair is valued with the start of the USR routine and information about **exp** is in ACUM0 and/or ACUM1 with the type in (40AFH).



To pass **exp** to the HL register pair, execute a CALL MKINT as the first instruction in the USR routine. **exp** can be any precision and must be between -32768 and 32767.

If **exp** is a string: For the MODEL I and MODEL III, its VARPTR is in the DE register pair. For the MAX-80, MODEL 4, and ESOTERIC, its VARPTR can be placed into the HL register pair by executing CALL MKSTR as the first instruction.

The value placed in **var** upon return from the USR call is the value of **exp**; however, if **exp** is not an integer and MKINT is called, then **var** is INT(**exp**). If you want to pass the contents of the HL register pair to **var**, JP STINT as the last instruction.

	MODEL I MODEL III	MAX-80 MODEL 4	ESOTERIC
MKINT	0A7F	0909	1B09
MKSTR	29DA	092D	1B2D
STINT	0A9A	0942	1B42

See line 17700 in sample program.

MAX-80, MODEL 4, and ESOTERIC

WPEEK Examine the contents of a 16 bit word in RAM.

WPEEK(addr)

addr = RAM address in the range of -32768 to 32767 or 0 to 65535.

WPEEK returns a value between -32768 and 32767. WPEEK is more efficient than PEEK when you want to find a word value stored in RAM.

WPEEK(**address**) is equivalent to PEEK(**address**)+256*PEEK(**address+1**).

In addition, if the sum of the PEEKs is greater than 32767, you do not have to convert for an integer expression.

EXAMPLE: You have stored a machine language routine in B$, and do not need to pass values to the routine, nor do you want to retrieve any values from the routine. This is accomplished in one statement: CALL WPEEK(VARPTR(B$) + 1)

See lines 17500 and 19200 in sample program.

MAX-80, MODEL 4, and ESOTERIC

ZERO Zero array.

ZEROvv[,vv[,...]]

vv = any valid array variable name.

ZERO sets all string arrays to null and sets every element of a numeric array to zero. ZERO does not remove the array from RAM. Use ERASE to remove an array.

See line 33200 in sample program.

BOSS SUPERBASIC (MODEL I and MODEL III)

BOSS SUPERBASIC is an enhanced version of MODEL I and MODEL III SUPERBASIC named BBASIC/CMD. BBASIC/CMD is started similar to BASIC/CMD for the MODEL I and MODEL III. BBASIC/CMD has all of the features of BASIC/CMD with the addition of single step, trace, argument review, and program pushing commands. Upon entering BBASIC, the <@> key becomes a control key, and '@' is printed by pressing <SHIFT·SPACE>.

The additional commands available with BBASIC are invoked by pressing down the <@> key first then, without removing your appendage from this key, press one of the keys on the top row of the keyboard. Seven of these commands can be obtained through program execution by poking an appropriate number into 16667 (411BH). This is ROM's trace byte. With the incorporation of these new trace commands, the TRON and TROFF commands are disabled.

The additional commands are described with <@> preceding a character.

<@·1> Trace off

This will turn off all trace commands.

<@·2> Trace on: to VIDEO

<@·2> enables printing each line number entered in a controlled trace area: the first four video lines in the upper right corner. The first line entered, after <@·2>, prints the line number on the first video line in the upper right corner of the display prefixed by the # sign. As each new line is entered the new line number prints below the previous line number until four video lines are used. When the fifth line is entered, the new line number overprints the first line number. When the sixth line is entered the sixth line number overprints the second line number. This process continues, displaying the last four line numbers entered with the # sign prefixing the last entered line.

<@·3> Trace on: to PRINTER

<@·3> enables printing each line number, prior to execution, onto the printer. The printout is similar to:

```
LPRINT USING "######";LINE;
```

where LINE is the line number. This format, using six print positions per line number, enables you to control the printout by adjusting the FORMS width prior to using TRON 1.

SINGLE STEPPING

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

<@·4> Single Step off

<@·4> turns off the single step command and enables the program to run as normal. If the trace command was in use, it will continue to command until turned off via the <@·1> command.

<@·5> Single Step to the end of a Line

<@·5> pauses execution of the BASIC program at the end of each line entered until any key is pressed. The trace to video mode is initiated to show the line numbers being executed. The tracing command can be disabled by the <@·1> command, while the single step mode continues.

<@·6> Single Step Statement

<@·6> pauses execution of the BASIC program at the end of each statement until any key I pressed. The trace to video mode is initiated to show the line numbers being executed. This command can be useful but be wary of using it if a program contains lines sunch as:

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

To single step statement through this line through this line would require 201 presses of a key. Instead use **Single Step to the end of a Line,** <@·5>, for this type of line. Reference: BREAK POINTS.

<@·7> Single Step with Timed Wait

<@·7> executes one program line or statement, pauses for a predetermined amount of time, then continues. The trace to video mode is initiated to show the line numbers being executed. <@·5> and <@·6> become sub-commands after the <@·7> command is initiated. Pressing <@·6> after <@·7> is initiated causes the delay to occur at a statement separator, in addition to the beginning of a line. Pressing <@·5> causes the delay to occur at the beginning of a line. To speed up execution (decrease delay [about halves]) press <@·↑>. To slow down execution (increase delay [about doubles]) press <@·↓>. The delay has nine settings: approximately 3 milliseconds, 7 milliseconds, 14 milliseconds, 29 milliseconds, 60 milliseconds, 120 milliseconds, 241 milliseconds, 482 milliseconds, and 964 milliseconds at normal CPU speed. When BBASIC is initializes, the setting is approximately 241 milliseconds.

BREAK POINTS

The trace and single step commands previously described can be invoked while a program is running by inserting as many POKE statements into the program where the command is desired. The following codes are used:

Command	POKE 16667	Key equivalent
Trace off	1	<@·1>
Trace on: to VIDEO	2	<@·2>
Trace on: to PRINTER	3	<@·3>
Single Step off	4	<@·4>
Single Step to the end of a Line	5	<@·5>
Single Step Statement	6	<@·6>
Single Step with Timed Wait	7	<@·7>

BACKGROUND ARGUMENT PRINTING is primarily a debugging tool that enables you to print variable values or complex calculations without messing up the display. And all of the MINIDOS commands are available. You can use this ability to save the screen to perform a MINIDOS command and return to the previous display contents intact.

BBASIC (MODEL I and MODEL III
SUPERBASIC and ESOBASIC (MAX-80, MODEL 4, and ESOTERIC)

You can save contents of the video display, suspend program execution, and select or review argument values. When argument reviewing is complete, the display is restored and execution resumed. If insufficient memory is available to save the contents of the video display, a graphic block (143d) appears in the upper right of the video, and selecting and/or reviewing argument is inhibited.

SELECTING ARGUMENTS

@·N = select arguments - **N** for **N**ew arguments (MODEL I and MODEL III BBASIC)
CTRL·N = select arguments - **N** for **N**ew arguments (MAX-80, MODEL 4, and ESOTERIC)

Pressing **CTRL·N** selects the arguments to print during program execution. This command can be entered at any time, before the program runs or during program execution. After invoking **CTRL·N**, the screen clears and the query 'Maximum length? ' is displayed. Respond from the following choices:

Response	Result
<BREAK>	exit and return to BASIC program (previous arguments intact)
1	1 character argument
2 or 3	maximum of 3 character argument
4 - 7	maximum of 7 character argument (default)
8 - 15	maximum of 15 character argument
16 - 31	maximum of 31 character argument
32 - 63	maximum of 63 character argument
64 - 99	maximum of 127 character argument
<ENTER>	defaults to maximum of 7 character argument

The maximum number of arguments for review is limited by the length selected:

Maximum argument length	Maximum number of arguments
1	128
3	64
7	32
15	16
31	8
63	4
127	2
255	1

Please note that the length includes all characters and key words are tokenized into one byte: A$(21,5) is eight characters, F(R(3,8)) is nine characters, and LOG(A/(B+C)) is ten characters in length (LOG is one). After successfully entering a length, the message 'Enter arguments:' is displayed and all previously entered arguments are erased. **CTRL·N** accepts anything that can be printed and fits in the argument length.

Once all of the arguments to review are entered, press <BREAK> to continue with the review. If the maximum number of arguments allowed is entered, **CTRL·O** as described next is automatically invoked.

REVIEWING ARGUMENTS

@·O = reviewing arguments - **O** for **O**ld arguments (MODEL I and MODEL III BBASIC)
CTRL·O = reviewing arguments - **O** for **O**ld arguments (MAX-80, MODEL 4, and ESOTERIC)

Pressing <CTRL·O> during program execution saves the contents of the video display and the message:

'<C>hange <D>elete <I>nsert <T>op'

appears along with the first argument and its value. Arguments are displayed in the sequence entered with **CTRL·N**. If insufficient memory is available to save the contents of the video display, a graphic block (143d) appears in the upper right of the video, and the program continues.

- Pressing <BREAK> resumes program execution and returns the original video display.
- Pressing <C> erases the last argument displayed, and puts you in an input mode. The new argument and its value are displayed immediately after entered. Remember, the argument is limited in length per the original choice when **CTRL·N** was selected.
- Pressing <D> deletes the last displayed argument.
- Pressing <I> inserts an argument prior to the last displayed argument.
- Pressing <T> clears the display and presents you with the first sequenced argument.
- Pressing any key other than <BREAK>, <C>, <D>, <I>, and/or <T> advances the display to the next argument in sequence.

If an attempt to print an invalid argument, the message 'Redo' is displayed and the <C> command invoked. A valid argument is required to exit from this command. If an element of an array (subscript < 11) is reviewed and the array has not yet been dimensioned by the program, then this array is dimensioned for eleven elements (0-10). If the program subsequently attempts to dimension this array via a DIM instruction, an error occurs. Therefore, I recommend you dimension all used arrays before review.

CMD"I" Insert arguments for background printing.

CMD"I",len;arg;arg;…;arg

len = maximum length of argument
arg = argument or variable – each preceded by a semicolon

CMD"I" is a program statement that sequences arguments similar to pressing **CTRL·N** and entering arguments. The maximum number of arguments depends on the length specified; and, if the maximum number of arguments is exceeded, a syntax error results.

MAX-80, MODEL 4, and ESOTERIC
CMD"N" is a program statement with no arguments or parameters that effects a **CTRL·N**.
CMD"N" should not be used in the command mode, use **CTRL·N** in command mode.

CMD"O" is a program statement with no arguments or parameters that effects a **CTRL·O**.
CMD"O" should not be used in the command mode, use **CTRL·O** in command mode.

STACKING BASIC PROGRAMS (MODEL I and MODEL III)

BASIC programs can be stacked (saved) into high memory while working or running another program. Of course, this ability is limited by the amount of free memory space available. There are five commands for this function.

<@·-> = Save the resident BASIC program into high memory
<@·:> = Recall the last saved program from high memory
<@·0> = Recall the next to last saved program from high memory
<@·9> = Append the next to last saved program from high memory to the resident BASIC program
<@·8> = Append the last saved program from high memory to the resident BASIC program

Each of these commands invokes a CLEAR 50 and halts program execution.

Pressing **<@·->** saves the resident program into high memory and adjusts memory size to the beginning of the saved program, thus protecting it from BASIC. The original program is intact if memory permits. Subsequent saves can be made as desired because memory size is adjusted. When a program is saved, execution halts and a tall vertical bar (graphic 170d) appears in the upper right corner (relative position 62) of the video display. This is to the right of the line number trace region, indicating a program has been saved into high memory and program is in BASIC program space for your execution or modification. If insufficient memory is available to save a program and maintain it in BASIC's program space, the program is saved and NEW invoked. This condition is indicated by a small block (graphic 130d) in the upper right of the video display.

Pressing **<@·:>** recalls the last saved program from high memory, overlaying the resident program and adjusting memory size appropriately. If there is no saved program when **<@·:>** is executed, the error message "Nothing to recall." is displayed.

Pressing **<@·0>** recalls the next to the last saved program from high memory, overlaying the resident program and adjusting memory size appropriately. **<@·0>** provides a means to switch the resident program with the last saved program by saving the present program with **<@·:>**; and, then recall the next to the last saved program with **<@·:>**. If only one program is saved when **<@·0>** is executed, the error message "Only one program saved." is displayed.

Pressing **<@·9>** recalls the next to the last saved program and appends it to the resident program.

Pressing **<@·8>** recalls the last saved program and appends it to the resident program.

Line number sequence is mandatory for proper execution of the appending commands. The recalled program should have its lowest line number greater that the highest line number if the resident program. The **<@·9>** and **<@·8>** commands append, they do not merge.

SUPERBASIC FILE MANIPULATION

MAX-80, MODEL 4, and ESOTERIC

CHAIN Load and execute a program keeping variable values.

```
        CHAIN var$[,B]
    or  CHAIN "filespec"[,B]
```

This command chains programs which are too large to fit in memory. This command loads and executes the file var$ and/or filespec beginning with the lowest line number as if it were a new BASIC program. Previous arguments remain intact, except for DEFFN arguments. The B option loads and runs the program without closing any file buffer areas. NOTE: SORT/BOL must be on a system disk, mounted in logical drive 0.

KILL Delete a file from a disk.

```
        KILL var$
    or  KILL "filespec"

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

This command deletes a file from the directory. If the statement is within the program, then the file should be closed first. If KILL is executed in the command mode, KILL closes all file buffer areas before it attempts to remove the target filespec.

LOAD Load a BASIC program from disk into RAM.

```
        LOAD var$[,B]
    or  LOAD "filespec"[,B]

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

This command loads a BASIC program from disk. The B option loads the program without closing any file buffer areas. MAX-80, MODEL 4, and ESOTERIC: LOAD performs a CLEAR 50.

MERGE Combine two programs in RAM. {changes program}

```
        MERGE var$
    or  MERGE "filespec"

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

This command combines the program currently in RAM with the program lines in var$, and closes all file buffer areas. If a line number in var$ coincides with a line number in the resident program, then the resident line is overwritten. MERGE leaves you in the command mode.

MODEL I and MODEL III

NAME Load and execute a program keeping most variable values.

```
        NAME var$[,B]
    or  NAME "filespec"[,B]
```

This command chains programs which are too large to fit in memory. This command loads and executes the file var$ and/or filespec beginning with the lowest line number as if it were a new BASIC program. Previous arguments remain intact, except for DEFFN, strings created via READ, and string assignments directly in program text. The B option loads and runs the program without closing any file buffer areas.

RUN Load a BASIC program and execute.

```
        RUN var$[,B]
    or  RUN "filespec"[,B]
```

This command loads a BASIC program from disk and immediately executes the program at the lowest program line. The B option loads and runs the program without closing any file buffer areas. MAX-80, MODEL 4, and ESOTERIC: RUN "filespec" performs a CLEAR 50.

SAVE Save a BASIC program to disk.

```
        SAVE var$[,A]
    or  SAVE "filespec"[,A]

          A = save the file in ASCII format
```

This command transfers the RAM resident BASIC program to disk overwriting filespec if it already exists. If the A option is not specified, the file is saved in compressed format. Compressed format uses less disk space; and, LOADs and SAVEs are faster. The ASCII format, when list from within DOS, is easier to read; and, is used to transfer a BASIC program to another BASIC interpreter that uses different 'tokens' for key words.

EXAMPLE:

```
        SAVE "FKEY/BAS:2",A
```

This command saves the file FKEY/BAS, in ASCII format, to logical drive number 2.

NOTE: When you see "filespec" within quotes. The drive number, if part of the filespec, is also enclosed in the quotes. e.g., "PROGMAX/BAS:2". The '**:2**' is within the double quotes.

SUPERBASIC FILE ACCESS

OPEN Set the mode and assign a file buffer area to a filespec.

(1) **OPEN** mode, buf, var$
(2) **OPEN** mode, buf, var$, udrl

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

mode	access mode
D	RANDOM I/O (direct-access) to an existing file.
E	[EXTEND] SEQUENTIAL OUTPUT to a file.
I	SEQUENTIAL INPUT from an existing file.
O	[OVERWRITE] SEQUENTIAL OUTPUT to a file.
R	RANDOM I/O (direct-access) to a file.

buf = the file buffer area, in the range of 1 to ff, to be assigned to var$. ff is the number of file buffer areas made available during SUPERBASIC initialization.
var$ = a string defined as a filespec. e.g., "BUFFER/TXT:2"
udrl = user defined record length. If not specified, 256 is used.

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

```
OPEN Q$,N,P$
```

Opens the file CHECKING/TXT for sequential input in file buffer area 2.

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

Opens the file BALANCE/TXT, and assigns file buffer area 3 to the file.

RANDOM I/O (direct-access): If mode = D, **udrl** is optional because **udrl** is obtained from the filespec. If mode is R, **udrl** updates the logical record length of the filespec when the file buffer area is closed. Therefore, you can use R to change the logical record length of a file. If you want to use user defined record lengths, then you must specify V when you initiate BASIC. NOTE: CUSTOM/CMD enables you to modify [B]BASIC[H]/CMD to always initiate with the expanded buffer areas.

EXAMPLES:

```
OPEN "D", 4, "TAXES/DAT", 124
```

This statement expects the file TAXES/DAT to exist and opens the file using the user defined record length when the file was created. The ',124' is ignored and does not cause an error.

```
OPEN "R", 4, "MENU/DAT", 174
```

This statement opens the file MENU/DAT if it exists on a mounted disk, or creates the file MENU/DAT if it does not exist. The user defined record length of 174 is used. All access to the file uses 174 as the **udrl**. If the file MENU/DAT was previously created with a user defined record length other than 174, then the previous **udrl** is replaced with 174 when the file is closed.

CLOSE Closes file buffer areas by buffer number.

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

buf = an expression with a value of 1 to 15. If buf is omitted, all open file buffer areas are closed.

EXAMPLE:

```
CLOSE # 3     Close file buffer area 3.
CLOSE 8,2,4   Close file buffer areas 2, 4, and 8.
CLOSE T       Close file buffer area equal to the value of T.
```

In addition to CLOSE, KILL"filespec" in the command mode and the 19 commands and statements that change a program (see the C command) close all file buffer areas.

INPUT# OPEN"I" read data command.

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

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

This statement inputs data sequentially from a disk file starting at the beginning of the file. And to INPUT# data successfully, you need to know how the data was placed on the disk. If the data is going to a string variable, BASIC ignores leading spaces, and begins the INPUT# with the first non-space character. A comma not enclosed in quotes and/or a carriage return (decimal 13) terminates the input assignment. If non-quoted text is encountered, followed by quoted text, then the quoted text also becomes part of the string variable assignment. If the data is going to a numeric variable, a comma, trailing space, and/or carriage return in the file terminates the variable assignment. The end of file terminates variable assignment also.

EXAMPLE:

```
INPUT#2, J, K
```

Sequentially inputs two numeric items from disk and places them into J and K.

LINE INPUT# OPEN"I" read a line of text command.

LINE INPUT #buf,var$

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

LINE INPUT# inputs all characters from the current position up to a carriage return (decimal 13), the end of file, and/or 255 characters, whichever comes first. LINE INPUT# differs from INPUT# in that only a carriage return is the terminator. A space is optional between LINE and INPUT#, i.e., LINE INPUT# or LINEINPUT#.

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

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

buf = a file buffer area 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 disk.

The delimiter, **m**, can be a semi-colon or comma. The use of these delimiters determines the format on the disk. If the comma is used, the items are zoned in 16 byte areas on the disk, just as a PRINT statement zones the data on the display. If the data is string data with punctuation, then the delimiter should be **;CHR$(34);**.

EXAMPLE: let X$ = "JONES, TOM", and Y$ = "SMITH, PAUL"
PRINT#2, CHR$(34);X$;CHR$(34);Y$;CHR$(34)

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

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

buf = a file buffer area 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$2 = the variable name of the second field.
... = subsequent len TO str$ (len AS str$) pairs for the balance of the buffer area size. The size is determined by the user for created files, or by the file's directory entry for existing files.

NOTE: More than one set of arguments can point to the same file buffer area.

EXAMPLE:
The data is stored in the following format. The first 64 bytes contain the client's name and address. The next 14 bytes contain the phone number.

```
FIELD 1, 56 TO AD$
FIELD 1, 78 TO AP$, 122 TO HG$
```

These statements assigns the first 56 bytes to AD$, the first 78 bytes to AP$, and 122 bytes to HG$. Printing AD$ would print the client's name but not the phone number. However, printing AP$ would print the phone number as well as the address.

MKI$, MKS$, and **MKD$** Data converters: numbers to string.

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

Numeric data must be converted to strings because all of the data items for a random (direct-access) file are defined as strings. 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.

EXAMPLE:

```
RSET PAY$ = MKI$(K%)
```

This statement converts K% to a two byte string, then RSETs this string into the variable PAY$.

```
LSET TAX$ = MKD$(K%*10)
```

This statement converts K%*10 into an eight byte string, then LSETs this string into the variable TAX$.

Typically these functions are used after an OPEN, FIELD, and before a PUT statement. See example on next page.

CVI, **CVS**, and **CVD** Data converters: string to numbers.

When reading a random (direct-access) file, certain string data must be converted back to numeric data because all of the data items for RANDOM I/O are defined as strings. Data conversion is handled using one of the following functions:

CVI (str$) str$ must be at least 2 bytes long
CVS (str$) str$ must be at least 4 bytes long
CVD (str$) str$ must be at least 8 bytes long

EXAMPLE:

```
TH%=CVI(WF$): K!=CVS(FR$)-3
```

The string arguments WF$ and FR$ were assigned to a file buffer area via the FIELD statement.

Typically these functions are used after an OPEN, FIELD, and GET statement.

EXAMPLE:

```
100 OPEN"D",1,"TEST/DAT"
110 FIELD 1, 2 TO JK$, 8 TO KL$, 30 TO LM$
120 GET 1, 1
130 BYTES! = CVD(KL$)
140 AGE% = CVI(JK$)
150 NAME$ = LM$
160 ...
```

LSET, RSET Place data in a RANDOM file buffer area.

```
        LSET str$ = exp$  RSET str$ = exp$

          str$ = a string variable usually assigned to a file buffer area via FIELD
          exp$ = the assignment for str$
```

EXAMPLE:

```
        LSET WF$ = MKI$(J%)
        RSET DR$ = B$ + C$
```

LSET left justifies the data and RSET right justifies the data. LSET and RSET do not increase the length of **str$**. If **exp$** is longer than **str$**, then the characters to the right are truncated.

LSET and/or RSET assigns **exp$** to **str$** even if **str$** is not a FIELD variable.

EXAMPLE:

```
        10 A$ = "TESTING"
        20 LSET A$ = "MULTIDOS"
```

After this program is run, the listing is:

```
        10 A$ = "MULTIDO"
        20 LSET A$ = "MULTIDOS"
```

Remember, the length of **str$** is not increased. If **str$** is not defined (length = 0) when an LSET and/or RSET statement is encountered, then the LSET and/or RSET statement simply uses space.

Typically these functions are used after an OPEN, FIELD, and before a PUT statement.

EXAMPLE:

```
        100 OPEN"R",1,"TEST/DAT",40
        110 FIELD 1, 2 TO FD$, 8 TO MD$, 30 TO XR$
        120 INPUT "What is your age";AG
        130 INPUT "What is your name";NM$
        140 PRINT "OK ";NM$;", How many bytes can a Z80 address";
        150 INPUT BT!
        160 PRINT "I will keep a record of this data"
        170 F$=MKI$(AG)
        180 LSET FD$ = F$
        190 LSET MD$ = MKD$(BT!)
        200 LSET XR$ = NM$
        210 PUT 1,1
```

PUT RANDOM write statement.

PUT [#]buf[,rec]

buf = a file buffer area 1 to 15.
rec = the record number (1 to 65535 ***or*** 1 to 32767 and -32768 to -1). If rec is omitted, then the current record is used. The current record is one unit greater than the last record written.

The PUT statement moves data from the file buffer area to a specified record in a file. Each PUT statement writes to a file if the record length is 256 bytes. However, if record length other than 256 is used, (only possible using the V option at BASIC initialization), then BASIC waits until the buffer is full before a write is executed.

```
OPEN"R",1,"POKER/TXT",67.
```

If PUT statements were to write records 1, 2, 3, and 4 in this sequence, then it would take 4 PUT statements before a write is made to the file. The PUT statement requires the following actions to occur before a write:

1. OPEN a file, using modes "R", or "D", assigning a buffer area for I/O.
2. FIELD the buffer, assigning str$ to specific positions in the buffer.
3. LSET or RSET the data into the buffer area. The converting of numeric data to string data (MKI$, MKS$, or MKD$) can take place prior to the LSET or RSET statements and/or the conversion can take place during the LSET and RSET statements.
4. PUT the data into the record via PUT buf[,rec].

If the record number in the PUT statement is greater than the number of records in the file, then the PUT statement increases the file by the necessary length to accommodate the record.

EXAMPLE:

```
100 OPEN"R",1,"TEST/DAT",40
110 FIELD 1, 2 TO FD$, 8 TO MD$, 30 TO XR$
120 INPUT "What is your age";AG
130 INPUT "What is your name";NM$
140 PRINT "OK ";NM$;", How many bytes can a Z80 address";
150 INPUT BT!
160 PRINT "I will keep a record of this data"
170 F$=MKI$(AG)
180 LSET FD$ = F$
190 LSET MD$ = MKD$(BT!)
200 LSET XR$ = NM$
210 PUT 1,1
```

GET RANDOM record reader.

GET [#]buf[,rec]

buf = a file buffer area 1 to 15.
rec = the record number (1 to 65535 ***or*** 1 to 32767 and -32768 to -1). If rec is omitted, then the current record is used. The current record is one unit greater than the last record read.

The GET statement reads data from the file to the file buffer area, reading sectors as necessary if the logical record length is less than 256. The GET statement requires the following to occur before a read:

1. OPEN the file (OPEN"R",buf,filespec[,udrl] or OPEN"D",buf,filespec)
2. FIELD the buffer area (FIELD buf, 8 TO X$, 23 TO Y$, ...)
3. GET the record
4. CVI, CVS, CVD as necessary.

EXAMPLE:

```
100 OPEN"D",1,"TEST/DAT"
110 FIELD 1, 2 TO JK$, 8 TO KL$, 30 TO LM$
120 GET 1, 1
130 BYTES! = CVD(KL$)
140 AGE% = CVI(JK$)
150 NAME$ = LM$
160 ...
```

EOF End of file detector.

EOF (buf)

This function returns a zero (false) if the end of file has not been read. Otherwise a -1 (true) is returned. EOF(n) is true if LOC(n) = LOF(n).

LOC File location indicator.

LOC (buf)

This function returns the current record read for OPEN"D" and OPEN"R". For OPEN"I", LOC returns the 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).

ERR/ERROR CODES and ERROR MESSAGES

ERR	ERROR	MESSAGE
0	1	NEXT without FOR.
2	2	Syntax error.
4	3	RETURN without GOSUB.
6	4	Out of DATA.
8	5	Improper parameter used.
10	6	Overflow.
12	7	Insufficient MEMORY.
14	8	Line number non-existent.
16	9	Subscript undefined.
18	10	Redimensioned array.
20	11	Division by zero.
22	12	Misuse of "INPUT". (MODEL I and MODEL III)
22	12	Undefined USR function. (MAX-80, MODEL 4, and ESOTERIC)
24	13	Assignment mismatch.
26	14	Insufficient string space.
28	15	String is longer than 255 bytes.
30	16	String operation to complex.
32	17	Cannot continue.
34	18	No RESUME.
36	19	RESUME without error.
38	20	Undefined error.
40	21	Missing operand.
42	22	Bad file data. (MODEL I and MODEL III)
42	22	EXIT without FOR. (MAX-80, MODEL 4, and ESOTERIC)
44	23	User abort.
46	24	Incorrect dimension cardinality.
48-98	25-50	Undefined error.
100	51	Field organization exceeded the LRL.
102	52	Disk I/O error, use CMD"E" for specific.
104	53	The buffer number is not available.
106	54	File is not in specified directory.
108	55	Incorrect file mode.
110	56	File buffer previously assigned.
112	57	Disk read error, use CMD"E" for specific.
114	58	Disk write error, use CMD"E" for specific.
116	59	Incorrect password used to access file.
118	60	EOF encountered.
120	61	Drive not available.
122	62	Disk space full.
124	63	EOF reached before any characters read.
126	64	Attempted to access record ZERO.
128	65	Improper file name.
130	66	Access mode differs from OPEN mode.
132	67	I/O buffer overflow.
134	68	Directory space full.
136	69	Write protected media.
138	70	Access attempted to protected file.
140	71	Full directory. The file cannot be extended.
142	72	The buffer has not been assigned.
144	73	Undefined function.
146	74	File not found.
148	75	Sequence overlaps.
150 up	76 up	Undefined error.

MODEL I and MODEL III ERROR values above 23 cannot be simulated with ERROR ***code***.

ERROR MESSAGE DEFINITIONS

Access attempted to protected file.
· ACCESS password given, but UPDATE password required.

Access mode differs from open mode.
· The usage of "PRINT#" with a buffer opened by mode "D", "I", and/or "R".
· The usage of "INPUT#" with a buffer opened by mode "D", "E", "O", and/or "R".
· An attempt to PUT or GET to a file opened by mode "E", "I", and/or "O".

Assignment mismatch.
· An attempt to assign a string expression to a non-string variable.
· An attempt to use a string variable as a numeric variable.
· An attempt to assign a numeric expression to a string variable.
· An attempt to use a numeric variable as a string variable.

Attempted to access record ZERO.
· The record number via PUT and/or GET is evaluated as being zero.

Bad file data.
· The tape data does not match the variable (Assignment mismatch) or is unreadable.

Cannot continue.
· An attempt to continue program execution after a CLEAR is invoked - directly and/or indirectly.

Directory space full.
· All DIREC slots are used. See ZAP/CMD for DIREC detail.

Division by zero.
· In division, the denominator and/or the divisor is evaluated as zero.

Disk I/O error, use CMD"E" for specific.
· Use CMD"E", then refer to the DOS error message definitions.

Disk read error, use CMD"E" for specific.
· Use CMD"E", then refer to the DOS error message definitions.

Disk write error, use CMD"E" for specific.
· Use CMD"E", then refer to the DOS error message definitions.

Disk space full.
· The file being saved or updated requires more disk space than available.

Drive not available.
· The DCT entry is NULL, and/or a disk is not rotating in the specified drive.

EOF encountered.
· An attempt to CHAIN, LOAD, and/or RUN an empty file.
· A GET to a user defined record length file which extends beyond the file size (the file size is not an exact multiple of the logical record length).

EOF reached before any characters read.
· An attempt to input beyond the end of a sequential file.

EXIT without FOR.
· EXIT is used without any FOR-NEXT loops active. (CLEARn nullifies all FOR-NEXT loops.)

Field organization exceeds the logical record length.
· The sum of the "len"'s for the "len TO str$" ("len AS str$") pairs in a FIELD statement is greater than the logical record length of the file.

File buffer previously assigned.
· The file buffer number is already assigned to another file. The file you are trying to open may or may not be the same filespec.

File is not in specified directory.
· Filespec not found on disk in logical drive specified. This error is only displayed when the drive number is part of the filespec and the file is not on the specified logical disk.

File not found.
· Filespec is not on any mounted disk.

Full directory. File cannot be extended.
· A file is being expanded and requires an extended directory entry; however, there are no more DIREC slots available.

I/O buffer overflow.
· More than 255 characters encountered since the last terminator while loading a program stored in ASCII format. Or a line number is not after the last terminator. Terminators for ASCII files are 00H and 0DH.

Improper file name.
· A null filespec.
· The filespec exceeded 24 characters (filename/ext.password:d' = 24).
· Filespec syntax incorrect. e.g., LOAD"(/':)"

Improper parameter used.
· An attempt to use an out of range argument.
· An attempt to dimension (DIM) and/or access an array with a negative subscript.

Incorrect dimension cardinality.
· An attempt to access an array with a different number of elements the array was dimensioned to.

Incorrect file mode.
· An attempt to open a file without using "D", "E", "I", "O" and/or "R".
· An attempt to open a random file with the logical record length not equal to 256, and the "V" option was not specified when SUPERBASIC initialized to allocate the additional 256 bytes for the file buffer area.
· An attempt to FIELD a file buffer area and the file was opened via "E", "I", and/or "O".
· An attempt to MERGE a file saved in compressed format (not saved via "filespec",A).

Incorrect password used to access file.
· A file is password protected and the incorrect password given.

Insufficient MEMORY.
· Insufficient MEMORY available to execute the statement or command.
· An attempt to DIM an array that requires more memory than available.

Insufficient string space.
· The operation requires more string space than the amount allocated.
(MAX-80, MODEL 4, and ESOTERIC: you can issue a CLEARn then continue the program.)

Line number non-existent.
· A statement contains a reference to a line and/or "label" that is not in the program text.

Missing operand.
· An operator was encountered without a necessary operand following.

Misuse of "INPUT".
· An INPUT statement was keyed in the command mode.

NEXT without FOR.
· A NEXT statement without a corresponding FOR statement. (CLEARn nullifies all FOR-NEXT loops.)

No RESUME.
· The program naturally terminated with error-trapping active.

Out of DATA.
· A READ attempt without sufficient DATA items.

Overflow.
· An attempt to assign a value greater than 32767 and/or less than -32768 to an integer variable.
· An attempt to evaluate an expression, even interim results, to an absolute value that is greater than 1.70141E38 in single-precision and/or an absolute value that is greater than 1.701411834604692D38 in double-precision.
· An attempt to renumber a program where a new number would exceed the maximum line number allowed.

Redimensioned Array.
· An attempt to dimension an array previously dimensioned - by a previous DIM statement or by default (11).

RESUME without error.
· A RESUME was encountered without a current ON ERROR GOTO active.

RETURN without GOSUB.
· A RETURN was encountered without a GOSUB active.

Sequence Overlaps.
· An attempt to renumber where the new sequence would create a line number higher than a line number not being re-sequenced. e.g.,

String is longer than 255 bytes.
· An attempt to assign more than 255 characters to a string variable.

String operation too complex.
· More than ten sub-string operations pending.

Subscript undefined.
· An attempt to access an array element undefined by the DIM statement.

Syntax error.
· A variable with a key word embedded. (DIS**TAN**CE, C**LOC**K, VER**IF**Y)
· Missing and/or wrong punctuation.
· Misspelled key word.

The buffer has not been assigned.
· An attempt to access a file buffer area not assigned to a file.

The buffer number is not available.

· The buffer number is less than one and/or greater than 15.
· An attempt to access a file buffer area greater than the number specified when SUPERBASIC initialized.

Undefined function.

· An attempt is made to execute a FN statement and the specified function was not previously defined.

Undefined user function.

· An attempt is made to execute a USR statement and the specified USR number was not defined via DEFUSR.

User abort.

· A deliberate error produced by ERROR 23. This is used for debugging.

Write protected media.

· An attempt to write to a write protected disk.

KEY WORDS in MAX-80, MODEL 4 and ESOTERIC BASIC (in alphabetical order)

Word	Dec	Hex	Word	Dec	Hex	Word	Dec	Hex	Word	Dec	Hex
'	251	FB	DEFSTR	152	98	LIST	180	B4	RESUME	159	9F
*	207	CF	DIM	138	8A	LLIST	181	B5	RETURN	146	92
+	205	CD	ELSE	149	95	LOAD	167	A7	RIGHT$	249	F9
-	206	CE	END	128	80	LOC	234	EA	RND	222	DE
/	208	D0	EOF	233	E9	LOF	235	EB	**ROW**	**255**	**FF**
<	214	D6	**ERASE**	**182**	**B6**	LOG	223	DF	RSET	172	AC
=	213	D5	ERL	194	C2	LPRINT	175	AF	RUN	142	8E
>	212	D4	ERR	195	C3	LSET	171	AB	SAVE	173	AD
ABS	217	D9	ERROR	158	9E	MEM	200	C8	SET	131	83
AND	210	D2	**EXIT**	**179**	**B3**	MERGE	168	A8	SGN	215	D7
ASC	246	F6	EXP	224	E0	MID$	250	FA	SIN	226	E2
ATN	228	E4	FIELD	163	A3	MKD$	238	EE	**SORT**	**157**	**9D**
BIN$	**254**	**FE**	FIX	242	F2	MKI$	236	EC	**SOUND**	**140**	**8C**
CALL	**174**	**AE**	FN	190	BE	MKS$	237	ED	SQR	221	DD
CDBL	241	F1	FOR	129	81	NEW	187	BB	STEP	204	CC
CHAIN	**169**	**A9**	FRE	218	DA	NEXT	135	87	STOP	148	94
CHR$	247	F7	GET	164	A4	NOT	203	CB	STR$	244	F4
CINT	239	EF	GOSUB	145	91	ON	161	A1	STRING$	196	C4
CLEAR	184	B8	GOTO	141	8D	OPEN	162	A2	TAB(	188	BC
CLOSE	166	A6	**HEX$**	**252**	**FC**	OR	211	D3	TAN	227	E3
CLS	132	84	**HR** or **WC**	**185**	**B9**	OUT	160	A0	THEN	202	CA
CMD	133	85	IF	143	8F	PEEK	229	E5	TIME$	199	C7
COS	225	E1	INKEY$	201	C9	POINT	198	C6	TO	189	BD
CSNG	240	F0	INP	219	DB	POKE	177	B1	TROFF	151	97
CVD	232	E8	INPUT	137	89	POS	220	DC	TRON	150	96
CVI	230	E6	INSTR	197	C5	PRINT	178	B2	USING	191	BF
CVS	231	E7	INT	216	D8	PUT	165	A5	USR	193	C1
DATA	136	88	KILL	170	AA	RANDOM	134	86	VAL	245	F5
DEF	176	B0	**LABEL**	**186**	**BA**	READ	139	8B	VARPTR	192	C0
DEFDBL	155	9B	LEFT$	248	F8	REM	147	93	**WPEEK**	**253**	**FD**
DEFINT	153	99	LEN	243	F3	RESET	130	82	**ZERO**	**183**	**B7**
DEFSNG	154	9A	LINE	156	9C	RESTORE	144	90	[	209	D1

Notes

All numbers are hexadecimal unless suffixed with d. When specific models are noted for an entry point, any additional entry points refer to those specific model(s) until different specific models are noted. If an entry point has multiple capabilities, some capabilities may be noted by specific models. This does not supersede the specific models for entry points.

MODEL 4 and ESOTERIC (MAX-80 either SHIFT key):
<LEFT-SHIFT·F1> cancels <LEFT-SHIFT·F2> and/or <LEFT-SHIFT·F3>

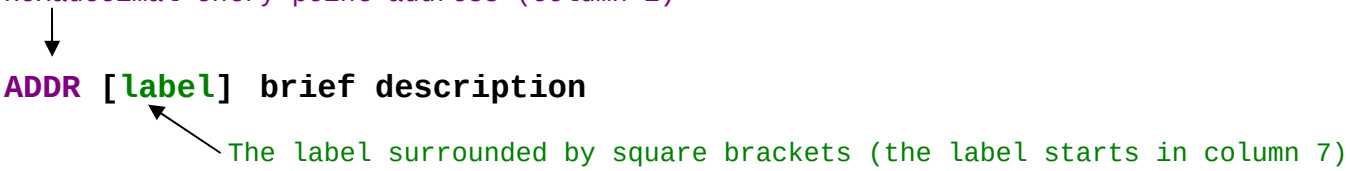
<LEFT-SHIFT·F2> activates the <CLEAR> key as an ALT key. When the <CLEAR> key is held down and a letter key, A through Z, is pressed, the value returned is the letter key code plus 40. The <LEFT-SHIFT·F2> function recognizes the condition of the CAPS mode. i.e., In CAP-LOCK <ALT·C> = 83, and <ALT·SHIFT·C> = 83. In upper/lower case <ALT·C> = A3, and <ALT·SHIFT·C> = 83.

<LEFT-SHIFT·F3> activates the <CLEAR> key as an ALTCTRL key. When the <CLEAR> key is held down and another key is pressed including <ENTER>, <BREAK>, and the arrow keys, the value returned is the key pressed plus 80. When the <CLEAR> key is held down and a shifted (either <SHIFT> key) letter key is pressed, the value returned is the letter key code plus 60. i.e., <ALTCTRL·C> = E3, <ALTCTRL·SHIFT·C> = C3. The <LEFT-SHIFT·F3> function disregards the condition of the CAPS mode.

For the values returned by these keys, please refer to Appendix A.

The following is the format used for entry points:

Hexadecimal entry point address (column 1)



MAX-80:
0000 [SUPV] CLS, VIDEO WIDTH, and FUNCTION KEYS

INPUT A	INPUT B	AF on exit	FUNCTION
0	n/a	1F	Clear display
1	n/a	00	Set VIDEO to 40x10 and clear display
2	n/a	00	Set VIDEO to 50x18 and clear display
3	0	B reg	Set F1 through F4 to their default values
3	1	B reg	Set F1 to routine @ DE and VD (see below)
3	2	B reg	Set F2 to routine @ DE and VD
3	3	B reg	Set F3 to routine @ DE and VD
3	4	B reg	Set F4 to routine @ DE and VD
3	5	B reg	Set F5 <SHIFT·F4> to routine @ DE and VD
3	>5	B reg	VD: value (DATA) with F-key jump -2
4	n/a	00	perform <SHIFT-F1>
5	n/a	00	perform <SHIFT-F2>
6	n/a	00	perform <SHIFT-F3>
>6	n/a	A-6	no function (ignored)

MODEL 4:
0000 [SUPV] CLS, VIDEO WIDTH, and FUNCTION KEYS

INPUT A	INPUT B	AF on exit	FUNCTION
0	n/a	1F	Clear display
1	n/a	00	Set VIDEO to 40x10 and clear display
2	n/a	00	Set VIDEO to 50x18 and clear display
3	0	B reg	Set F1 through F6 to their default values
3	1	B reg	Set F1 to routine @ DE and VD (see below)
3	2	B reg	Set F2 to routine @ DE and VD
3	3	B reg	Set F3 to routine @ DE and VD
3	4	B reg	Set F4 <RIGHT-SHIFT·F1> to routine @ DE and VD
3	5	B reg	Set F5 <RIGHT-SHIFT·F1> to routine @ DE and VD
3	6	B reg	Set F6 <RIGHT-SHIFT·F3> to routine @ DE and VD
3	>6	B reg	VD: value (DATA) with F-key jump -2
4	n/a	00	perform <LEFT-SHIFT-F1>
5	n/a	00	perform <LEFT-SHIFT-F2>
6	n/a	00	perform <LEFT-SHIFT-F3>
>6	n/a	A-6	no function (ignored)

With a function key routine set, the KEYBOARD scan routine executes the function, and then returns with whatever is in the A register. If a key value is also desired, then LD A with the character just before the last RET instruction. If no key value is desired, then execute a XOR A just before the last RET instruction. Function key commands/characters do not repeat.

MODEL I, MODEL III, MAX-80, and MODEL 4:
The following entry points are for the MODEL I, MODEL III, MAX-80 and MODEL 4 using MULTIDOS. For the Model 4 using ESOTERIC, please refer to Entry points for ESOTERIC for entry point addresses; and, additional entry points.

MODEL I, MODEL III, MAX-80, and MODEL 4:
0003 [IRAM] If 17, then we are in a MAX-80 or MODEL 4. If 11 then we are in ESOTERIC.

MAX-80 and MODEL 4:
0004 Contains the VIDEO width

0005 [STKEY] Jumps to special first position keystroke when 40 is called. The address of the routine is loaded into 06. After performing some test on the keystroke, if it is not what you want simply return. Otherwise, you will have to remove (at least) two calls from the stack similar to this form:

```
POP QQ  ;call to 0005
POP HL  ;restore HL register
POP QQ  ;call to 0040
```

Entry Points for MULTIDOS

MODEL I, MODEL III, MAX-80, and MODEL 4:

0008 Not used in DOS. BASIC normal RST 08.
0010 Jumps to PARSE under DOS. BASIC normal RST 10.
0013 [GETBT] Get file byte into A register: DE => FCB (opened).
0018 Compare HL to DE using only AF.
001B [PUTBT] Put byte in A register to device or file: DE => DCB or FCB (opened).
0020 Not used by DOS. BASIC normal RST 20.
0028 Load overlay without returning to address after RST 28. For the MAX-80, MODEL 4, and ESOTERIC pressing the <BREAK> key does not go here!
002B [GETKI] Scans keyboard once, returns 0 if no key, otherwise returns with ASCII value of key pressed in A register.
0030 Call to load/execute DEBUG.
0033 [PUTVO] Puts character in A register onto video display. Returns with displayed character in A and the Z flag set.
0038 Used for Interrupt Mode 1.
003B [PUTPR] Outputs character in A register to the device in the printer DCB. Returns with printed character in A and the Z flag set.
0040 [LINE] Input into (HL) B characters (jumps to 05D9, use in place of 05D9).

0049 [KWAIT] Calls 2B until a key is pressed.

MAX-80 and MODEL 4:

0050 [GETSI] Get byte from RS-232 into A register.
0055 [MACH] If 01, then we are in a MODEL I or MAX-80 environment.

MODEL 4:

0055 [PUTSO] Put byte in A register to RS-232 (yes, the 0055 is correct same as MACH).

MAX-80:

0056 [PUTSO] Put byte in A register to RS-232.

MODEL 4:

005A [IMAGE] Contains image of PORT 84.

MODEL 4:

005C [WHERE] Call to locate PC register. i.e., LD HL,PC. This is the same as 000B in the ROM.

MODEL I, MODEL III, MAX-80, and MODEL 4:

0060 [DELAY] Delay for interval. Uses BC for delay value. Exit: BC =0, A =0, “Z” set.
Some estimated delays (in microseconds)

BC	MODEL I	MODEL III	MAX-80	MODEL 4
0000	**960,472d**	**969,713d**	**944,231d**	1,034,339d
F780	928,285d	937,516d	912,879d	**999,996d**
8790	508,630d	513,510d	**500,007d**	547,723d
8544	**499,998d**	504,810d	491,535d	538,443d
83FF	495,235d	**500,001d**	486,852	533,314d
7BC0	494,297d	468,776d	456,437d	**499,996d**

MODEL 4:

0800 - 08AF RIGID disk driver area.
08B0 - 08DF Extended memory auxiliary driver area.
08E0 - 08FF Screen saver auxiliary driver area.

MAX-80 and MODEL 4:
0900 - 2FFF Application area 3: If an application needs this area; and, CMD"fffff" is available, then the application must perform instructions similar to the code segment on the last page of Entry points for ESOTERIC.

MODEL 4:
3000 - 37FF Application area 2: Used by several LIBRARY commands and utilities. An application may this area during CMD"fffff" provided the application sets bit 4 of SM1. Please see the last page of Entry points for ESOTERIC.

MAX-80:
3000 – 33FF The second page of VIDEO refresh RAM.

MODEL I, MODEL III, MAX-80, and MODEL 4:
3800 - 38FF Pseudo-RAM used for detecting key presses.

3C00 - 3FFF VIDEO refresh RAM. The MULTIDOS and ESOTERIC video codes:

CODE	ACTION
00-06	Ignored
07	Beep (MAX-80, MODEL 4, and ESOTERIC)
08	Backspace cursor one position and erase character under cursor
09	Ignored
0A	Linefeed/carriage return and erase new line
0B-0C	Ignored
0D	Linefeed/carriage return and erase new line
0E	Enable cursor character
0F	Disable cursor character
10	Enable reverse video and set reverse video high bit routine (MODEL 4 and ESOTERIC)
11	Reset reverse video high bit routine (MODEL 4 and ESOTERIC)
12	Set all VIDEO refresh RAM from cursor position to end of frame to value in (404C) (MAX-80, MODEL 4, and ESOTERIC [1221])
13	Position cursor to HOME position (does not reset WIDE display, does not disable reverse video) (MAX-80, MODEL 4, and ESOTERIC)
14	Scroll VIDEO and cursor up one line (MAX-80, MODEL 4, and ESOTERIC)
15	Swap space compression with special characters (MODEL III, MODEL 4, and ESOTERIC)
16	Swap alternate special characters with special characters (MODEL III flips bit 3 of 4210) (MODEL III, MODEL 4, and ESOTERIC)
17	Set video to WIDE display
18	Backspace cursor one position
19	Advance cursor one position
1A	Advance cursor one line
1B	Back cursor one line
1C	Homes cursor, disables reverse video, and resets WIDE display
1D	Position cursor to beginning of line
1E	Set all VIDEO refresh RAM from cursor position to end of line to 20
1F	Set all VIDEO refresh RAM from cursor position to end frame to 20
20-BF	Display character at current cursor position and advance cursor.
C0-FF	Display N-0C0 spaces and advance cursor position accordingly. If in special character mode, display character and advance cursor.

Entry Points for MULTIDOS

The following labels have the same meaning for all versions of MULTIDOS and ESOTERIC. Please refer to this table to determine the address for each model. If the address is the same for all versions of MULTIDOS then the address is shown; otherwise, the address is blank.

Label	MODEL I	MODEL III	MAX-80	MODEL 4	ESOTERIC
JCL1	404B	446C	400A	400A	1212
SPOOL	4053	4053	400B	400B	123B
LINK	4054	4054	400C	400C	123C
ROUTE	4055	4055	400D	400D	123D
MEMTP	4056	4056	400E	400E	1213
DOBUF	4058	4058	4010	4010	1215
NULLS	405A	405A	4012	4012	122A
KBDCB	4015	4015	4015	4015	120B
VODCB	401D	401D	401D	401D	1217
PRDCB	4025	4025	4025	4025	1222
TODOS	402D	402D	402D	402D	0883
ABORT	4030	4030	4030	4030	0846
SIDCB	n/a	41E5	403A	403A	122B
SODCB	n/a	41ED	403D	403D	122C
TIC	4040	4216	4040	4040	120A
Time Copy	41FB	426F	n/a	4151	1351
Time Storage	4041	4217	n/a	4217	0050
MMB	430E	4215	430E	421D	1245
OLC	430F	426A	430F	421E	1246
OLB	4310	426B	4310	421F	1247
OLA	4311	426C	4311	4220	1248
SM0	430C	426D	430C	4221	1249
SM1	430D	426E	430D	4222	124A
KIM	4047	4265	4048	4223	005E
COMBF	4318	4225	4318	4225	(0060)
STKLD	41E5	41E5	43FE	42FE	14FE
IOBUF	4200	4300	4200	4300	1500
DOS	4400	4400	4400	4400	085B
DATA	4403	4403	4403	4403	1259
EXEC	4405	4405	4405	4405	0865
ERROR	4409	4409	4409	4409	0861
DEBUG	440D	440D	4308	440D	084C
TOPMM	4049	4411	4049	4411	0062
MXLIN	n/a	n/a	4416	4416	n/a
CAT	4419	4419	4419	4419	0849
FILK	441C	441C	441C	441C	0803
INIT	4420	4420	4420	4420	081C
OPEN	4424	4424	4424	4424	0828
CLOSE	4428	4428	4428	4428	080C
DOING	442B	442B	442B	442B	1211
DLETE	442C	442C	442C	442C	080F
LOAD	4430	4430	4430	4430	081F
LRUN	4433	4433	4433	4433	0822

Entry Points for MULTIDOS

Label	MODEL I	MODEL III	MAX-80	MODEL 4	ESOTERIC
READ	4436	4436	4436	4436	0837
WRITE	4439	4439	4439	4439	083D
VERF	443C	443C	443C	443C	083A
POBOF	443F	443F	443F	443F	082E
POSN	4442	4442	4442	4442	0834
BCKSP	4445	4445	4445	4445	0806
POEOF	4448	4448	4448	4448	0831
DIVID	4451	4451	4451	4451	0858
FIXDT	445C	n/a	n/a	n/a	n/a
PUNCH	n/a	n/a	445C	445C	005A
DSTAT	n/a	n/a	445E	445E	08D5
JKL	4461	4461	4461	4461	0877
CEOF	4464	4464	4464	4464	0809
VOD	4467	4467	4467	4467	088F
PRT	446A	446A	446A	446A	088C
GTIME	446D	446D	446D	446D	086E
GDATE	4470	4470	4470	4470	0868
DEXT	4473	4473	4473	4473	0800
PARAM	4476	4476	4476	4476	087A
DOTIT	4479	4479	4479	4479	0889
ADRQ	447C	447C	447C	447C	08C6
LWAIT	4497	4497	**4496**	4497	08CC
FPOS	44A1	44A1	44A1	44A1	0812
STATP	n/a	n/a	44AB	44AB	0069
MOTON	44C9	44C9	44C9	44C9	08C9
GRDUM	44CC	44CC	44CC	44CC	086B
SPRMT	44CF	44CF	44CF	44CF	087D
IUS	4410	44D2	4410	44D2	0871
DUS	4413	44D5	4413	44D5	085E
GTDCC	44DA	44DA	44DA	44DA	08D8
GTDCT	44DB	44DB	44DB	44DB	08D9
READV	44DE	44DE	44DE	44DE	n/a
READS	44E1	44E1	44E1	44E1	08DF
WRITS	44E4	44E4	44E4	44E4	08E2
DIRRD	44E8	44E8	44E8	44E8	08E6
DIRWT	44EB	44EB	44EB	44EB	08E9
RDIRP	44EE	44EE	44EE	44EE	08EC
WDIRP	44F1	44F1	44F1	44F1	08EF
USFR	44F4	44F4	44F4	44F4	08F2
GETDT	44F7	44F7	44F7	44F7	08DC
FIXRR	n/a	n/a	44FA	44FA	08A1
DCT	4500	4500	4500	4500	0900

Although this table includes ESOTERIC, please refer to the Entry points for ESOTERIC section for additional ESOTERIC entry points.

[JCL1] Holds the nesting level of a DO file when input is required.

[SPOOL] This byte is described as follows:

BIT	IF SET	IF RESET
0	inhibit LINK	permit LINK
1	inhibit un-LINK	permit un-LINK
2	inhibit ROUTE	permit ROUTE
3	inhibit un-ROUTE	permit un-ROUTE
4	ROUTE to file active	ROUTE to file inactive
5	SPOOLER active	SPOOLER inactive
6	SPOOLER buffer full	SPOOLER buffer not full
7	SPOOLER buffer has data	SPOOLER buffer empty

[LINK] This byte is described as follows:

BIT	IF SET	IF RESET
0	PR linked to SO	PR not linked to SO
1	PR linked to DO	PR not linked to DO
2	DO linked to PR	DO not linked to PR
3	DO linked to SO	DO not linked to SO
4	SO linked to PR	SO not linked to PR
5	SO linked to DO	SO not linked to DO

Please note the next two bits are used for printer functions:

6	PR output to SO	PR output to parallel port
7	Linefeed generated after each Carriage return	Carriage return alone

[ROUTE] This byte is described as follows:

BIT	IF SET	IF RESET
0	PR routed to SO or file	PR not routed to SO
1	PR routed to DO	PR not routed to DO
2	DO routed to PR or file	DO not routed to PR
3	DO routed to SO	DO not routed to SO
4	SO routed to PR	SO not routed to PR
5	SO routed to DO or file	SO not routed to DO
6	KI routed to SI	KI not routed to SI
7	SI routed to KI	SI not routed to KI

[MEMTP] This word contains TOPMEM prior to executing DO files.

[DOBUF] This word is the pointer to current DO File Control Block.

[NULLS] Bits 7-3 contain the left margin, bits 2-0 contain the number of NULLS (00) sent after a linefeed.

```
MODEL I, MODEL III, MAX-80, and MODEL 4:
4015 [KBDCB]  These 8 bytes are defined as the KEYBOARD Device Control Block (DCB).
    DCB+0     This byte normally contains 1
    DCB+1,2   Driver address
  MODEL I and MODEL III:
    DCB+3
          BIT 0 Used by printer routine             SET = hard form feed
              1 Used by printer routine             SET = software vertical tab
  MODEL I:
    DCB+4
          BIT 5 lower case                          SET = lower case
  MODEL III:
    DCB+4
          BIT 0 upper case                          SET = upper case
  MODEL I and MODEL III:
    DCB+5,6   Blink rate
    DCB+7     Cursor blinks if this byte is zero.
  MAX-80, and MODEL 4:
    DCB+3,4   Driver address storage during route
    DCB+5
          BIT 0 Used by printer routine             SET = hard form feed
              1 Used by printer routine             SET = software vertical tab
              2 Space compression/Special           SET = special characters
              3 lower case/CAP lock                 SET = lower case
              4 key beep                            SET = key beep
              5 blinking cursor                     SET = blink
              6 graphics                      <[LEFT]-SHIFT·F2> sets this bit.
              7 high bit set                  <[LEFT]-SHIFT·F3> sets this bit.
    DCB+6   VIDEO cursor character
    DCB+7   VIDEO scroll protect. The number of lines protected is determined by the
            highest bit set. The number of lines = N+1 where bit N is set. i.e., set
            bit 4 to protect 5 lines (protects 1 to 8 lines).

MODEL I, MODEL III, MAX-80, and MODEL 4:
401D [VODCB]  These 8 bytes are defined as the VIDEO Device Control Block (DCB).
    DCB+0     This byte normally contains 7
    DCB+1,2   Driver address
    DCB+3,4   Refresh RAM position
    DCB+5     Character storage if cursor is on
  MODEL I and MODEL III:
    DCB+6     VIDEO cursor character
    DCB+7     not used
  MAX-80, and MODEL 4:
    DCB+6     VIDEO column position
    DCB+7     VIDEO row position
```

MODEL I, MODEL III, MAX-80, and MODEL 4:

4025 [PRDCB] These 8 bytes are defined as the PRINTER Device Control Block (DCB).

DCB+0	This byte normally contains a 6
DCB+1,2	Driver address
DCB+3	Maximum number of printed lines per page
DCB+4	Line counter
DCB+5	Maximum number of characters per line
DCB+6	Character counter
DCB+7	The number of non-printed lines per page

402D [TODOS] Exit to indicate the completion of a DOS command, error or no error.

4030 [ABORT] If not in BASIC, jumps to 402D after resetting Stack Pointer to the DOS stack, [STKLD], and prompts you "Insert SYSTEM <ENTER>". If in BASIC, returns you to the prompt (safe exit from unknown territory provide system RAM has not been contaminated).

MODEL III is eight bytes, and; MAX-80 and MODEL 4 are three bytes

[SIDCB] DCB for the RS-232-C input.

[SODCB] DCB for the RS-232-C output.

[TIC] This bytes increases by one for each real time interrupt.

[KIM] Points to the terminator byte after executing a DOS function.

[TOPMM] This word contains the address of the highest available memory location for DOS. BASIC temporarily sets this word to a different value during CMD"fffff" execution. The contents of (TOPMM) is referred to as TOPMEM.

MODEL 4:

4048 [BEE0] Pointer to sound routine. Tone in A and duration in E.

MAX-80 and MODEL 4:

404C [CEOFC] Stores character for write to end of VIDEO when a 12 is sent to VIDEO routine.

404D [BREAK] BREAK KEY status. If non-zero, the KEYBOARD returns a zero when <BREAK> is pressed. The DOS command mode sets this byte to zero.

404E [PARSE] Increments the HL register pair, interrogates the byte in HL, skips spaces and returns with "C" set if numeric or "Z" set if 3A and/or 0.

MODEL I, MODEL III, MAX-80, and MODEL 4:

405D [DBSP] This area (35 bytes) are used as DEBUG scratch pad when DEBUG is active.

4080 - 4151 Scratch pad used by BASIC and several DOS functions.

[MMB] Set to non-zero to inhibit Minidos. The DOS and BASIC command modes set this byte to zero. The overlay loader sets this byte to a non-zero value before loading an overlay, and resets this byte to zero after completion of the overlay function.

[OLC] Storage of the overlay mask of the overlay in the overlay area when Minidos is invoked. Upon Minidos' exit, Minidos reloads this overlay and sets this byte to zero.

[OLB] The overlay mask of the previous overlay when a new overlay is executed.

[OLA] Current overlay mask. The overlay mask is bits 5, 3, 2, 1, and 0.

[SM0] File attributes. The following comments apply if the noted bit is set.
BIT 0. After a file is OPENed this bit is set if the file has an attribute of EXEC and/or NONE.
BIT 1. Set if LRUN, 4433, is entered. This bit is reset before the program executes.
BIT 2. Ignore PASSWORDS. Reset when file is OPENed.
BIT 3. Get file attributes from special 7-byte area (doesn't RAM date). Reset after file is CLOSEd.
BIT 4. Set if EXEC, 4405, is entered. This bit is reset at the DOS command prompt.
BIT 5. Accept ! " # $ % & < > ? @ in a filespec.
BIT 6. Accept numbers as the first character in a filespec and/or extension.
BIT 7. Do not convert lowercase to uppercase. (accepts any character greater than 3D in filespec.)

[SM1] System attributes. The following comments apply if the noted bit is set.
BIT 0. Debug active.
BIT 1. Library1/EXT is in RAM area 5200 to 68FF.
BIT 2. Library2/EXT is in RAM area 5200 to 68FF.
MODEL I and MODEL III:
BIT 3. APPEND and/or RESTOR invoked in Library2/EXT.
BIT 4. DIR invoked in Library1/EXT
MAX-80, MODEL 4, and ESOTERIC
BIT 3. Library3/EXT is in RAM area 5200 to 68FF.
MAX-80:
BIT 4. 3400 to 36FF and/or 3900 to 3BFF has been used.
MODEL 4:
BIT 4. 3000 to 37FF has been used.
ESOTERIC:
BIT 4. 4200 to 49FF has been used.
BIT 5. In BASIC. This bit is reset during CMD"fffff".
BIT 6. Type ahead has bytes in the type ahead buffer.
BIT 7. DO active.

[COMBF] DOS command mode input buffer. MODEL I and MODEL III are 64 bytes. MAX-80 MODEL 4, and ESOTERIC are 81 bytes.

[STKLD] The DOS stack area is quite sufficient. For the MAX-80 and MODEL 4, the stack has ABORT above STKLD in case a return is invoked with no return pending. The stack depth is as follows:

MODEL I	MODEL III	MAX-80	MODEL 4	ESOTERIC
175d	175d	149d	136d	144d

For both the MAX-80 and MODEL 4, the command buffer is just below the stack area (sort of gives you another 81 bytes).

Time Storage is in ssmmhhyyddmm format. If the interrupts are enabled Time Storage is updated once per second.

Time Copy is a copy of Time Storage that is used to set the date and time (approximate time) if the computer is re-booted. NOTE: MODEL I time copy does not have seconds (format is mmhhyyddmm).

MODEL I and MAX-80:

4200 [IOBUF] General purpose 100 byte I/O buffer. Reference MODEL III/4 as 4300.

MODEL III and MODEL 4:

4300 [IOBUF] General purpose 100 byte I/O buffer. Reference MODEL I & MAX-80 as 4200.

4400 [DOS] Entry to load Command/DOL and enter the DOS command level. The Stack Pointer is loaded with the system stack, STKLD, and DO is checked. If DO is active, key characters are obtained from the DO file. If DO is not active, the contents of KIM is examined to see if the terminator was a comma, indicating a multiple DOS command. If the contents of KIM is not a comma, then 'MULTIDOS' is printed and you may input a DOS command.

4403 [DATA] This word has information for extension data.

4405 [EXEC] Entry to execute the command at HL. On exit the HL is at the first byte past the terminator - comma and/or space. {CALL EXEC}

4409 [ERROR] Entry point to DOS error library. Error code is the lower 6 bits of the A register. If bit 6 is set, then the decimal error is not printed. If bit 7 is set, then this routine returns to the caller. If bit 7 is not set, this routine exits to 402D.

[DEBUG] Debugger load/execute code.

[TOPMM] This word contains the address of the highest available memory location for DOS. BASIC temporarily sets this word to a different value during CMD"fffff" execution.

MAX-80 and MODEL 4:

4416 [MXLIN] VIDEO maximum line condition. Returns in 40X10: A=40, D=10; in 50X18: A=50, D=18. i.e., A call to this address returns the number of columns in the A register and the number of rows in the D register.

4419 [CAT] - Obtain directory (unsorted) on a mounted disk.

Entry:
A = not used.
B = function. (see below)
C = logical drive spec.
DE = not used.
HL = RAM area, if function directs to RAM. You must either reserve all of memory from (TOPMM) to FFFF or ensure that this is below (TOPMM).

Temporary exit: (if any of bits 3,4, and/or 5 is set in B at entry). Call again to get the balance of the directory.

A <> 0 with Z set if no error.
HL = free granules.
BC = ?
DE = ?
SP = free to abort reentry. However, proper abort is put FF in OLA, 4220.

Full exit:
A = error code if Z flag not set.
B = number of directory sectors read.
C = ?
HL = free granules.
DE = last RAM byte used +1, else 0 if function does not direct to RAM.

Function:
Bit pattern of B register.

BIT	SET	RESET
0	Place files in RAM	Display onto VIDEO
1	Include I files	
2	Include S files – requires bit 1 set	
3-5	Bits 3 - 5 are the lower protect area in lines + 3	
6	override bits 0-5, and return with free granules in HL.	
7	Return with error in A register.	Display error at current cursor position.

The A register contains the error, regardless of bit 7.

When the directory information is directed to RAM, the format is: 8 bytes disk name, 8 bytes disk date, then 10 bytes for each file - padded to the right with spaces. If RAM area is below (TOPMM), then usage is limited to (TOPMM). See sample program 1.

441C [FILK] Entry DE => FCB (20 bytes), HL => filespec. FILK transfers the filespec from (HL) to (DE). Case conversion is in accordance to the bits in SM0. A = error.

File Control Block

A file control block, FCB, is 20 contiguous bytes of RAM designated by the user. Before open, the FCB has the filespec left justified, terminated with 0D and/or 03. After open the FCB is defined as follows:

FCB+00: BIT 7 Set to indicate an open file.
BIT 6 Set by POSN (i.e., RANDOM I/O in BASIC). Having this bit set retains the EOF on close, unless the file is extended.
BIT 5 Set if the buffer does not contain the current sector.
BIT 4 Set if the buffer contents have been changed.
BITS 3-1 Access password level.
BIT 0 Set forces CLOSE to write a new DIREC. This bit is set whenever a write is made to the file.

FCB+01: reserved for future use

FCB+02: Only used by BASIC during OPEN"O" and/or OPEN"E".

FCB+03,+04: Buffer address for reads and/or writes. Initially set to HL when INIT and/or OPEN is executed.

FCB+05: PNR. Position in Next Record (NRN = FCB+0A, FCB+0B)

FCB+06: The logical drive number (bit 4 indicates the second side of a two-volume drive).

FCB+07: DIREC position code.

FCB+08: PER. Position in Ending Record. (ERN = FCB+0C, FCB+0D)

FCB+09: LRL. Logical Record Length.

FCB+0A,+0B: NRN. Next record number.

FCB+0C,+0D: ERN. Ending record number.

FCB+0E,+0F: A copy of DIREC+16 and DIREC+17 (all numbers are HEX!).

The balance of the FCB - FCB+10 through FCB+1F - contain data in 4 byte clusters.

FCB+10 through FCB+13: First two bytes are the total granules prior, the next byte is the starting cylinder for this extent, and the last byte contains the relative granule (bits 7,6,5) and the number of contiguous granules less one (bits 4,3,2,1,0).

FCB+14 through FCB+17, FCB+18 through FCB+1B, and FCB+1C through FCB+1F are the same as FCB+10 through FCB+13.

Entry Points for MULTIDOS

In the description of following entry points, if an error can occur, is indicated with "A = error" at the end of the description. No error if the Z flag is set.

4420 [INIT] Entry: DE => FCB (with filespec), HL => 100 byte buffer. INIT calls OPEN, 4424, to see if the file exists. If the file exists, control returns to the caller. Otherwise, the file is created with the logical record length set to the contents of the B register, then the file is opened and the C flag is set. A = error.

4424 [OPEN] Entry: DE => FCB (with filespec), HL = > 100 byte buffer. OPEN modifies the FCB containing the filespec to open. The LRL is extracted from the DIREC of the filespec and not from the B register. A = error.

4428 [CLOSE] Entry: DE => FCB (opened). CLOSE completes, if necessary, the last write to the filespec, updates the DIREC and dates the file. Mandatory after a write to a file. A = error.

442B [DOING] This byte contains the nesting level of DO activity.

442C [DLETE] Entry: DE => FCB (opened). DLETE deallocates all granules assigned to the filespec. A = error.

4430 [LOAD] Entry: DE => FCB (filespec). LOAD places the filespec into RAM. A = error.

4433 [LRUN] Entry: DE => FCB (filespec). LRUN calls LOAD, and pushes the entry point. A return is executed if the file has an attribute of EXEC and/or NONE; else, if DEBUG is active, LRUN transfers to DEBUG. A = error.

4436 [READ] Entry: DE => FCB (opened). If the LRL = 0, READ reads a physical record into the contents of FCB+3 and FCB+4. If the LRL <> 0, READ transfers a LRL number of bytes from FCB+3 and FCB+4 to (HL), reading a physical record into FCB+3 and FCB+4 as necessary. A = error.

4439 [WRITE] Entry: DE => FCB (opened). If the LRL = 0, WRITE writes one physical record from the contents of FCB+3 and FCB+4. If the LRL <> 0, WRITE transfers a LRL number of bytes from the (HL) to the contents of FCB+3 and FCB+4, writing a physical record if necessary. A = error.

443C [VERF] Entry: DE => FCB (opened). VERF calls WRITE then rereads the sector for parity. A = error.

443F [POBOF] Entry: DE => FCB (opened). POBOF sets the FCB to the status of a just opened file - position zero. A = error.

4442 [POSN] Entry: DE => FCB (opened). POSN sets the FCB to the logical record in the BC register. A = error.

4445 [BCKSP] Entry: DE => FCB (opened). BCKSP sets the FCB one logical record less than the FCB had. A = error.

4448 [POEOF] Entry: DE => FCB (opened). POEOF sets the FCB at the end of file. POEOF should return the error 1C.

4451 [DIVID] DIVID divides the contents in the HL register by the contents in A register, leaving the quotient in the HL register and the remainder in the A register. If A is zero on entry, HL returns FFFF.

MODEL I:

445C [FIXDT] Required to ensure that the correct FDC chip is selected. i.e., the MODEL I switches density by switching out the 1771 FDC chip instead of selecting density in the FDC chip capable of double-density.

MAX-80 and MODEL 4:

445C [PUNCH] Contains address of the 10 byte table that has the codes returned for <ENTER>, <SHIFT·ENTER>, <CLEAR>, <SHIFT·CLEAR>, <BREAK>, <SHIFT·BREAK>, <↑>, <SHIFT·↑>, <↓>, <SHIFT·↓>, <←>, <SHIFT·←>, <→>, <SHIFT·→>, <SPACE> and <SHIFT·SPACE>.

445E [DSTAT] Returns the status of the logical drive in the C register. The Z flag is reset if no disk, the Z flag is set if there is a disk mounted, and the C flag is set if write protected.

MODEL I, MODEL III, MAX-80, and MODEL 4:

4461 [JKL] Dumps entire VIDEO refresh RAM to the address in the PRDCB+1 and PRDCB+2, converting non-ASCII characters to periods. This is where <J><K><L> goes.

4464 [CEOF] Entry: IX => FCB (opened). CEOF returns with the HL = NRN, C = PNR. If the FCB is positioned at the EOF, A = 1C. If the FCB is positioned beyond the EOF, A = 1D. If the FCB is less than the EOF, A = 0 and Z flag is set.

4467 [VOD] Entry: HL => message. VOD is the general message display routine. VOD outputs a byte to PUTVO, 33, until it encounters a 03, and/or 0D terminator. Only the 0D terminator is sent to PUTVO. The HL register is positioned one byte after the terminator character.

446A [PRT] Entry: HL => message. PRT is similar VOD, except the output is PUTPR, 3B, instead of PUTVO, 33.

446D [GTIME] Entry: HL => 8 byte buffer. GTIME returns the current RAM time in the format hh:mm:ss. On exit, HL is one position after the 8 byte buffer.

4470 [GDATE] Entry: HL => 8 byte buffer. GDATE returns the current RAM date in the format mm/dd/yy. On exit, HL is one position after the 8 byte buffer.

4473 [DEXT] Entry: DE => FCB (filespec). HL => extension. DEXT puts extension in filespec if filespec does not have an extension. This function is also available at 444B.

4476 [PARAM] Entry: DE => parameter list. PARAM interrogates the parameter list and sets the addresses accordingly. A parameter list consists of one to six character phrases, padded to the right with spaces if necessary, followed by a two byte address, and terminated with zero. . This function is also available at 4454.

4479 [DOTIT] The FORMS routine calls this address when the software indicates the form is at the top of page. DOTIT points to a return instruction whenever MULTIDOS is initialized. See sample program 2.

447C [ADRQ] Entry: IY => DCT entry. If floppy and the DCT is not locked, ADRQ loads DCT+0 with the contents of the A register, then sets DCT+6, DCT+7, DCT+8, and DCT+9, accordingly.

[LWAIT] This routine loads the floppy disk command register with the contents of the A register, then waits about 62 micro seconds before returning.

44A1 [FPOS] Entry: IX => FCB. FPOS returns the ERN in the HL register, and the PER in the A register.

MAX-80 and MODEL 4:

44AB [STATP] This routine returns the printer status considering status of ROUTE, spooler, serial output, and/or the parallel port. Z = device ready.

MODEL I, MODEL III, MAX-80, and MODEL 4:

44C9 [MOTON] Entry: C = logical drive number. This jumpss to the routine for floppy disk drive select, and motor turn on.

44CC [GRDUM] Dumps entire VIDEO refresh RAM to the address in the PRDCB+1 and PRDCB+2, including graphic characters. This is where <H><J><K> goes.

44CF [SPRMT] Displays "Insert SYSTEM <ENTER>", and waits for the <ENTER> key to be pressed, then returns. This is similar the ABORT routine without setting SP to STKLD.

[IUS] Insert a task into 1 of 4 slots (numbered 0 through 3). The A register contains the slot number and the DE register contains the address of the routine. Uses HL. When an interrupt task is entered, the AF, BC, DE, HL, and IX register pairs have been saved and the interrupts are disabled - and must remain disabled. The HL register is at the start of the routine upon entry. MULTIDOS and ESOTERIC use slot 0 for special utilities (e.g., Screen Saver), slot 1 for the spooler, slot 2 for type-ahead (not MODEL I), and slot 3 for the clock (not MODEL III). For the MAX-80, MODEL 4 and ESOTERIC, there are 8 slots (numbered 0 through 7).

[DUS] Delete task in slot A. Uses AF, DE, and HL.

44DA [GTDCC] Position IY register to the logical Drive Control Table (DCT) entry for the drive in the C register.

44DB [GTDCT] Position IY register to the logical Drive Control Table (DCT) entry for the drive in the A register.

44DE [READV] Entry: D = cylinder, E = sector, C = logical drive. READV checks the sector for readability. A = error.

44E1 [READS] Entry: D = cylinder, E = sector, C = logical drive, HL = 100 byte buffer. READS transfers the sector data into (HL). A = error.

44E4 [WRITS] Entry: D = cylinder, E = sector, C = logical drive, HL = 100 byte buffer. WRITS transfers the data from (HL) to the sector. A = error.

44E8 [DIRRD] DIRRD issues a READS specifically looking for a DELETED DATA MARK (DDM). DIRRD returns if a DDM sector is read, otherwise DIRRD reads physical cylinder zero, physical sector zero, relative byte two, and updates DCT+3 with this byte (directory cylinder), then reads the new cylinder, sector zero. If the new cylinder, same sector does not return a DDM, then DIRRD tries "P" density1 and "P" density2. If still a DDM isn't found, DIRRD returns an error of 11. A = error.

44EB [DIRWT] Same as WRITS, except issues DDM. A = error.

44EE [RDIRP] Read a DIREC. Entry: B = DIREC position code, C = logical drive number. The DIREC position code is a bit pattern loaded into FCB+7 when a file is opened. (The DIREC code is the only way the system knows where to put any new information. e.g., CLOSE. This is why you should never change diskettes after a file is opened and remains open until CLOSE is called, even if nothing was written to the file. CLOSE simply checks the information in the FCB with the data in the DIREC, if CLOSE finds a difference, CLOSE writes the data in the FCB to the DIREC whose position code is in FCB+7.) Bits 7, 6, and 5 contain the relative offset in the sector. Bits 4, 3, 2, 1, and 0 contain the relative directory file sector. If a directory starts on physical sector 0, then sector 2 is relative directory file sector 0. RDIRP returns with the HL => IOBUF + B's DIREC position code, e.g. B = 110xxxxxB then HL = 43C0. A = error.

44F1 [WDIRP] Write a DIREC. Entry: HL => IOBUF, B = DIREC position code, C = logical drive number, and the information is in IOBUF. WDIRP exits with HL => IOBUF. A = error.

44F4 [USRF] Entry: D = cylinder, E = sector, C = logical drive, HL = buffer, and the A register contains the Floppy Disk Controller code. USRF accepts any of the read sector, write sector, and/or write cylinder commands. A = error.

44F7 [GETDT] Entry: C = logical drive. On exit the D register contains the contents of DCT+3 for the C register DCT.

MAX-80 and MODEL 4:

44FA [FIXRR] This routine sets the cursor to the BC register. B = row, and C = column. On exit the HL has the refresh RAM video (cursor) position. Uses DE, AF. FIXRR is used to position the cursor. e.g., to position the cursor at column 3, row 20d:

```
LD    BC,1403H
CALL  FIXRR
```

This enables your program to easily position the cursor without calculating the position on the second video page and, if a Model 4, setting the second page bit. The execution of this code segment running MULTIDOS on a Model 4 would set the second page bit and return 3E43 in the HL register pair. The execution of this code segment running MULTIDOS on a MAX-80 would return 3043 in the HL register pair. Remember the lowest column is zero and the lowest row is zero.

4500 [DCT] Drive control table Eight DCTs (16 bytes each). Each DCT is defined as follows:

DCT+00: The main control byte containing the configuration information.

Floppy bit pattern:

- BIT 0 Step rate LSB.
- BIT 1 Step rate MSB.
- BIT 2 If set - Double step cylinders with a single step command.
- BIT 3 If set - Perform a 1.8 to 1 division + 1 to read "P" density1.
- BIT 4 If set - Perform a 1.8 to 1 division to read "P" density2.
- BIT 5 If set - Double sided diskettes are treated as two volume.
- BIT 6 If set - 8" floppy, otherwise 3½" and/or 5¼" floppy.
- BIT 7 If set - Double density media, otherwise single density.

Hard disk bit pattern:		Memory DISK bit pattern:	
BIT 0 Step code LSB		N/A	
BIT 1 Step code		N/A	
BIT 2 Step code		N/A	
BIT 3 Step code MSB		SET TO ZERO	0
BIT 4 SET TO ONE	1	SET TO ONE	1
BIT 5 SET TO ZERO	0	SET TO ZERO	0
BIT 6 SET TO ONE	1	SET TO ONE	1
BIT 7 SET TO ZERO	0	SET TO ONE	1

DCT+01: (NIL is indicated by having ALL bits set.)

- BIT 0 Physical drive LSB
- BIT 1 Physical drive nSB
- BIT 2 Physical drive MSB
- BIT 3 MEMDISK
- BIT 4 Set to one indicates HARD/MEMORY. A zero indicates floppy.
- BIT 5 Locks DCT. (No automatic update to DCT+06 through DCT+08 and no automatic density switching)
- BIT 6 Set to one to indicate software write protect.
- BIT 7 Set to zero to indicate drive in system.

DCT+02: Number of heads per volume. 01 to 08.

DCT+03: The directory cylinder. This byte is obtained from cylinder zero, sector zero, relative byte two. 01 to 3F for floppies.

DCT+04: Floppy only: The current cylinder where the head was positioned when this drive was last accessed and another drive is selected. This byte is not where the head is, if this drive is selected. The Disk Controller contains the value where the head is located when this drive is being accessed.

DCT+05: Sector offset. 00 or 01 (used by CAT, CAT/CMD, [V]FU/CMD and ZAP/CMD)

DCT+06: The number of sectors per granule. 01 to 0000 (00 = 256)

DCT+07: The number of granules per cylinder. 01 to 08

DCT+08: The number of sectors per head. 01 to 00 (00 = 256)

DCT+09: The number of directory files sectors. 01 to 20

DCT+0A: Reserved for system use (direction for allocation: 00 = ignore DCT+0B, or FE = allocate down from DCT+0B [in contiguous granules, if possible], or FF = allocate up from DCT+0B [not necessarily contiguous]).

DCT+0B: Reserved for system use (starting cylinder for allocation 00 to FD).

DCT+0C: The most significant byte for the cylinder offset. 00 to FF

DCT+0D: The least significant byte for the cylinder offset. 00 to FF

DCT+0E: The head offset. 00 to 07

DCT+0F: The number of cylinders for hard disks (Floppies updated by CAT/CMD).

When the configuration is floppy, DCT+06 through DCT+08 are updated to these values unless bit 5 is set in DCT+01: (SS = single-sided, DS = double-sided, SD = single-density, DD = double-density)

	DCT+06	DCT+07	DCT+08	DCT+09
SS/SD	05	02	0A	08
DS/SD	05	04	0A	12
SS/DD	06	03	12	10
DS/DD	06	06	12	20

You can establish other values to read a special format pattern by placing the respective values in the DCT and set bit 5 of DCT+01. e.g., NEWDOS/80 PDRIVE setting with TC=80, SPT=36, GPL=8, and DDGA=6.

DCT+00 = 100110xxB (98H = 6mS, 99H = 12mS, etc. xx = stepping rate)
DCT+02 = 02H
DCT+06 = 05H ("....there are still 5 sectors per granule.")
DCT+07 = 08H (granules per LUMP)
DCT+08 = 28H (sectors per LUMP)
DCT+09 = 1CH (DDGA * 5 - 2, 5 = DCT+06, 2 = 1 GAT + 1 HIT)

4D00 to 51FF Overlay area. Reference ESOTERIC 1600 to 1AFF. All /DOL and /BOL files load into this area.

5200 to (TOPMM) Application area 1. Reference ESOTERIC 4A00 to (TOPMM).

Sample Programs

Sample program 1 – Obtain directory on specified drive (default zero) protecting the bottom six lines.

```
00001 KWAIT   EQU 0049H
00002 ERROR   EQU 4409H
00003 CAT     EQU 4419H
00004         ORG   5200H
00005 START   LD    C,0             ;default drive zero
00006         DEC   HL              ;compensate for RST 10H action
00007         RST   10H             ;fetch next character
00008         JR    C,HDRV          ;"C" if numeric – Has DRiVe
00009         JR    NZ,NODRV        ;"NZ" if not colon (default drive zero)
00010         RST   10H             ;bypass colon
00011 HDRV    AND   7               ;strip ASCII
00012         LD    C,A             ;put drive number into C register
00013         RST   10H             ;fetch next character
00014 NODRV   CP    27H             ;is it an apostrophe (ancient two volume)
00015         JR    NZ,ONEVOL       ;"NZ" if single volume drive
00016         SET   4,C             ;indicate 'prime' side
00017 ONEVOL  LD    B,18H           ;value to protect six lines
00018 DOMORE  CALL  CAT             ;execute code in Minidos/DOL
00019         JP    NZ,ERROR        ;"NZ" if error – display error
00020         OR    A               ;did we display all files?
00021         RET   Z               ;"Z" if all files displayed
00022         CALL  KWAIT           ;wait for key press
00023         JR    DOMORE          ;loop until all files displayed (or error)
00024         END   START
```

Sample program 2 - Create a title (header) for printouts via DOTIT, 4479

```
00001 TODOS   EQU   402DH
00002 TOPMM   EQU   4411H
00003 DOTIT   EQU   4479H
00004 PRT     EQU   446AH
00005         ORG   0FC00H
00006 START   LD    HL,MYCDE
00007         LD    A,0C3H
00008         LD    (DOTIT),A
00009         LD    (DOTIT+1),HL
00010         DEC   HL
00011         LD    (TOPMM),HL      ;protects routine
00012         JP    TODOS
00013 MYCDE   EXX                   ;MULTIDOS resident doesn't use alt regs.
00014         LD    HL,MYTIT
00015         CALL  PRT             ;can call itself (safe because of EXX)
00016         OR    A               ;must return a non zero
00017         EXX                   ;put 'em back
00018         RET                   ;back to caller
00019 MYTIT   DEFM  'THIS IS MY WORK',0DH
00020         END   START
```

0000 [SUPV] CLS, and FUNCTION KEYS

INPUT A	INPUT B	AF on exit	FUNCTION
0	n/a	1F	Clear display
1	n/a	1F	Clear display
2	n/a	1F	Clear display
3	0	B reg	Set F1 through F6 to their default values
3	1	B reg	Set F1 to routine @ DE and VD (see below)
3	2	B reg	Set F2 to routine @ DE and VD
3	3	B reg	Set F3 to routine @ DE and VD
3	4	B reg	Set F4 <RIGHT-SHIFT·F1> to routine @ DE and VD
3	5	B reg	Set F5 <RIGHT-SHIFT·F1> to routine @ DE and VD
3	6	B reg	Set F6 <RIGHT-SHIFT·F3> to routine @ DE and VD
3	>6	B reg	VD: Value (DATA) with F-key jump -2
4	n/a	00	perform <LEFT-SHIFT-F1>
5	n/a	00	perform <LEFT-SHIFT-F2>
6	n/a	00	perform <LEFT-SHIFT-F3>
>6	n/a	A-6	no function (ignored)

With a function key routine set, the KEYBOARD scan routine executes the function, and then returns with whatever is in the A register. If a key value is also desired, then LD A with the character just before the last RET instruction. If no key value is desired, then execute a XOR A just before the last RET instruction. Function key commands/characters do not repeat.

0003 [PUTPR] Outputs character in A register to the device in the printer DCB. Returns with printed character in A and the Z flag set.
0008 Not used in DOS. BASIC normal RST 08.
0010 Jumps to PARSE under DOS. BASIC normal RST 10.
0013 [PUTVO] Puts character in A register onto video display. Returns with displayed character in A and the Z flag set.
0018 Compare HL to DE using only AF.
001B [GETKI] Scans keyboard once, returns 0 if no key, otherwise returns with ASCII value of key pressed in A register.
0020 Not used by DOS. BASIC normal RST 20.
0023 [PUTSO] Outputs character in A register to the RS-232-C. Returns with charcter set in A and the Z flag set.
0028 Load overlay without returning to address after RST 28. For the MAX-80, MODEL 4, and ESOTERIC pressing the <BREAK> key does not go here!
002B [GETSI] Get character from RS-232-C into A.
0030 Call to load/execute DEBUG.
0033 [PUTBT] Put byte in A register to device or file: DE => DCB or FCB (opened).
0038 Used for Interrupt Mode 1.
003B [GETBT] Get file byte into A register: DE => FCB (opened).

0050 to 0055 Storage for seconds, minutes, hours, year, day, and month

005E [KIM] Points to the terminator byte after executing a DOS function.

0060 [DBUF] Points to the DOS command buffer.

0062 [TOPMM] This word contains the address of the highest available memory location for DOS. BASIC temporarily sets this word to a different value during CMD"fffff" execution.

0064 [BREAK] BREAK KEY status. If non-zero, the KEYBOARD returns a zero when <BREAK> is pressed. The DOS command mode sets this byte to zero.

0069 [STATP] This routine returns the printer status considering status of ROUTE, spooler, serial output, and/or the parallel port. Z = device ready.

0850 [DELAY] Delay for interval. Uses BC for delay value. Exit: BC =0, A =0, “Z” set. Some estimated delays (in microseconds)

BC	ESOTERIC
0000	1,034,349d
F780	**1,000,005d**
7BC0	**500,005d**

0880 [STKEY] Jumps to special first position keystroke when 40 is called. The address of the routine is loaded into 0881. After performing some test on the keystroke, if it is not what you want simply return. Otherwise, you will have to remove (at least) two calls from the stack similar to this form:

```
POP QQ  ;call to 0881
POP HL  ;restore HL register
POP QQ  ;call to 08B8
```

08A1 [FIXRR] This routine sets the cursor to the BC register. B = row, and C = column. On exit the HL has the refresh RAM video (cursor) position. Uses DE, AF. FIXRR is used to position the cursor. e.g., to position the cursor at column 3, row 20d:

```
LD    BC,1403H
CALL  FIXRR
```

The execution of this code segment running ESOTERIC on a Model 4 would return FE43 in the HL register pair.

Remember the lowest column is zero and the lowest row is zero.

08B1 [KWAIT] Calls 2B until a key is pressed.

08B8 [LINE] Input into (HL) B characters.

08C1 [WHERE] Call to locate PC register. i.e., LD HL,PC. This is the same as 000B in the ROM.

08C3 [PARSE] Increments the HL register pair, interrogates the byte in HL, skips spaces and returns with "C" set if numeric or "Z" set if 3A and/or 0.

120A [TIC] This bytes increases by one for each real time interrupt.

120B [KBDCB] These 6 bytes are defined as the KEYBOARD Device Control Block (DCB).

DCB+0	This byte normally contains 1
DCB+1,2	Driver address
DCB+3,4	Driver address storage during route
DCB+5	

BIT 0	Used by printer routine	SET = hard form feed
1	Used by printer routine	SET = software vertical tab
2	Space compression/Special	SET = special characters
3	lower case/CAP lock	SET = lower case
4	key beep	SET = key beep
5	blinking cursor	SET = blink
6	graphics	<LEFT-SHIFT·F2> sets this bit.
7	high bit set	<LEFT-SHIFT·F3> sets this bit.

1211 [DOING] This byte contains the nesting level of DO activity.

1212 [JCL1] Holds the nesting level of a DO file when input is required.

1213 [MEMTP] This word contains TOPMEM prior to executing DO files.

1215 [DOBUF] This word is the pointer to current DO File Control Block.

1217 [VODCB] These 10 bytes are defined as the VIDEO Device Control Block (DCB).

DCB+0	This byte normally contains 2
DCB+1,2	Driver address
DCB+3,4	Refresh RAM position
DCB+5	Character storage if cursor is on
DCB+6	VIDEO column position
DCB+7	VIDEO row position
DCB+8	VIDEO cursor character
DCB+9	VIDEO scroll protect. The number of lines protected is determined by the highest bit set. The number of lines = N+1 where bit N is set. i.e., set bit 4 to protect 5 lines (protects 1 to 8 lines).

1221 [CEOFC] Stores character for write to end of VIDEO when a 12 is sent to VIDEO routine.

1222 [PRDCB] These 9 bytes are defined as the PRINTER Device Control Block (DCB).

DCB+0	This byte normally contains a 2
DCB+1,2	Driver address
DCB+3	Maximum number of printed lines per page
DCB+4	Line counter
DCB+5	Maximum number of characters per line
DCB+6	Character counter
DCB+7	The number of non-printed lines per page
DCB+8	Bits 7-3 contain the left margin, bits 2-0 contain the number of NULLS sent after a linefeed.

122B [SIDCB] These 3 bytes are defined as the RS-232-C input device control block.

122E [SODCB] These 5 bytes are defined as the RS-232-C output device control block.

1500 [IOBUF] General purpose 100 byte I/O buffer. Reference MODEL I and MAX-80 as 4200; and, MODEL III and MODEL 4 as 4300.

1600 to 1AFF Overlay area. Reference MULTIDOS 4D00 to 51FF.

1B00 to 41FF Application area 3. if this area is used and invoked from ESOBASIC (always a possibility), then application must abort ESOBASIC. Reference MODEL 4 MULTIDOS 0900 – 2FFF.

Sample code segment to use application areas 3, 2, and 1 (needed for area 3)

```
00001 SBSPA EQU   1280H          ;ESOTERIC, for MAX-80 and MODEL 4 = 4080H
00002 CMDTP EQU   12C0H          ;ESOTERIC, for MAX-80 and MODEL 4 = 40C2H
00003 NEST  EQU   12E7H          ;ESOTERIC, for MAX-80 and MODEL 4 = 411BH
00004 START LD    A,(TODOS+2)    ;get exit MSB
00005       CP    DOS<-8         ;is it the DOS entry point?
00006       JR    Z,star2        ;"Z" if executed from DOS
00007       PUSH  HL             ;save command parser
00008       LD    HL,MSGBA       ;locate HL to message
00009       CALL  VOD            ;display message
00010       LD    A,(DOING)      ;get 'DO' status
00011       OR    A              ;is 'DO' active?
00012       CALL  Z,KWAIT        ;wait for key press if 'DO' inactive
00013       LD    HL,DOS         ;DOS entry point
00014       LD    (TODOS+1),HL   ;value exit point with DOS entry
00015       LD    A,(NEST)       ;get BASIC's nesting level
00016       LD    HL,(CMDTP)     ;get TOPMEM prior to CMD"fffff"
00017 star0 DEC   A              ;reduce nesting level: is it zero?
00018       JR    Z,star1        ;"Z" if unnested
00019       LD    DE,CMDTP-SBSPA+1  ;get difference between start and ...
00020                            ;where nested TOPMEM is stored
00021       ADD   HL,DE          ;compute where nested TOPMEM is
00022       LD    E,(HL)         ;get nested TOPMEM LSB
00023       INC   HL             ;to where nested TOPMEM MSB is
00024       LD    D,(HL)         ;get nested TOPMEM MSB
00025       EX    DE,HL          ;value HL with nested TOPMEM
00026       JR    star0          ;loop until unnested
00027
00028 MSGBA DEFM  28,31,'BASIC aborted!',10,10
00029       DEFM  'Press any key to continue.',3
00030 star1 LD    (TOPMM),HL     ;update TOPMEM
00031       LD    (MEMTP),HL     ;update MEMTOP in case invoked by 'DO'
00032       POP   HL             ;retrieve command parser
00033 star2                      ;where application starts
```

4200 to 49FF Application area 2. If this area is used, application must set bit 4 in SM1 - 124A. Reference MODEL 4 MULTIDOS 3000 to 37FF.

Sample code segment to use application area 2 and 1 (needed for area 2)

```
00001       PUSH  HL             ;save parser
00002       LD    HL,SM1         ;position HL to system attributes
00003       SET   4,(HL)         ;indicate application area 2 used
00004       POP   HL             ;retrieve parser
```

4A00 to (TOPMM) Application area 1. Reference MULTIDOS 5200 to (TOPMM).

GLOSSARY

access - a means of involving a file in a read or write operation.

address - a location in memory specified by a two-byte hexadecimal address.

alphabetic - consisting of only the letters A through Z and/or a through z.

alphanumeric - consisting of the numerals 0 through 9, and the letters A through Z and/or a through z.

argument - an input to a function used to derive a value.

array - an organized set of elements that is referenced by the array name and one or more subscripts.

ASCII - American Standard Code for Information Interchange for values 0 through 127.

ASCII file - a file that is an exact duplicate of the user input data, readable by LISTing the file.

background task - a task performed by the microprocessor during an interrupt. Examples of background tasks are CLOCK, the SPOOLer, and TYPE-ahead.

backup - a necessary process required to maintain your important data.

BASIC - Beginner's All-purpose Symbolic Instruction Code.

baud - the rate of serial data transfer in bits per second.

binary – a base two number represented by the digits 0 and 1.

bit - one binary digit; either a 0 or a 1.

boot - the process of reloading the executive program in RAM.

break - to interrupt a program or Library command.

buffer - an area in RAM where data is held for processing.

byte - the smallest unit of reading/writing from/to RAM.

close - to terminate access to a file.

configuration - The access state of the disk drives.

cylinder - a track location of the disk drive's read/write head.

DAM - Data address mark for floppy diskettes. The DAM for the directory cylinder is different than the DAM for file cylinders.

DCB - Device Control Block: a small contiguous area in RAM used for the status and/or control of a device.

DCT - Drive Control Table: a small contiguous area in RAM that has the drive's configuration information.

GLOSSARY

decimal – a base ten number represented by the digits 0 through 9.

default - an action and/or value that is supplied by the executing program when you only press <ENTER> to a prompt or query. Or an action and/or value supplied by the program if you take no action.

delimiter - a character that marks the beginning and end of data, and is not part of the data. e.g., the double quotation marks surrounding a string value

density - the recording method used to write/read to/from a floppy diskette.

destination - the device or address that receives data during a data transfer process.

device - a physical piece of hardware such as the video display, a printer, a disk drive, a RS-232 interface, etc.

directory - a file on every disk that contains all of the information about other files.

disk – mass storage magnetic media that can be a floppy disk or a hard disk.

diskette - mass storage magnetic media that is a floppy disk

DO - the video Display Output.

drivespec - a particular logical disk drive number (0 to 7).

:d - a colon followed by an integer number from 0 through 7' to indicate a file's drivespec.

edit - a process used to change existing information.

EOF - End Of File.

entry point - the address of a machine-language program where execution is to begin. Also referred to as the transfer address.

Exp - the base 2 exponent used in single-precision and double-precision numbers.The exponent is offset by 129.

expression - a variable or a logical sequence of variables, operators and/or functions.

/ext - an optional extension of a filespec. Extensions are delimited by a preceding slash.

FCB - File control block: a small contiguous area in RAM used for the status and/or control between the DOS and a disk file.

file - a collection of unique data.

filename - the name of a file.

filespec - the name of a file with up to three additional options: the extension, the password, and the drivespec.

filter - a program, positioned between a device an the computer, that can alter the data as it passes through the filter.

foreground task - jobs performed by the microprocessor that you control.

format - the process of initializing a disk with track and sector identification marks (ID) and creating a new directory on the disk.

gran/granule - the smallest file allocation unit.

hexadecimal/hex - a base 16 number represented by the numerals 0 through 9 and the letters A through F.

input - the transference of data from you an/or a device to the computer.

interrupt - a microprocessor signal that causes the software to suspend a foreground task in order to perform a background task. The background task usually returns control back to the foreground task.

I/O - Input/Output.

KI - Keyboard Input.

kilobyte - 1024 bytes, i.e., 64 kilobytes or 64K = 1024 * 64 = 65,536 bytes.

logical record - contiguous bytes of data that is accessed a one unit of a disk's file. MULTIDOS supports logical record lengths of 1 through 256.

Library commands - a set of commonly used commands.

LSB - Least Significant Byte.

machine-language - data consisting of the microprocessor's instruction set.

modulo - the domain of remainders from integer division by a modulus. e.g., a modulus of eight has eight possible remainders: 0 through 7, referred to as modulo 8.

MSB - Most Significant Byte.

nil/null - inactive and/or missing value. Null does not imply zero.

NRN - Next Record Number.

nSB - neither LSB nor MSB Not Most Significant Byte. Single-precision numbers have one nSB, and double-precision numbers have five nSBs.

Single-precision numbers are stored in this format:

```
LSB nSB MSB Exp
```

Double-precision numbers are stored in this format:

```
LSB nSB nSB nSB nSB nSB MSB Exp
```

object code - meaningful machine-language data or file. All files executed by the computer must be in machine-language.

open - the process of accessing a file.

output - the transference of data from the computer to a device.

parameter - an argument that you supply in a command line. Parameters are enclosed in parentheses and are usually optional.

.password - the optional password assigned to a file, or the mandatory password required to access a file. Passwords are delimited by a preceding period.

physical record - a disk file's sector

RAM - Random Access Memory. RAM will lose data when power is removed.

reset - the process of returning something to it's initial state. Reset also means to boot the computer.

ROM - Read Only Memory. ROM maintains information when power is removed.

RS-232 - the Electronic Industries Association's (EIA) standard for communication between two computers.

sector - the smallest unit used to read/write to a disk. MULTIDOS uses 256 contiguous bytes for each physical record.

SI - Standard Input or Serial Input.

SO - Standard Output or Serial Output.

track - a recording surface on a disk.

transfer address - the address of a machine-language program where execution is to begin. Also referred to as the entry point.

utility - a program that serves specific purpose. Utilities are much larger than Library commands and may use all of RAM.

word - two adjacent bytes.

INDEX

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

******** ZAP 001 ******** 12/27/84 ******** - SYSTEM (optional)

High speed modification. Most software high speed modifications for the Model I require an output to an I/O port. Due to size constraints, none of the library commands have a facility to accomplish this task. However, MULTIDOS will OUT the byte in relative position F-H to the PORT in relative byte F1H located on track zero sector one.

```
DM   F0 0CFE 0000 0504 0000 0000 0000 0000 0000 ................
              (partial track 0, sector 1)
```

The above partial ZAP screen of track zero, sector one indicates a 12 will be output to port 254 during initialization.

******** ZAP 002 ******** 12/27/84 ******** - SYSTEM (optional)

Disk I/O tries. MULTIDOS is distributed with the number of disk I/O tries set to 5. This is a good value. If a diskette starts to give you errors, BACKUP the file(s) immediately. Increasing the number of I/O tries may enable you to read the file; however, eventually it will fail. The number of I/O tries is in the lower nibble (bits 0 to 3) of relative byte F4H (maximum number of tries is 15). If you want to change the I/O tries, modify the byte in relative position F4H on track zero sector one.

******** ZAP 003 ******** 12/27/84 ******** - SYSTEM (optional)

Head load delay. Whenever MULTIDOS selects a different drive, even if the motors are up to speed, a delay of about 15mS occurs before an attempt is made to read/write to the newly selected drive. The amount of delay is increased 3.756mS for each unit in relative byte F5H on track zero sector one.

******** ZAP 004 ******** 12/27/84 ******** - Open/DOL (optional)
******** ZAP 004 updated 01/01/86 ********
******** ZAP 004 updated 11/30/90 ********
******** ZAP 004 DELETED 07/07/98 ********

Open/DOL error codes. The error code most application programs expect when an attempt is made to access a file, and the "file is not found", is 18H. This error can occur in three distinct circumstances. 1) The drivespec given does not contain the file, 2) the drivespec given is not available, or 3) the file was searched for and not found on all mounted drives. Error code 18H is correct for case 1, error code 08H is correct for case 2, and error code 1FH is correct for case 3. MULTIDOS can return the correct error code by zapping two bytes in Open/DOL. Please note, with this zap, some application programs might return "Program not found" if an access is made to a file without a drivespec.

File: Open/DOL, Relative sector: 4, relative byte: 83H

```
Change:   1818
    To:   081F
```

******** ZAP 005 ******** 12/27/84 ******** - SYSTEM (optional)

Default value for SM0. If any of the high three bits in SM0 are set, you may have unusual filenames. However, keep in mind not all operating systems recognize the new syntax. Relative byte F7H on track zero sector one is loaded into SM0 during initialization. Please refer to the discussion of the bit definitions for SM0 in the Entry points for MULTIDOS section.

DM F0 0CFE 0000 0504 00**00** 0000 0000 0000 0000 ....
(partial track 0, sector 1)

******** ZAP 006 ******** 12/27/84 ******** - ZAP/CMD (optional)

Default response for ZAP/CMD menu. ZAP/CMD's first sector in relative byte 80H, contains the menu default character. This is distributed with 43H, C, "Copy sectors". The value here is ZAP's default when you press <ENTER> at the "Choice" prompt. Zap 44H if you want the default to be "Disk sectors", 46H if you want the default to be "File sectors", etc. The new default does not go into effect until ZAP is initialized again.

******** ZAP 007 ******** 12/27/84 ******** - BASIC/CMD, BBASIC/CMD (optional)

Default number and type of file buffer areas for SUPERBASIC. The first sector of BASIC/CMD and BBASIC/CMD in relative byte 84H contains the default number of file buffer areas allocated during SUPERBASIC initialization. Valid values are 00 through 0F. And relative byte 85H determines the type of buffer areas opened during initialization. If this byte is non-zero then the required "V" for user defined record lengths during random I/O is not required upon initialization, e.g., BASIC 3V<ENTER> if the byte is zero, or BASIC<ENTER> if the byte is non-zero. The "V" buffer areas are 545 bytes each, and non-"V" buffer areas are 289 bytes each.

******** ZAP 008 ******** 01/24/85 ******** - BACKUP/CMD (optional)
******** ZAP 008 updated 01/01/86 ********
******** ZAP 008 updated 10/22/90 ********
******** ZAP 008 updated 06/15/99 ********

The automatic feature of verifying a diskette after all granules are duplicated onto the destination diskette can be modified to: (1) never verify, or (2) to verify only if the DOS verify is on.

File: BACKUP/CMD, Relative sector: 9, relative byte: E0H

NEVER RE-VERIFY	PER LIBRARY COMMAND VERIFY
Change: 44**AF 2F**32	Change: 44**AF 2F**32
To: 44**AF AF**32	To: 44**96 2F**32

******** ZAP 009 ******** 02/06/85 ******** - VisiCalc VC/CMD (interface)

VisiCalc print enable. Visicalc uses the interrupt task processor to print. Changes to the way the interrupt task processor operates with MULTIDOS version 1.7 and later doesn't support VisiCalc printing. The zapped Visicalc will only work with MULTIDOS.

File: VC/CMD, relative sector 75, relative byte 16H

Change: 289C
    To: EE9B

******** ZAP 010 ******** 02/11/85 ******** - INFO (informational)
******** ZAP 010 updated 12/30/85 ********
******** ZAP 010 updated 03/29/89 ********

CAT/CMD dated 10/22/88 or later automatically turns SKIP on or off, and CONFIGs for side and/or volumes with or without the "N" parameter. NOTE: The "N" parameter was changed to "T" in June 1999.

This ZAP is for transmitting information on the release of Version 1.71. It is impossible to ZAP all of the modules involved in reading NeWDOS/80 diskettes with the GPL parameter not equal to 2. The original release of Version 1.7 will only read a NeWDOS/80 diskette with a GPL=2. SYSRES/SYS has been updated to handle extended directories on NeWDOS/80 version 2 diskettes. See DCT+01 bit 5 definition.

DCT+01: (NIL is indicated by having ALL bits set.)

BIT 0 Physical drive LSB.
BIT 1 Physical drive.
BIT 2 Physical drive MSB.
BIT 3 MEMDISK.
BIT 4 Set to one indicates HARD/MEMORY. A zero indicates floppy.
BIT 5 Locks DCT. (no automatic update to DCT+06 through DCT+08)
BIT 6 Set to one to indicate software write protect.
BIT 7 Set to zero to indicate drive in system.

For floppies, DCT+06 through DCT+08 are updated each time the DOS must invoke automatic data recognition. The default decimal values are:

	DCT+06	DCT+07	DCT+08	DCT+09
3½", 5¼" SD/SS	5	2	10	8
3½", 5¼" SD/DS	5	4	10	18
3½", 5¼" DD/SS	6	3	18	16
3½", 5¼" DD/DS	6	6	18	32

You can establish other values to read a special format pattern by placing the respective values in the DCT and set bit 5 of DCT+01.

EXAMPLE: NeWDOS/80 PDRIVE setting with SPT=36, GPL=8, and DDGA=6.

```
    DCT+00 = 100110xxB (xx is the stepping rate) 98H = 6mS
    DCT+02 = 02H
    DCT+06 = 05H ("...there are still 5 sectors per granule,...")
    DCT+07 = 08H (GPL)
    DCT+08 = 28H (sectors per LUMP)
    DCT+09 = 1CH (DDGA * 5 - 2, 5 = DCT+06, 2 = 1 GAT + 1 HIT)
```

The file CAT/CMD has been modified to set bit 5 of DCT+01. Bit 5 of DCT+01 will be reset if the new "N" or "L" parameter is specified (while reading a MULTIDOS diskette). In addition to the "N" and "L", CAT/CMD has a "W" parameter to write, onto a NeWDOS/80 diskette's GAT sector, the necessary two bytes in relative bytes 0CCH and 0CDH to ensure the entire NeWDOS/80 disk space can be used. NeWDOS/80 doesn't care that these two bytes are set to a non-zero value. If the "W" parameter is used to update these two bytes, the affected disk can function normally in a NeWDOS/80 environment. Use the "W" only on NeWDOS/80 diskettes!

June 1999 update: The "N" parameter has been changed to "T", the "W" parameter has been changed to "Z", and the "L" parameter eliminated.

The files Allocate/DOL, COPY/CMD, and VFU/CMD has been modified to limit the tracks accessed if the bytes at 0CCH and 0CDH are zero. This enables MULTIDOS to write to a NeWDOS/80 diskette without trying to write to a non-existent track. However, MULTIDOS probably will indicate a full diskette if these two bytes are zero.

The file CDIR/CMD will clear the entire directory, provided the diskette was accessed via CAT/CMD prior to CDIR.

The file FMAP/CMD will display the correct information, provided the diskette was accessed via CAT/CMD prior to FMAP.

The library command DIR (Library1/EXT), and Minidos will display correct NeWDOS/80 directory data, provided the diskette was accessed via CAT/CMD prior to these commands.

The file ZAP/CMD will not complain "Garbage in HIT sector." for extended NeWDOS/80 directories if the diskette was accessed via CAT/CMD prior to the "X" function in ZAP. In addition, ZAP can now display all files on a NeWDOS/80 disk, with a GPL greater than two.

To ensure correct reading of a NeWDOS/80 diskette, issue a CAT :d command prior to reading the NeWDOS/80 diskette. CONFIG (Library2/EXT) will indicate a locked DCT. Subsequent CATs to a different diskette in the drive that had the NeWDOS/80 diskette will automatically lock and/or unlock the DCT.

Of course, all of this depends on correct PDRIVE settings.

******** ZAP 011 ******** 02/17/86 ******** - BACKUP, FORMAT, and ZAP/CMD (optional)

Faster disk I/O by using a different interleave with double-density diskettes. Interleave allows logical contiguous sectors of data on a given track to be mapped onto non-adjacent physical sectors. An interleave of two (single-density) means that every second physical sector is transferred as the next contiguous logical sector of data. An interleave of three (double-density) means that every third physical sector is transferred as the next contiguous logical sector of data. An interleave of six (Model 3 TRSDOS®) means that every sixth physical sector is transferred as the next contiguous logical sector of data. This also means the number of sectors per track divided by the interleave is transferred in one revolution of the diskette. If the diskette is single-density with an interleave of two and 10 sectors per track, then each revolution of the diskette transfers five sectors requiring two revolutions of the diskette to transfer one track of data. If the diskette is double-density with an interleave of three and 18 sectors per track, then each revolution of the diskette transfers six sectors requiring three revolutions of the diskette to transfer one track of data. If the diskette is Model 3 TRSDOS® with an interleave of six and 18 sectors per track, then each revolution of the diskette transfers three sectors requiring six revolutions of the diskette to transfer one track of data.

The time the system requires to transfer one sector of data is the primary factor in determining the interleave. If a read operation is in progress and the interleave is three, after a sector of data is read into the sector buffer, the sector buffer is processed while the read/write head is passing over the next two sectors. If the system cannot process the sector buffer during the two-sector time available, then the system has to wait a full revolution before the next logical sector can be read from the diskette. For example, Model 3 TRSDOS® cannot process a sector buffer in a two-sector time; therefore, Model 3 TRSDOS® uses an interleave of six. Without using a calculator, you can see why Model 3 TRSDOS® requires 1.2 seconds —— 6 times 200mS (time required for one revolution of the diskette) —— to read a full track of data.

An interleave of three was used with the introduction of DBLDOS™ because the Model I has a standard CPU speed of 1.77 MHz. The other DOS makers also used an interleave of three.

The interleave for double-density diskettes can be reduced to two because the Model 4/4P/4D has a standard CPU speed of 4 MHz. In fact, a CPU speed greater than 2.5 MHz can handle an interleave of two if the operating system is MULTIDOS. An interleave of two is 33 percent faster than a interleave of three —— 600mS vs 400mS.

If you have a high speed modification, I recommend you ZAP a byte in BACKUP/CMD, FORMAT/CMD, and ZAP/CMD to incorporate an interleave of two for double-density diskettes. The interleave for single-density diskettes remains at two. In all cases a value of zero is an interleave of three, and a non-zero value is an interleave of two.

File: BACKUP/CMD, Relative sector: 0, relative byte: A0H
FORMAT/CMD, Relative sector: 0, relative byte: B0H
ZAP/CMD, Relative sector: 0, relative byte: 81H

******** ZAP 001 ******** 12/27/84 ******** - SYSTEM (optional)
******** ZAP 001 updated 06/27/91 ********
******** ZAP 001 updated 11/03/99 ********

MULTIDOS sets bit 6 of 4210H and outputs a one to port 95 during initialization. The code to accomplish this is on track zero sector one from relative byte 72H through relative byte 7FH. Bytes 75H and 76H set a Model 4 to 4 MHz running in the Model III mode. If high speed is not desired, then zero bytes 75H and 76H. Bytes 7CH through 7FH sets the HOLMES board to high speed. If another byte/port combination is desired, then update the appropriate code into bytes 7CH through 7FH.

```
              (partial track 0, sector 1)
Cyl  70 1144 2110 42CB F6CB E67E D3EC 3E01 D35F .D!.B....~..>.._

  LD    HL,4210H
  SET   6,(HL)
  SET   4,(HL)

                                  LD    A,1
                                  OUT   (95),A
```

******** ZAP 002 ******** 12/27/84 ******** - SYSTEM (optional)
******** ZAP 002 updated 12/30/85 ********

Disk I/O tries and side compare. MULTIDOS is distributed with the number of disk I/O tries set to 5. This is a good value. If a diskette starts to give you errors, BACKUP the file(s) immediately. Increasing the number of I/O tries may enable you to read the file; however, eventually it will fail. The number of I/O tries is in the lower nibble (bits 0 to 3) of relative byte F4H (maximum number of tries is 15). The higher nibble uses bit 7 to enable side compare for double-sided diskettes. If this bit is set, side compare is enabled. If this bit is reset, then side compare is disabled. MULTIDOS **versions earlier than 2.00 did not have side compare enabled.** If you want to change the I/O tries and/or side compare, modify the byte in relative position F4H on track zero sector zero.

```
DM   F0 0000 0000 8504 0000 0000 0000 0000 0000 ................
              (partial track 0, sector 0)
```

******** ZAP 003 ******** 12/27/84 ******** - SYSTEM (optional)

Head load delay. Whenever MULTIDOS selects a different drive, even if the motors are up to speed, a delay of about 15mS occurs before an attempt is made to read/write to the newly selected drive. The amount of delay is 3.79mS for each unit in relative byte F5H on track zero sector zero.

```
DM   F0 0000 0000 8504 0000 0000 0000 0000 0000 ................
              (partial track 0, sector 0)
```

******** ZAP 004 ******** 12/27/84 ******** - Open/DOL (optional)
******** ZAP 004 updated 01/01/86 ********
******** ZAP 004 updated 11/30/90 ********
******** ZAP 004 DELETED 07/07/98 ********

Open/DOL error codes. The error code most application programs expect when an attempt is made to access a file, and the "file is not found", is 18H. This error can occur in three distinct circumstances. 1) The drivespec given does not contain the file, 2) the drivespec given is not available, or 3) the file was searched for and not found on all mounted drives. Error code 18H is correct for case 1, error code 08H is correct for case 2, and error code 1FH is correct for case 3. MULTIDOS can return the correct error code by zapping two bytes in Open/DOL. Please note, with this zap, some application programs might return "Program not found" if an access is made to a file without a drivespec.

File: Open/DOL, Relative sector: 4, relative byte: 83H

Change: 1818
    To: 081F

******** ZAP 005 ******** 12/27/84 ******** - SYSTEM (optional)

Default value for SM0. If any of the high three bits in SM0 are set, you may have unusual filenames. However, keep in mind not all operating systems recognize the new syntax. Relative byte F7H on track zero sector zero is loaded into SM0 during initialization. Please refer to the discussion of the bit definitions for SM0 in the Entry points for MULTIDOS section.

DM   F0 0000 0000 8504 00**00** 0000 0000 0000 0000 ....
             (partial track 0, sector 0)

******** ZAP 006 ******** 12/27/84 ******** - ZAP/CMD (optional)

Default response for ZAP/CMD menu. ZAP/CMD's first sector in relative byte 80H, contains the menu default character. This is distributed with 43H, C, "Copy sectors". The value here is ZAP's default when you press <ENTER> at the "Choice" prompt. The new default does not go into effect until ZAP is initialized again.

******** ZAP 007 ******** 12/27/84 ******** - BASIC/CMD, BBASIC/CMD (optional)

Default number and type of file buffer areas for SUPERBASIC. The first sector of BASIC/CMD and BBASIC/CMD in relative byte 84H contains the default number of file buffer areas allocated during SUPERBASIC initialization. Valid values are 00 through 0F. And relative byte 85H determines the type of buffer areas opened during initialization. If this byte is non-zero then the required "V" for user defined record lengths during random I/O is not required upon initialization, e.g., BASIC 3V<ENTER> if the byte is zero, or BASIC<ENTER> if the byte is non-zero. The "V" buffer areas are 545 bytes each, and non-"V" buffer areas are 289 bytes each.

******** ZAP 008 ******** 01/24/85 ******** - BACKUP/CMD (optional)
******** ZAP 008 updated 01/01/86 ********
******** ZAP 008 updated 10/22/90 ********
******** ZAP 008 updated 06/15/99 ********

The automatic feature of verifying a diskette after all granules are duplicated onto the destination diskette can be modified to: (1) never verify, or (2) to verify only if the DOS verify is on.

File: BACKUP/CMD, Relative sector: 9, relative byte: DEH

NEVER RE-VERIFY	PER LIBRARY COMMAND VERIFY
Change: 44**AF 2F**32	Change: 44**AF 2F**32
To: 44**AF AF**32	To: 44**96 2F**32

******** ZAP 010 ******** 02/11/85 ******** - INFO (informational)
******** ZAP 010 updated 12/30/85 ********
******** ZAP 010 updated 03/29/89 ********

CAT/CMD dated 10/22/88 or later automatically turns SKIP on or off, and CONFIGs for side and/or volumes with or without the "N" parameter. NOTE: The "N" parameter was changed to "T" in June 1999.

This ZAP is for transmitting information on the release of Version 1.71. It is impossible to ZAP all of the modules involved in reading NeWDOS/80 diskettes with the GPL parameter not equal to 2. The original release of Version 1.7 will only read a NeWDOS/80 diskette with a GPL=2. SYSRES/SYS has been updated to handle extended directories on NeWDOS/80 version 2 diskettes. See DCT+01 bit 5 definition.

DCT+01: (NIL is indicated by having ALL bits set.)

BIT	0	Physical drive LSB.
BIT	1	Physical drive.
BIT	2	Physical drive MSB.
BIT	3	MEMDISK.
BIT	4	Set to one indicates HARD/MEMORY. A zero indicates floppy.
BIT	5	Locks DCT. (no automatic update to DCT+06 through DCT+08)
BIT	6	Set to one to indicate software write protect.
BIT	7	Set to zero to indicate drive in system.

For floppies, DCT+06 through DCT+08 are updated each time the DOS must invoke automatic data recognition. The default decimal values are:

	DCT+06	DCT+07	DCT+08	DCT+09
3½", 5¼" SD/SS	5	2	10	8
3½", 5¼" SD/DS	5	4	10	18
3½", 5¼" DD/SS	6	3	18	16
3½", 5¼" DD/DS	6	6	18	32

You can establish other values to read a special format pattern by placing the respective values in the DCT and set bit 5 of DCT+01.

EXAMPLE: NeWDOS/80 PDRIVE setting with SPT=36, GPL=8, and DDGA=6.

```
    DCT+00 = 100110xxB (xx is the stepping rate) 98H = 6mS
    DCT+02 = 02H
    DCT+06 = 05H ("...there are still 5 sectors per granule,...")
    DCT+07 = 08H (GPL)
    DCT+08 = 28H (sectors per LUMP)
    DCT+09 = 1CH (DDGA * 5 - 2, 5 = DCT+06, 2 = 1 GAT + 1 HIT)
```

The file CAT/CMD has been modified to set bit 5 of DCT+01. Bit 5 of DCT+01 will be reset if the new "N" or "L" parameter is specified (while reading a MULTIDOS diskette). In addition to the "N" and "L", CAT/CMD has a "W" parameter to write, onto a NeWDOS/80 diskette's GAT sector, the necessary two bytes in relative bytes 0CCH and 0CDH to ensure the entire NeWDOS/80 disk space can be used. NeWDOS/80 doesn't care that these two bytes are set to a non-zero value. If the "W" parameter is used to update these two bytes, the affected disk can function normally in a NeWDOS/80 environment. Use the "W" only on NeWDOS/80 diskettes!

June 1999 update: The "N" parameter has been changed to "T", the "W" parameter has been changed to "Z", and the "L" parameter eliminated.

The files Allocate/DOL, COPY/CMD, and VFU/CMD has been modified to limit the tracks accessed if the bytes at 0CCH and 0CDH are zero. This enables MULTIDOS to write to a NeWDOS/80 diskette without trying to write to a non-existent track. However, MULTIDOS probably will indicate a full diskette if these two bytes are zero.

The file CDIR/CMD will clear the entire directory, provided the diskette was accessed via CAT/CMD prior to CDIR.

The file FMAP/CMD will display the correct information, provided the diskette was accessed via CAT/CMD prior to FMAP.

The library command DIR (Library1/EXT), and Minidos will display correct NeWDOS/80 directory data, provided the diskette was accessed via CAT/CMD prior to these commands.

The file ZAP/CMD will not complain "Garbage in HIT sector." for extended NeWDOS/80 directories if the diskette was accessed via CAT/CMD prior to the "X" function in ZAP. In addition, ZAP can now display all files on a NeWDOS/80 disk, with a GPL greater than two.

To ensure correct reading of a NeWDOS/80 diskette, issue a CAT :d command prior to reading the NeWDOS/80 diskette. CONFIG (Library2/EXT) will indicate a locked DCT. Subsequent CATs to a different diskette in the drive that had the NeWDOS/80 diskette will automatically lock and/or unlock the DCT.

Of course, all of this depends on correct PDRIVE settings.

******** ZAP 011 ******** 02/17/86 ******** - BACKUP, FORMAT, and ZAP/CMD (optional)

Faster disk I/O by using a different interleave with double-density diskettes. Interleave allows logical contiguous sectors of data on a given track to be mapped onto non-adjacent physical sectors. An interleave of two (single-density) means that every second physical sector is transferred as the next contiguous logical sector of data. An interleave of three (double-density) means that every third physical sector is transferred as the next contiguous logical sector of data. An interleave of six (Model 3 TRSDOS®) means that every sixth physical sector is transferred as the next contiguous logical sector of data. This also means the number of sectors per track divided by the interleave is transferred in one revolution of the diskette. If the diskette is single-density with an interleave of two and 10 sectors per track, then each revolution of the diskette transfers five sectors requiring two revolutions of the diskette to transfer one track of data. If the diskette is double-density with an interleave of three and 18 sectors per track, then each revolution of the diskette transfers six sectors requiring three revolutions of the diskette to transfer one track of data. If the diskette is Model 3 TRSDOS® with an interleave of six and 18 sectors per track, then each revolution of the diskette transfers three sectors requiring six revolutions of the diskette to transfer one track of data.

The time the system requires to transfer one sector of data is the primary factor in determining the interleave. If a read operation is in progress and the interleave is three, after a sector of data is read into the sector buffer, the sector buffer is processed while the read/write head is passing over the next two sectors. If the system cannot process the sector buffer during the two-sector time available, then the system has to wait a full revolution before the next logical sector can be read from the diskette. For example, Model 3 TRSDOS® cannot process a sector buffer in a two-sector time; therefore, Model 3 TRSDOS® uses an interleave of six. Without using a calculator, you can see why Model 3 TRSDOS® requires 1.2 seconds —— 6 times 200mS (time required for one revolution of the diskette) —— to read a full track of data.

An interleave of three was used with the introduction of DBLDOS™ because the Model I has a standard CPU speed of 1.77 MHz. The other DOS makers also used an interleave of three.

The interleave for double-density diskettes can be reduced to two because the Model 4/4P/4D has a standard CPU speed of 4 MHz. In fact, a CPU speed greater than 2.5 MHz can handle an interleave of two if the operating system is MULTIDOS. An interleave of two is 33 percent faster than a interleave of three —— 600mS vs 400mS.

If you have a high speed modification, I recommend you ZAP a byte in BACKUP/CMD, FORMAT/CMD, and ZAP/CMD to incorporate an interleave of two for double-density diskettes. The interleave for single-density diskettes remains at two. In all cases a value of zero is an interleave of three, and a non-zero value is an interleave of two.

File: BACKUP/CMD, Relative sector: 0, relative byte: A0H
FORMAT/CMD, Relative sector: 0, relative byte: B0H
ZAP/CMD, Relative sector: 0, relative byte: 81H

******** ZAP 001 ******** 02/27/85 ******** - SYSTEM (optional)
******** ZAP 001 updated 12/30/85 ********
******** ZAP 001 updated 07/05/88 ********
******** ZAP 001 updated 05/04/91 ********

Information on the initialization configuration for MULTIDOS can be found on track zero sector zero.

```
    Hex 00  0000 1100 0000 0000 0000 0000 0000 0000 ................
        10  0000 0000 0001 FE45 0000 208F 0007 CC01 .......E.. .....
        20  003C 0000 0006 AE47 3C00 5000 06C3 0044 .<.....G<......D

        E0  0000 0000 0000 0000 0000 0000 0000 0000 ................
    DM  F0  00FF 0100 8503 C800 0000 0000 0000 0000 ................
              (partial track 0, sector 0)
```

Relative byte F0H - EPSON™ graphics conversion.
The LIBRARY command KEYBRD modifies this byte. If this byte is non-zero, then the FORMS routine converts characters in the range from 80H through 0BFH to the required values of 0A0H through 0DFH for the earlier EPSON™ printers to replicate TRS-80® screen graphic characters. EPSON printers equipped with GRAFTRAX 80™ can still produce TRS-80® graphics. However, the GRAFTRAX PLUS™ modification disables the reproduction of TRS-80® graphics.

Relative byte F1H - Video Width.
This byte is modified by the library command KEYBRD. If this byte is non-zero, then power-up is in the 80 column mode.

Relative byte F2H - Type-Ahead.
The Type-Ahead feature is activated on power-up if this byte has a ZERO value. The TYPE library command does not modify this byte.

Relative byte F3H - SPOOLER.
The SPOOLER is activated on power-up if this byte is non-zero. The file SPOOL/CMD does not modify this byte.

Relative byte F4H - Disk I/O tries and side compare.
MULTIDOS is distributed with the number of disk I/O tries set to 5. This is a good value. If a diskette starts to give you errors, BACKUP the file(s) immediately. Increasing the number of I/O tries may enable you to read the file; however, eventually it will fail. The number of I/O tries is in the lower nibble (bits 0 to 3), yielding the maximum number of tries to 15. The higher nibble uses bit 7 to enable side compare for double-sided diskettes. If this bit is set, side compare is enabled. If this bit is reset, then side compare is disabled. MULTIDOS **versions earlier than 2.00 did not have side compare enabled.**

Relative byte F5H - Head load delay.
Whenever MULTIDOS selects a different drive, even if the motors are up to speed, a delay of about 15mS occurs before an attempt is made to read/write to the newly selected drive. The amount of delay is increased 4.096mS for each unit over the distributed value of 3.

Relative byte F6H - SPOOLER delay.
This byte controls the amount of time the SPOOLER waits after each character is sent to the printer before the SPOOLER attempts to send another character. The purpose of this delay is to keep the printer running at full speed when the SPOOLER is active. The distributed value of C8H (the optimum value for an EPSON MX-80F/T printer) is placed into 48F8H (subject to change) during system initialization. You can modify this value, via POKEs or DEBUG while the SPOOLER is outputting to your printer, to obtain an optimum value. An optimum value is a value that the printer runs at approximately full speed and the system does not seem sluggish. If you select a value too large the printer runs at normal speed; however, you will encounter delays in keyboard responses and a hesitant display.

Relative byte F7H - Default value for SM0.
If any of the high three bits in SM0 are set, you may have unusual filenames. However, keep in mind not all operating systems recognize the new syntax. Please refer to the discussion of the bit definitions for SM0 in your manual.

******** ZAP 002 ******** 02/27/85 ******** - Open/DOL (optional)
******** ZAP 002 updated 11/30/90 ********
******** ZAP 002 updated 08/29/94 ********
******** ZAP 002 updated 07/07/95 ********
******** ZAP 002 updated 09/01/96 ********

Open/DOL error codes. The error code most application programs expect when an attempt is made to access a file, and the "file is not found", is 18H. This error can occur in three distinct circumstances. 1) The drivespec given does not contain the file, 2) the drivespec given is not available, or 3) the file was searched for and not found on all mounted drives. Error code 18H is correct for case 1, error code 08H is correct for case 2, and error code 1FH is correct for case 3. MULTIDOS can return the correct error code by ZAPping two bytes in Open/DOL. Please note, with this zap, some application programs might return "Program not found" if an access is made to a file without a drivespec. *Your diskette has this ZAP applied.*

File: Open/DOL, Relative sector: 4, relative byte: 58H

Change: C9**18 18**02
    To: C9**08 1F**02

******** ZAP 003 ******** 02/27/85 ******** - ZAP/CMD (optional)

Default response for ZAP/CMD menu. ZAP/CMD's first sector in relative byte 80H, contains the menu default character. This is distributed with 43H, C, "Copy sectors". The value here is ZAP's default when you press <ENTER> at the "Choice" prompt. Zap 44H if you want the default to be "Disk sectors", 46H if you want the default to be "File sectors", etc. The new default does not go into effect until ZAP is initialized again.

******** ZAP 004 ******** 02/27/85 ******** - BASIC/CMD (optional)

Default number of file buffer areas for SUPERBASIC. BASIC/CMD's first sector (relative sector 0) in relative byte 80H contains the default number of file buffer areas allocated during SUPERBASIC initialization. Valid values are 0 to 0F. Relative byte 82H determines the type of buffer areas opened during initialization. If this byte is non-zero then the required "V" for user defined record lengths during random I/O is not required upon initialization, e.g., BASIC 3V<ENTER> if the byte is zero, or BASIC<ENTER> if the byte is non-zero. The "V" buffer areas are 545 bytes each, and non-"V" buffer areas are 289 bytes each.

******** ZAP 005 ******** 02/27/85 ******** - BASIC/CMD (optional)

Default variable types for SUPERBASIC. Relative byte 81H in BASIC/CMD's first sector contain SUPERBASIC's internal code for the type of variable upon initialization and CLEAR.

DESIRED VARIABLE TYPE	APPROPRIATE CODE
Integer precision	02
Strings	03
Single precision	04
Double precision	08

The initialization code has the necessary checks to validate a correct code. If an incorrect value is encountered, SUPERBASIC uses the value of 04.

******** ZAP 006 ******** 02/27/85 ******** - BACKUP/CMD (optional)
******** ZAP 006 updated 12/30/85 ********
******** ZAP 006 updated 10/22/90 ********
******** ZAP 006 updated 08/26/94 ********
******** ZAP 006 updated 07/07/95 ********

The automatic feature of verifying a diskette after all granules are duplicated onto the destination diskette can be modified to: (1) never verify, or (2) to verify only if the DOS verify is on.

File: BACKUP/CMD, Relative sector: 9, relative byte: CFH

NEVER RE-VERIFY	PER LIBRARY COMMAND VERIFY
Change: 44**AF 2F**32	Change: 44**AF 2F**32
To: 44**AF AF**32	To: 44**96 2F**32

******** ZAP 007 ******** 02/27/85 ******** - SYSTEM (optional)
******** ZAP 007 DELETED 12/30/85 ********

This ZAP dealt with the one second delay for reads. This feature has been eliminated.

******** ZAP 008 ******** 02/27/85 ******** - INFO (information)
******** ZAP 008 updated 12/30/85 ********

This ZAP is for transmitting information on the release of Version 1.71. It is impossible to ZAP all of the modules involved in reading NeWDOS/80 diskettes with the GPL parameter not equal to 2. The original release of Version 1.7 will only read a NeWDOS/80 diskette, with a GPL=2.

SYSRES/SYS has been updated to handle extended directories on NeWDOS/80 version 2 diskettes. See DCT+01 bit 5 definition.

DCT+01: (NIL is indicated by having ALL bits set.)

BIT 0 Physical drive LSB.
BIT 1 Physical drive.
BIT 2 Physical drive MSB.
BIT 3 MEMDISK.
BIT 4 Set to one indicates HARD/MEMORY. A zero indicates floppy.
BIT 5 Locks DCT. (no automatic update to DCT+06 through DCT+08)
BIT 6 Set to one to indicate software write protect.
BIT 7 Set to zero to indicate drive in system.

For floppies, DCT+06 through DCT+08 are updated each time the DOS must invoke automatic data recognition. The default values are:

	DCT+06	DCT+07	DCT+08	DCT+09
3½", 5¼" SD/SS	5	2	10	8
3½", 5¼" SD/DS	5	4	10	18
3½", 5¼" DD/SS	6	3	18	16
3½", 5¼" DD/DS	6	6	18	32

You can establish other values to read a special format pattern by placing the respective values in the DCT and set bit 5 of DCT+01.

EXAMPLE: NeWDOS/80 PDRIVE setting with SPT=36, GPL=8, and DDGA=6.

    DCT+00 = 100110xxB (xx is the stepping rate) 98H = 6mS
    DCT+02 = 02H
    DCT+06 = 05H ("...there are still 5 sectors per granule,...")
    DCT+07 = 08H (GPL)
    DCT+08 = 28H (sectors per LUMP)
    DCT+09 = 1CH (DDGA * 5 - 2, 5 = DCT+06, 2 = 1 GAT + 1 HIT)

The file CAT/CMD has been modified to set bit 5 of DCT+01. Bit 5 of DCT+01 will be reset if the new "N" or "L" parameter is specified (while reading a MULTIDOS diskette). In addition to the "N" and "L", CAT/CMD has a "W" parameter to write, onto a NeWDOS/80 diskette's GAT sector, the necessary two bytes in relative bytes 0CCH and 0CDH to ensure the entire NeWDOS/80 disk space can be used. NeWDOS/80 doesn't care that these two bytes are set to a non-zero value. If the "W" parameter is used to update these two bytes, the affected disk can function normally in a NeWDOS/80 environment. Use the "W" only on NeWDOS/80 diskettes!

The files Allocate/DOL, COPY/CMD, and VFU/CMD has been modified to limit the tracks accessed if the bytes at 0CCH and 0CDH are zero. This enables MULTIDOS to write to a NeWDOS/80 diskette without trying to write to a non-existent track. However, MULTIDOS probably will indicate a full diskette if these two bytes are zero.

The file CDIR/CMD will clear the entire directory, provided the diskette was accessed via CAT/CMD prior to CDIR.

The file FMAP/CMD will display the correct information, provided the diskette was accessed via CAT/CMD prior to FMAP.

The library command DIR (Library1/EXT), and Minidos will display correct NeWDOS/80 directory data, provided the diskette was accessed via CAT/CMD prior to these commands.

The file ZAP/CMD will not complain "Garbage in HIT sector." for extended NeWDOS/80 directories if the diskette was accessed via CAT/CMD prior to the "X" function in ZAP. In addition, ZAP can now display all files on a NeWDOS/80 disk, with a GPL greater than two.

To ensure correct reading of a NeWDOS/80 diskette, issue a CAT :d command prior to reading the NeWDOS/80 diskette. CONFIG (Library2/EXT) will indicate a locked DCT and can unlock the DCT with a CONFIG :d (N) followed by a CONFIG :d (P=p). This is an additional method of unlocking the DCT other than CAT (N).

Of course, all of this depends on correct PDRIVE settings.

******** ZAP 009 ******** 06/02/85 ******** - UltraTerm (interface)

Modifications for UltraTerm to work with Model 4 MULTIDOS. The first two ZAPS will not inhibit UltraTerm from working with other DOSes on the Model I, Model III or Model 4 operating in the Model III mode.

File sector 0, relative byte 3DH

Change: 3A **2501 FE49 20**
    To: 3A **5500 FE01 28**

File sector 1, relative byte CDH

Change: 3A **2501 FE49 20**
    To: 3A **5500 FE01 28**

This last ZAP will make UltraTerm operate only with Model 4 MULTIDOS. However, I have decided to incorporate a change into the resident DOS to provide a vector to the CLS routine from the address 01C9H. This ZAP is not necessary for release 2.00 or later. The screen can be cleared with either XOR A followed by RST 0 or CALL 01C9H for MULTIDOS version 2.00 or later.

File sector 17, relative byte 38H

Change: **CDC9 01**
    To: **AFC7 00**

******** ZAP 010 ******** 06/02/85 ******** - STERM/CMD (interface)

Modifications for STERM/CMD to work with Model 4 MULTIDOS. The following three ZAPS will not inhibit STERM/CMD from working with other DOSes on the Model I, Model III, or Model 4 operating in the Model III mode. The last ZAP overcomes three ROM calls starting at 61F7H.

File sector 1, relative byte C6H

Change: 3A**1B 02FE 7E20**
    To: 3A**55 00FE 0128**

File sector 11, relative byte 76H

Change: 3A**1B 02FE 7E28**
    To: 3A**55 00FE 0120**

File sector 17, relative byte AAH

Change: 00**02 02A6 57**
    To: 00**01 3200 4110**
27E8 0364 000A 0001 00E5 0105 0521 0041
5E23 5623 E33E 2FB7 3CED 5230 FB19 E30D
2804 FE30 2805 CD33 000E 0110 E3E1 C901
05F7 61C3 0A41 0202 A657

******** ZAP 011 ******** 06/09/85 ******** - INFO (information)
************* UPDATE ZAP 008 *************
******** ZAP 011 updated 03/29/89 ********

The file CAT/CMD has been modified to automatically set bit 5 of DCT+01, eliminating the "L" and "U" parameters. CAT/CMD will automatically lock or unlock the DCT as appropriate. In addition, an "N" parameter is added to automatically turn SKIP on or off, and CONFIG for side and/or volume. CAT/CMD now automatically turns SKIP on or off, and CONFIGs for side and/or volume with or without the "N" parameter —— version 10/22/88 or later.

******** ZAP 012 ******** 07/04/85 ******** (mandatory)
******** ZAP 012 DELETED 01/01/86 ********

This ZAP dealt with booting a double-sided single-volume system diskette.

******** ZAP 013 ******** 09/17/85 ******** - VC/CMD (interface)

VisiCalc was written to use the address at 37E8H for printer status checking. This will work fine on a Model 4 running in the Model III mode. However, Model 4 MULTIDOS only uses the F8H port to interface with the printer. Therefore, we need to ZAP VisiCalc to work with Model 4 MULTIDOS. The nice thing about these ZAPs is they do not disable VisiCalc from working in the Model III mode.

Version VC-150Y0-T83

Relative sector: 83, Relative byte: 9CH

Change: C9**F5 C5D5 E5CD 8D02 280A CD2D A4E1 D1C1 F137 1808 CD35 A4E1 D1C1 F1B7**
To: C9**C5 D5E5 F53A 4038 E604 2809 CD2D A4F1 37E1 D1C1 C9CD 35A4 F1B7 18F5**

Relative sector: 84, Relative byte: CCH

Change: **3203 42**CD
To: **0000 00**CD

Relative sector: 84, Relative byte: 26H

Change: **3203 42**CD 35A4 **3AE8 37**E6
To: **0000 00**CD 35A4 **DBF8 00**E6

Version VC-160Y0-T83

Relative sector: 83, Relative byte: BAH

Change: C9**F5 C5D5 E5CD 8D02 280A CD4B A4E1 D1C1 F137 1808 CD53 A4E1 D1C1 F1B7**
To: C9**C5 D5E5 F53A 4038 E604 2809 CD4B A4F1 37E1 D1C1 C9CD 53A4 F1B7 18F5**

Relative sector: 84, Relative byte: F0H

Change: **3203 42**CD
To: **0000 00**CD

Relative sector: 85, Relative byte: 46H

Change: C9**32 0342** CD53 A4**3A E837**
To: C9**00 0000** CD53 A4**DB F800**

******** ZAP 014 ******** 02/17/86 ******** - BACKUP, FORMAT, and ZAP/CMD (optional)

Faster disk I/O by using a different interleave with double-density diskettes. Interleave allows logical contiguous sectors of data on a given track to be mapped onto non-adjacent physical sectors. An interleave of two (single-density) means that every second physical sector is transferred as the next contiguous logical sector of data. An interleave of three (double-density) means that every third physical sector is transferred as the next contiguous logical sector of data. An interleave of six (Model 3 TRSDOS®) means that every sixth physical sector is transferred as the next contiguous logical sector of data. This also means the number of sectors per track divided by the interleave is transferred in one revolution of the diskette. If the diskette is single-density with an interleave of two and 10 sectors per track, then each revolution of the diskette transfers five sectors requiring two revolutions of the diskette to transfer one track of data. If the diskette is double-density with an interleave of three and 18 sectors per track, then each revolution of the diskette transfers six sectors requiring three revolutions of the diskette to transfer one track of data. If the diskette is Model 3 TRSDOS® with an interleave of six and 18 sectors per track, then each revolution of the diskette transfers three sectors requiring six revolutions of the diskette to transfer one track of data.

The time the system requires to transfer one sector of data is the primary factor in determining the interleave. If a read operation is in progress and the interleave is three, after a sector of data is read into the sector buffer, the sector buffer is processed while the read/write head is passing over the next two sectors. If the system cannot process the sector buffer during the two-sector time available, then the system has to wait a full revolution before the next logical sector can be read from the diskette. For example, Model 3 TRSDOS® cannot process a sector buffer in a two-sector time; therefore, Model 3 TRSDOS® uses an interleave of six. Without using a calculator, you can see why Model 3 TRSDOS® requires 1.2 seconds —— 6 times 200mS (time required for one revolution of the diskette) —— to read a full track of data.

An interleave of three was used with the introduction of DBLDOS™ because the Model I has a standard CPU speed of 1.77 MHz. The other DOS makers also used an interleave of three.

The interleave for double-density diskettes can be reduced to two because the Model 4/4P/4D has a standard CPU speed of 4 MHz. In fact, a CPU speed greater than 2.5 MHz can handle an interleave of two if the operating system is MULTIDOS. An interleave of two is 33 percent faster than a interleave of three —— 600mS vs 400mS.

You can ZAP a byte in BACKUP/CMD, FORMAT/CMD, and ZAP/CMD to incorporate an interleave of two for double-density diskettes. The interleave for single-density diskettes remains at two. In all cases a value of zero is an interleave of three, and a non-zero value is an interleave of two.

File: BACKUP/CMD, Relative sector: 0, relative byte: A0H
FORMAT/CMD, Relative sector: 0, relative byte: B0H
ZAP/CMD, Relative sector: 0, relative byte: 81H

******** ZAP 015 ******** 06/22/86 ******** - BASIC/CMD (mandatory)
******** ZAP 015 DELETED 03/29/89 ********

Only apply this mandatory ZAP, if BASIC/CMD is dated prior to the ZAP date.

Fix for PRINT@
    Relative File Sector 27d or 1BH

    Relative byte 7EH
    Change: **17**
        To: **1A**

    Relative byte 94H
    Change: **181F CD46 1AE5 3A04 0021 80F8**
             E610 2003 2100 FC19 DA8B 1B21 003C 19CD

        To: **E1CF 2CBF C9CD 461A EB3A 0400**
             CD51 4445 4F7C B7C2 8B1B EBCD 831A 18D8

******** ZAP 016 ******** 06/22/86 ******** - BASIC/CMD (mandatory)
******** ZAP 016 DELETED 03/29/89 ********

Only apply this mandatory ZAP, if BASIC/CMD is dated prior to the ZAP date.

Fix for LINEINPUT# (LINEINPUT sequential disk file)
    Relative File Sector 6, Relative byte F4H

    Change: **4220**
        To: **7A28**

******** ZAP 017 ******** 06/22/86 ******** - BASIC/CMD (mandatory)
******** ZAP 017 DELETED 03/29/89 ********

Only apply this mandatory ZAP, if your file is dated prior to the ZAP date.

Fix for TAN
    Relative File Sector 27d or 1BH, Relative Byte 3AH

    Change: **23**
        To: **22**

    Relative File Sector 44d or 2CH, Relative byte 5BH

    Change: **17 30FB 10F9**
             E6FE 0F12 130D 28B9 7E23 0730 F5E1 18C8
             CD08 15CD 7858 F1D1 CD09 15F5 C1

        To: **CB 2730 FA10**
             F80F 1213 0D28 BA7E 2307 30F5 E118 C9CD
             0815 CD78 58F1 D1CD 0915 F5C1 EB

******** ZAP 018 ******** 05/04/91 ******** - SYSRES/SYS (optional)

This ZAP enables MULTIDOS to detect a floppy diskette when the Z80 is operating above 4 MHz. You will need this if you are getting "Drive not available" error messages when a floppy is mounted properly in a particular floppy disk drive.

File: SYSRES/SYS

MULTIDOS 2.1
Relative file sector 7, relative byte 67H

Change: **36**
To: **6C**

The 6C value enables MULTIDOS to work with the Z80 operating at 8 MHz.

MULTIDOS 2.11 or later has this ZAP applied in another location.

******** ZAP 019 ******** 12/27/93 ******** - PCOPY/CMD (optional)

In PCOPY/CMD's first sector (relative sector 0) you can see "filename#01". This is the skeleton for storage of the filename and default extension. You can change the first character of the extension (relative byte 8CH) to:

! " # $ % & < > ? @

and execute PCOPY as documented. If you would like to have a lower case letter or graphic character, then ZAP relative byte 8FH to a non-zero value.

File sector 0, relative byte 8FH

Change: 31**00**
To: 31**01**

******** ZAP 020 ******** 07/01/95 ******** - Help/EXT (mandatory)
******** ZAP 020 DELETED 06/03/99 ********

The resetting of SM1 bit 4, should be exclusive to BASIC. This oversight became evident when I modified BASIC to complete a DO file when executed via CMD"DO ***file***". This is an obscure problem, and the consecution required to manifest the problem is a DO file executed from BASIC with a DIR command (sets SM1 bit 4) followed by a HELP command. e.g., CMD"DO CRASH" and the file CRASH/IDO has DIR<ENTER> HELP<ENTER>.

File sector 0, relative byte 89H

Change: 31**C1**
To: 31**D1**

******** ZAP 021 ******** 07/01/95 ******** - VFU/CMD (optional)

When selecting file with VFU, a plus sign is used to indicate a file is selected. This character can be changed to any displayable character, including special characters.

File sector 0, relative byte 80H

Change: **2B**
    To: any displayable character (21 through FF)

******** ZAP 022 ******** 07/01/95 ******** - VFU/CMD (optional)

When VFU is copying or moving a file, VFU opens the source file (to get date, time, and size), then opens the destination file (to get date, time, and size), then reads the source file, then writes to the destination file. While VFU is going through these steps, various characters appear in front of the filespecs. You can modify these characters as follows:

File sector 0, relative byte 81H - open source

Change: **2B**
    To: any displayable character (21 through FF)

File sector 0, relative byte 82H - open destination

Change: **23**
    To: any displayable character (21 through FF)

File sector 0, relative byte 83H - read source

Change: **24**
    To: any displayable character (21 through FF)

File sector 0, relative byte 84H - write to destination

Change: **2A**
    To: any displayable character (21 through FF)

As an initial test, try these four values: 81, 82, 84, and 88.

******** ZAP 023 ******** 07/10/95 ******** - BACKUP, FORMAT, and ZAP/CMD (optional)

Faster double-density floppy disk I/O by track to track sector skewing. MULTIDOS has always used sector skewing (set at five) to enable faster disk I/O since its inception. The five was set to handle slower drives with a track to track access speed greater than 20mS. Now that all newer disk drives can handle a track to track access speed of 6mS (limited by the Floppy Disk Controller [FDC] chip), I felt compelled to exploit this by decreasing the skew from five to a value compatible with the track to track stepping rate.

You can ZAP a byte in BACKUP/CMD, FORMAT/CMD, and ZAP/CMD to incorporate the faster skew for double-density diskettes. The sector skew for single-density diskettes remains at three. In all cases a value of zero is a skew of five, and a non-zero value uses the DCT track to track stepping rate. i.e., 30mS uses a skew of 5, 20mS uses a skew of 4, 12mS uses a skew of 3, and 6mS uses a skew of 2.

File: BACKUP/CMD, Relative sector: 0, relative byte: A1H
FORMAT/CMD, Relative sector: 0, relative byte: B1H
ZAP/CMD, Relative sector: 0, relative byte: 82H

******** ZAP 024 ******** 07/10/95 ******** - BACKUP, FORMAT, and ZAP/CMD (optional)

Formatting diskettes for use on IBM and clones require an index mark (MULTIDOS starting using index marks in 1983 because the LOBO MAX-80 required index marks). And I have found some TRS-80® to PC utilities have had difficulty in reading TRS-80® diskettes created by MULTIDOS. What I suspected the problem to be is the length of the pre-index, GAP 0 (the length of GAP 0 used by MULTIDOS is zero). MULTIDOS can be modified to increase the pre-index, GAP 0, from zero bytes to 32 bytes (note: the standard IBM format has 92 pre-index bytes, which never worked on the TRS-80®). The value of 32 was derived by starting from 92 and reducing the count until it worked then, for insurance, halving the result. MULTIDOS 3.00 had this set at 32, and one customer reported a problem. Therefore, in MULTIDOS 3.10, the pre-index is set to zero. For those who want to try the 32, change the following bytes to a non-zero value.

File: BACKUP/CMD, Relative sector: 0, relative byte: A2H
FORMAT/CMD, Relative sector: 0, relative byte: B2H
ZAP/CMD, Relative sector: 0, relative byte: 83H

******** ZAP 025 ******** 1/27/98 ******** - EMURES/SYS (optional)
******** ZAP 025 DELETED 3/26/99 ********

The key repeat timing must be adjusted for fast PC's, because PC's emulate a TRS-80® at approximately 2/15th the CPU speed. Timing changes can be made only with EMURES/SYS dated 01/23/98 or later.

File sector 7, relative byte 60H – repeat rate
Change: 08 to a higher value to increase the timing between repeating (do not use 00).

File sector 7, relative byte A6H – delay before repeat
Change: 30 to a higher value to increase the delay before repeat (00 is maximum).

Key Codes

DEC	HEX	KEY(s)
000	00	(not available)
001	01	BREAK or CTRL·A
002	02	CTRL·B
003	03	CTRL·C
004	04	CTRL·D
005	05	CTRL·E
006	06	CTRL·F
007	07	CTRL·G
008	08	← or CTRL·H
009	09	→ or CTRL·I
010	0A	↓ or CTRL·J
011	0B	CTRL·K (MAX-80 ↑)
012	0C	CTRL·L
013	0D	ENTER or CTRL·M
014	0E	CTRL·N
015	0F	CTRL·O
016	10	CTRL·P
017	11	CTRL·Q
018	12	CTRL·R
019	13	CTRL·S
020	14	CTRL·T
021	15	CTRL·U
022	16	CTRL·V
023	17	CTRL·W
024	18	SHIFT·← or CTRL·X
025	19	SHIFT·→ or CTRL·Y
026	1A	SHIFT·↓ (not later MODEL I and not MODEL III) or CTRL·Z
027	1B	SHIFT·↑ or CTRL·;
028	1C	CTRL·,
029	1D	CTRL·-
030	1E	CTRL·.
031	1F	CLEAR or CTRL·/
032	20	SPACE-BAR
033	21	!
034	22	"
035	23	#
036	24	$
037	25	%
038	26	&
039	27	'
040	28	(
041	29	)
042	2A	*
043	2B	+
044	2C	,
045	2D	-
046	2E	.
047	2F	/

Key Codes

DEC	HEX	KEY(s)
048	30	0
049	31	1
050	32	2
051	33	3
052	34	4
053	35	5
054	36	6
055	37	7
056	38	8
057	39	9
058	3A	:
059	3B	;
060	3C	<
061	3D	=
062	3E	>
063	3F	?
064	40	@
065	41	A
066	42	B
067	43	C
068	44	D
069	45	E
070	46	F
071	47	G
072	48	H
073	49	I
074	4A	J
075	4B	K
076	4C	L
077	4D	M
078	4E	N
079	4F	O
080	50	P
081	51	Q
082	52	R
083	53	S
084	54	T
085	55	U
086	56	V
087	57	W
088	58	X
089	59	Y
090	5A	Z
091	5B	[(MAX-80), (others the ↑ key)
092	5C	\ (MAX-80), (MODEL 4 or ESOTERIC default F1)
093	5D	] (MAX-80), (MODEL 4 or ESOTERIC default F2)
094	5E	^ (MAX-80), (MODEL 4 or ESOTERIC default F3)
095	5F	_ (MAX-80)

Key Codes

DEC	HEX	KEY(s)
096	60	` (SHIFT·@)
097	61	a
098	62	b
099	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	79	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{ (MAX-80), (MODEL 4 or ESOTERIC default F4 [RIGHT-SHIFT·F1])
124	7C	| (MAX-80), (MODEL 4 or ESOTERIC default F5 [RIGHT-SHIFT·F2])
125	7D	} (MAX-80), (MODEL 4 or ESOTERIC default F6 [RIGHT-SHIFT·F3])
126	7E	~ (MAX-80)
127	7F	± (MAX-80)

Key Codes MAX-80, Model 4, and ESOTERIC

DEC	HEX	KEY(s)
128	80	ALT·@ (see note at bottom of page)
129	81	ALT·A or ALTCTRL·BREAK (see note at bottom of page)
130	82	ALT·B
131	83	ALT·C
132	84	ALT·D
133	85	ALT·E
134	86	ALT·F
135	87	ALT·G
136	88	ALT·H or ALTCTRL·←
137	89	ALT·I or ALTCTRL·→
138	8A	ALT·J or ALTCTRL·↓
139	8B	ALT·K (MAX-80 ALTCTRL·↑)
140	8C	ALT·L
141	8D	ALT·M or ALTCTRL·ENTER
142	8E	ALT·N
143	8F	ALT·O
144	90	ALT·P
145	91	ALT·Q
146	92	ALT·R
147	93	ALT·S
148	94	ALT·T
149	95	ALT·U
150	96	ALT·V
151	97	ALT·W
152	98	ALT·X, (MODEL 4 and ESOTERIC ALTCTRL·LEFT-SHIFT·←)
153	99	ALT·Y, (MODEL 4 and ESOTERIC ALTCTRL·LEFT-SHIFT·→)
154	9A	ALT·Z, (MODEL 4 and ESOTERIC ALTCTRL·LEFT-SHIFT·↓)
155	9B	(MAX-80 ALT·[), (MODEL 4 and ESOTERIC ALTCTRL·LEFT-SHIFT·↑)
156	9C	(MAX-80 ALT·\)
157	9D	(MAX-80 ALT·])
158	9E	(MAX-80 ALT·^)
159	9F	(MAX-80 ALT·_)
160	A0	ALTCTRL·SPACE-BAR or ALT·SHIFT·@
161	A1	ALTCTRL·! or ALT·a
162	A2	ALTCTRL·" or ALT·b
163	A3	ALTCTRL·# or ALT·c
164	A4	ALTCTRL·$ or ALT·d
165	A5	ALTCTRL·% or ALT·e
166	A6	ALTCTRL·& or ALT·f
167	A7	ALTCTRL·' or ALT·g
168	A8	ALTCTRL·(or ALT·h
169	A9	ALTCTRL·) or ALT·i
170	AA	ALTCTRL·* or ALT·j
171	AB	ALTCTRL·+ or ALT·k
172	AC	ALTCTRL·, or ALT·l
173	AD	ALTCTRL·- or ALT·m
174	AE	ALTCTRL·. or ALT·n
175	AF	ALTCTRL·/ or ALT·o

NOTE: ALT is the CLEAR key after pressing LEFT-SHIFT·F2 (MODEL 4 and ESOTERIC), SHIFT·F2 (MAX-80). CAPS mode is considered.
ALTCTRL is the CLEAR key after pressing LEFT-SHIFT·F3 (MODEL 4 and ESOTERIC), SHIFT·F3 (MAX-80), and disregards the CAPS mode.

Key Codes MAX-80, Model 4, and ESOTERIC

DEC	HEX	KEY(s)
176	B0	ALTCTRL·0 or ALT·p
177	B1	ALTCTRL·1 or ALT·q
178	B2	ALTCTRL·2 or ALT·r
179	B3	ALTCTRL·3 or ALT·s
180	B4	ALTCTRL·4 or ALT·t
181	B5	ALTCTRL·5 or ALT·u
182	B6	ALTCTRL·6 or ALT·v
183	B7	ALTCTRL·7 or ALT·w
184	B8	ALTCTRL·8 or ALT·x
185	B9	ALTCTRL·9 or ALT·y
186	BA	ALTCTRL·: or ALT·z
187	BB	ALTCTRL·;
188	BC	ALTCTRL·<
189	BD	ALTCTRL·=
190	BE	ALTCTRL·>
191	BF	ALTCTRL·?
192	C0	ALTCTRL·SHIFT·@
193	C1	ALTCTRL·SHIFT·A
194	C2	ALTCTRL·SHIFT·B
195	C3	ALTCTRL·SHIFT·C
196	C4	ALTCTRL·SHIFT·D
197	C5	ALTCTRL·SHIFT·E
198	C6	ALTCTRL·SHIFT·F
199	C7	ALTCTRL·SHIFT·G
200	C8	ALTCTRL·SHIFT·H
201	C9	ALTCTRL·SHIFT·I
202	CA	ALTCTRL·SHIFT·J
203	CB	ALTCTRL·SHIFT·K
204	CC	ALTCTRL·SHIFT·L
205	CD	ALTCTRL·SHIFT·M
206	CE	ALTCTRL·SHIFT·N
207	CF	ALTCTRL·SHIFT·O
208	D0	ALTCTRL·SHIFT·P
209	D1	ALTCTRL·SHIFT·Q
210	D2	ALTCTRL·SHIFT·R
211	D3	ALTCTRL·SHIFT·S
212	D4	ALTCTRL·SHIFT·T
213	D5	ALTCTRL·SHIFT·U
214	D6	ALTCTRL·SHIFT·V
215	D7	ALTCTRL·SHIFT·W
216	D8	ALTCTRL·SHIFT·X
217	D9	ALTCTRL·SHIFT·Y
218	DA	ALTCTRL·SHIFT·Z
219	DB	ALTCTRL·↑ or ALTCTRL·CTRL·+
220	DC	ALTCTRL·CTRL·<
221	DD	ALTCTRL·CTRL·=
222	DE	ALTCTRL·CTRL·>
223	DF	ALTCTRL·CTRL·?

Key Codes MAX-80, Model 4, and ESOTERIC

DEC	HEX	KEY(s)
224	E0	ALTCTRL·@
225	E1	ALTCTRL·a
226	E2	ALTCTRL·b
227	E3	ALTCTRL·c
228	E4	ALTCTRL·d
229	E5	ALTCTRL·e
230	E6	ALTCTRL·f
231	E7	ALTCTRL·g
232	E8	ALTCTRL·h
233	E9	ALTCTRL·i
234	EA	ALTCTRL·j
235	EB	ALTCTRL·k
236	EC	ALTCTRL·l
237	ED	ALTCTRL·m
238	EE	ALTCTRL·n
239	EF	ALTCTRL·o
240	F0	ALTCTRL·p
241	F1	ALTCTRL·q
242	F2	ALTCTRL·r
243	F3	ALTCTRL·s
244	F4	ALTCTRL·t
245	F5	ALTCTRL·u
246	F6	ALTCTRL·v
247	F7	ALTCTRL·w
248	F8	ALTCTRL·x
249	F9	ALTCTRL·y
250	FA	ALTCTRL·z
251	FB	ALTCTRL·CTRL·; (MAX-80 ALTCTRL·{)
252	FC	ALTCTRL·CTRL·, (MAX-80 ALTCTRL·|)
253	FD	ALTCTRL·CTRL·- (MAX-80 ALTCTRL·})
254	FE	ALTCTRL·CTRL·. (MAX-80 ALTCTRL·~)
255	FF	ALTCTRL·CTRL·/ (MAX-80 ALTCTRL·_)

Sample MAX-80, Model 4, and ESOTERIC SUPERBASIC Program

```
10000 '"SQRSUM/BAS
10100 CLS
10200 CLEAR 600
10300 DEFSTR B
10400 DEFINT C - Y
10500 RR = &F800
10600 IF PEEK(3) = 23 THEN RR = &3C00:IF PEEK(4) = 64 THEN CMD "V80"
10700 B0 = CHR$(017) + CHR$(066) + CHR$(0) + CHR$(025) + CHR$(229) + CHR$(058) +
       CHR$(003) + CHR$(0) + CHR$(254) + CHR$(023) + CHR$(062) + CHR$(004) + CHR
      $(196) + CHR$(094) + CHR$(008) + CHR$(204) + CHR$(213) + CHR$(068) + CHR$(
      225) + CHR$(243)
10800 B1 = CHR$(024) + CHR$(048) + CHR$(017) + CHR$(045) + CHR$(0) + CHR$(025) +
       CHR$(229) + CHR$(058) + CHR$(003) + CHR$(0) + CHR$(254) + CHR$(023) + CHR
      $(196) + CHR$(009) + CHR$(027) + CHR$(204) + CHR$(009) + CHR$(009) + CHR$(
      227) + CHR$(017)
10900 B2 = CHR$(024) + CHR$(0) + CHR$(235) + CHR$(025) + CHR$(193) + CHR$(113) +
       CHR$(035) + CHR$(112) + CHR$(058) + CHR$(003) + CHR$(0) + CHR$(254) + CHR
      $(023) + CHR$(062) + CHR$(004) + CHR$(194) + CHR$(113) + CHR$(008) + CHR$(
      195) + CHR$(210)
11000 B3 = CHR$(068) + CHR$(0) + CHR$(049) + CHR$(050) + CHR$(051) + CHR$(052) +
       CHR$(010) + CHR$(043) + CHR$(053) + CHR$(192) + CHR$(054) + CHR$(010) + C
      HR$(001) + CHR$(251) + CHR$(255) + CHR$(009) + CHR$(229) + CHR$(221) + CHR
      $(225) + CHR$(058)
11100 B4 = CHR$(003) + CHR$(0) + CHR$(254) + CHR$(023) + CHR$(032) + CHR$(004) +
       CHR$(058) + CHR$(090) + CHR$(0) + CHR$(245) + CHR$(017) + CHR$(0) + CHR$(
      0) + CHR$(001) + CHR$(032) + CHR$(004) + CHR$(126) + CHR$(035) + CHR$(007)
       + CHR$(056)
11200 B5 = CHR$(056) + CHR$(032) + CHR$(028) + CHR$(235) + CHR$(229) + CHR$(205)
       + CHR$(017) + CHR$(066) + CHR$(126) + CHR$(113) + CHR$(018) + CHR$(225) +
       CHR$(035) + CHR$(019) + CHR$(016) + CHR$(244) + CHR$(005) + CHR$(221) + C
      HR$(112) + CHR$(0)
11300 B6 = CHR$(241) + CHR$(050) + CHR$(090) + CHR$(0) + CHR$(211) + CHR$(132) +
       CHR$(042) + CHR$(072) + CHR$(064) + CHR$(024) + CHR$(022) + CHR$(229) + C
      HR$(205) + CHR$(245) + CHR$(008) + CHR$(225) + CHR$(119) + CHR$(121) + CHR
      $(229) + CHR$(205)
11400 B7 = CHR$(250) + CHR$(008) + CHR$(225) + CHR$(035) + CHR$(016) + CHR$(241)
       + CHR$(005) + CHR$(221) + CHR$(112) + CHR$(0) + CHR$(042) + CHR$(083) + C
      HR$(018) + CHR$(030) + CHR$(001) + CHR$(123) + CHR$(233) + CHR$(032) + CHR
      $(022) + CHR$(235)
11500 B8 = CHR$(229) + CHR$(205) + CHR$(017) + CHR$(066) + CHR$(026) + CHR$(119)
       + CHR$(225) + CHR$(035) + CHR$(019) + CHR$(016) + CHR$(245) + CHR$(221) +
       CHR$(112) + CHR$(0) + CHR$(241) + CHR$(050) + CHR$(090) + CHR$(0) + CHR$(
      211) + CHR$(132)
11600 B9 = CHR$(201) + CHR$(126) + CHR$(229) + CHR$(205) + CHR$(250) + CHR$(008)
       + CHR$(225) + CHR$(035) + CHR$(016) + CHR$(247) + CHR$(221) + CHR$(112) +
       CHR$(0) + CHR$(201)
11700 BB = B0 + B1 + B2 + B3 + B4 + B5 + B6 + B7 + B8 + B9
11800 LABEL "Length"
11900 B = "0123456789"
12000 N = 0
12100 L = 0
12200 ON STOP GOTO 0
12300 INPUT !0, 0, #1, 95, USING B, "Number of squares per side (3-9) ";N
12400 IF N < 3 OR N > 9 THEN "Length"
12500 DIM H(N), V(N), M(N * N), I(N * N)
12600 LABEL "Draw"
12700 CLS
12800 S = 2 * INT(67 / N)
12900 BU = "####"
13000 A = INT(9 / N)
```

Sample MAX-80, Model 4, and ESOTERIC SUPERBASIC Program

```
13100 XL = S * N + 8
13200 'Horizontal lines
13300 FOR X = 10 TO XL - 1
13400 FOR Y = 3 TO A * 6 * N + 3 STEP A * 6
13500 SET(X, Y)
13600 NEXT
13700 NEXT
13800 'Vertical lines
13900 FOR X = 8 TO XL STEP S
14000 FOR Y = 3 TO A * 6 * N + 3
14100 SET(X, Y)
14200 SET(X + 1, Y)
14300 NEXT
14400 NEXT
14500 'Rows
14600 FOR Y = 1 TO N
14700 PRINT !Y * S / 2 + INT(4 - S / 4) - 2, 0, USING BU;Y;
14800 NEXT
14900 'Columns
15000 FOR X = 1 TO N
15100 PRINT !0, X * 2 * A - A + 1, USING "##";X;
15200 NEXT
15300 IF Q = 11 THEN GOSUB "MATRIX"
15400 LABEL "Column"
15500 P = 1
15600 DI = 1
15700 PRINT !0, 22, CHR$(030);
15800 LINE INPUT !0, 22, %Q, LEN(STR$(N)) - 1, 95, USING MID$(B, 4 - LEN(STR$(N)
      ), N), "Column (X) ";BX
15900 IF Q = 11 THEN "Draw"
16000 IF Q = 2 THEN "Done"
16100 X = VAL(BX)
16200 IF X > N OR X = 0 THEN "Column"
16300 U = POS(0) + LEN(BX) - LEN(STR$(N)) + 1
16400 LABEL "Row"
16500 PRINT !U, 22, CHR$(030);
16600 LINE INPUT !U, 22, %Q, LEN(STR$(N)) - 1, 95, USING MID$(B, 4 - LEN(STR$(N)
      ), N), ", Row (Y) ";BY
16700 IF Q = 8 THEN "Column"
16800 IF Q = 2 THEN "Done"
16900 Y = VAL(BY)
17000 IF Y > N OR Y = 0 THEN "Row"
17100 E = POS(0) + LEN(BY) - LEN(STR$(N)) + 1
17200 GOSUB "MSPOT"
17300 LABEL "Over"
17400 Q = FRE(B)
17500 DEF USR = WPEEK(VARPTR(BB) + 1) + LEN(B0) + 2
17600 ON STOP GOTO "No beep"
17700 Q = USR (RR + X * S / 2 + INT(4 - S / 4) - 2 + INT(Y * A * 2 - A + 1) * 80
      )
17800 LABEL "Value"
17900 PRINT !E, 22, ", Value ";STRING$(4, 95);
18000 INPUT !E + 8, 22, %Q, 1, 95, USING B + "-", BV
18100 IF Q = 13 THEN PRINT !E, 22, ", Value ";STRING$(4, 32);:GOTO "No beep"
18200 IF Q = 8 OR Q = 2 THEN "No beep"'if 8 back to row
18300 IF Q > 0 AND Q < 32 THEN "Value"
18400 BM = ""
18500 LINE INPUT !E + 9, 22, %Q, 3, 95, USING B, BM
18600 IF Q = 8 THEN "Value"
18700 IF Q <> 13 THEN IF Q > 2 AND Q < 32 THEN "Value"
```

```
18800 IF BV = "-" THEN M((X - 1) * N + Y) = - VAL(BM) ELSE M((X - 1) * N + Y) =
      VAL(BV) * 10 [ LEN(BM) + VAL(BM)
18900 LABEL "No beep"
19000 MN = LEN(B0) + LEN(B1) + LEN(B2) + LEN(B3) + LEN(B4)
19100 MID$(BB, MN, 1) = CHR$(024)
19200 CALL WPEEK(VARPTR(BB) + 1)
19300 MID$(BB, MN, 1) = CHR$(056)
19400 ON STOP GOTO 0
19500 IF Q = 8 AND K => 0 THEN "Row"
19600 LABEL "Done"
19700 IF Q = 2 THEN P = 2
19800 PRINT !0, 23, CHR$(030);
19900 ON P GOSUB "Partial Total", "USER"
20000 FOR K = 1 TO N - 1
20100 IF H(K) <> H(K + 1) OR V(K) <> V(K + 1) THEN EXIT "Column"
20200 NEXT
20300 IF LL <> LR OR LL <> H(K) THEN "Column"
20400 IF P = 2 THEN "Good"
20500 PRINT !0, 22, "Sorting... " CHR$(030);
20600 SORT N * N, * I(1), M(1)
20700 PRINT "Checking ..."
20800 FOR U = 1 TO N * N - 1
20900 IF M(I(U)) = M(I(U + 1)) THEN "Fail"
21000 NEXT
21100 LABEL "Good"
21200 PRINT !0, 23, "Good. <ENTER> = new values, <CLEAR> = size select, or <E> =
       end.";
21300 ON STOP GOTO "END"
21400 LABEL "Wait"
21500 BW = INKEY$
21600 IF BW = CHR$(013) THEN "Column"
21700 IF BW = "E" OR BW = "e" THEN "END"
21800 IF BW <> CHR$(031) THEN "Wait"
21900 CLS
22000 ERASE H, V, M, I
22100 GOTO "Length"
22200 LABEL "Partial Total"
22300 'initialize row and column
22400 H(Y) = 0
22500 V(X) = 0
22600 'calculate row and column
22700 FOR J = 1 TO N
22800 H(Y) = H(Y) + M((J - 1) * N + Y)
22900 V(X) = V(X) + M((X - 1) * N + J)
23000 NEXT
23100 GOSUB "MSPOT"
23200 IF H(Y) < - 999 OR H(Y) > 9999 THEN K = - 1:GOTO "Over"
23300 GOSUB "SIH"
23400 IF V(X) < - 999 OR V(X) > 9999 THEN K = - 1:GOTO "Over"
23500 GOSUB "SIV"
23600 LABEL "Diagonals"
23700 IF P = 2 AND DI = 2 THEN RETURN
23800 'initialize diagonals
23900 LR = 0
24000 LL = 0
24100 'calculate diagonals
24200 FOR F = 1 TO N
24300 LR = LR + M((F - 1) * N + F)
24400 LL = LL + M((N - F) * N + F)
24500 NEXT
```

```
24600 DI = 2
24700 PRINT !XL / 2, A * N * 2 + 2, USING BU;LR;
24800 PRINT !0, A * N * 2 + 2, USING BU;LL;
24900 RETURN
25000 LABEL "SIH"
25100 PRINT !XL / 2 + 1, Y * A * 2 - A + 1, USING "#####";H(Y);
25200 RETURN
25300 LABEL "MSPOT"
25400 PRINT !X * S / 2 + INT(4 - S / 4) - 2, Y * A * 2 - A + 1, USING BU;M((X -
      1) * N + Y);
25500 RETURN
25600 LABEL "SIV"
25700 PRINT !X * S / 2 + INT(4 - S / 4) - 2, A * N * 2 + 2, USING BU;V(X);
25800 RETURN
25900 LABEL "USER"
26000 PRINT !0, 22, CHR$(030)
26100 LINE INPUT !0, 23, #4, 95, USING B + "-", "Starting integer ";BS
26200 U = POS(0) + LEN(BS) - 4
26300 L = VAL(BS)
26400 IF L = 0 THEN BS = "0":PRINT !17, 23, BS;:U = POS(0)
26500 LINE INPUT !U, 23, %Q, 3, 95, USING B + "-", ", Increment ";BI
26600 R = VAL(BI)
26700 IF R = 0 THEN "USER"
26800 IF Q = 8 THEN "USER"
26900 IF N * N * R + L - R > 9999 THEN "USER"'maximum value
27000 Z = (N * N * N * R + N * R + 2 * N * (L - R)) / 2
27100 IF Z < - 999 OR Z > 9999 THEN "USER"'total values
27200 G = N / 2
27300 IF G = N / 2 THEN "EVEN"
27400 Y = 1
27500 O = L
27600 FOR T = 1 TO N
27700 FOR W = 1 TO N
27800 X = W + G
27900 IF X > N THEN X = X - N
28000 M((X - 1) * N + Y) = O
28100 PRINT !X * S / 2 + INT(4 - S / 4) - 2, Y * A * 2 - A + 1, USING BU;O;
28200 Y = Y - 1
28300 IF Y = 0 THEN Y = N
28400 O = O + R
28500 NEXT
28600 G = G - 1
28700 IF G < 0 THEN G = G + N
28800 Y = Y + 2
28900 IF Y > N THEN Y = Y - N
29000 NEXT
29100 GOTO "TOTALS"
29200 LABEL "EVEN"
29300 C = INT(N / 4) = N / 4
29400 O = N * N * R + L - R
29500 FOR Y = 1 TO N
29600 FOR X = 1 TO N
29700 E = 0
29800 IF ABS(N - 2 * Y + 1) = N / 2 THEN E = X:X = N - X + 1
29900 M((X - 1) * N + Y) = O
30000 O = O - R
30100 IF E THEN X = E
30200 NEXT
30300 NEXT
30400 GOSUB "MATRIX"
```

```
30500 FOR Y = 1 TO N / 2
30600 IF Y AND 1 THEN FOR W = N / 2 TO 1 STEP - 1 ELSE FOR W = 1 TO N
30700 IF Y = W OR Y + W - 1 = N THEN "Skip COL"
30800 GOSUB "ROW swap"
30900 GOSUB "ROW swap"
31000 LABEL "Skip COL"
31100 NEXT
31200 NEXT
31300 D = N
31400 IF C THEN D = N / 2
31500 FOR X = 1 TO N / 2
31600 FOR W = 1 TO D
31700 IF X = W OR X + W - 1 = N THEN "Skip ROW"
31800 GOSUB "COL swap"
31900 IF C THEN GOSUB "COL swap"
32000 LABEL "Skip ROW"
32100 NEXT
32200 IF V(X) <> Z THEN GOSUB "Adjust COL"
32300 NEXT
32400 RETURN
32500 LABEL "MATRIX"
32600 FOR X = 1 TO N
32700 FOR Y = 1 TO N
32800 GOSUB "MSPOT"
32900 NEXT
33000 NEXT
33100 LABEL "TOTALS"
33200 ZERO H, V
33300 LL = 0
33400 LR = 0
33500 FOR Y = 1 TO N
33600 FOR X = 1 TO N
33700 H(Y) = H(Y) + M((X - 1) * N + Y)
33800 V(X) = V(X) + M((X - 1) * N + Y)
33900 IF Y = N THEN GOSUB "SIV"
34000 NEXT
34100 GOSUB "SIH"
34200 NEXT
34300 GOTO "Diagonals"
34400 LABEL "ROW swap"
34500 X = W
34600 E = M((X - 1) * N + N - Y + 1) - M((X - 1) * N + Y)
34700 IF H(Y) <> Z THEN T = M((X - 1) * N + Y):M((X - 1) * N + Y) = M((X - 1) *
      N + N - Y + 1):GOSUB "Partial Total":M((X - 1) * N + N - Y + 1) = T:T = Y:
      Y = N - Y + 1:GOSUB "Partial Total":Y = T
34800 W = N - W + 1
34900 RETURN
35000 LABEL "COL swap"
35100 Y = W
35200 E = M((N - X) * N + Y) - M((X - 1) * N + Y)
35300 IF V(X) >= Z THEN IF V(X) + E < Z OR E > 0 THEN "Skip swap"
35400 IF V(X) < Z THEN IF V(X) + E > Z OR E < 0 THEN "Skip swap"
35500 LABEL "DOIT"
35600 T = M((X - 1) * N + Y)
35700 M((X - 1) * N + Y) = M((N - X) * N + Y)
35800 GOSUB "Partial Total"
35900 M((N - X) * N + Y) = T
36000 T = X
36100 X = N - X + 1
36200 GOSUB "Partial Total"
```

```
36300 X = T
36400 LABEL "Skip swap"
36500 IF C THEN W = N - W + 1
36600 RETURN
36700 LABEL "Adjust COL"
36800 FOR Y = 1 TO D - 3
36900 IF X = Y OR X + Y - 1 = N THEN "Skip it"
37000 E = M((N - X) * N + Y) - M((X - 1) * N + Y)
37100 F = M((N - X) * N + Y + 1) - M((X - 1) * N + Y + 1)
37200 G = M((N - X) * N + Y + 2) - M((X - 1) * N + Y + 2)
37300 J = M((N - X) * N + Y + 3) - M((X - 1) * N + Y + 3)
37400 IF E + F + V(X) = Z THEN GOSUB "DOIT":GOTO "UP1"
37500 IF E + G + V(X) = Z THEN GOSUB "DOIT":GOTO "UP2"
37600 IF E + J + V(X) = Z THEN GOSUB "DOIT":GOTO "UP3"
37700 IF F + G + V(X) = Z THEN GOSUB "UP1":GOTO "UP1"
37800 IF F + J + V(X) = Z THEN GOSUB "UP1":GOTO "UP2"
37900 IF G + J + V(X) = Z THEN GOSUB "UP2":GOTO "UP1"
38000 IF E + F + G + V(X) = Z THEN GOSUB "DOIT":GOSUB "UP1":GOTO "UP1"
38100 IF E + F + J + V(X) = Z THEN GOSUB "DOIT":GOSUB "UP1":GOTO "UP2"
38200 IF E + G + J + V(X) = Z THEN GOSUB "DOIT":GOSUB "UP2":GOTO "UP1"
38300 IF F + G + J + V(X) = Z THEN GOSUB "UP1":GOSUB "UP1":GOTO "UP1"
38400 LABEL "Skip it"
38500 NEXT
38600 RETURN
38700 LABEL "UP1"
38800 Y = Y + 1
38900 GOTO "DOIT"
39000 LABEL "UP2"
39100 Y = Y + 2
39200 GOTO "DOIT"
39300 LABEL "UP3"
39400 Y = Y + 3
39500 GOTO "DOIT"
39600 LABEL "Fail"
39700 K = I(U) / N - (I(U) / N > INT(I(U) / N))
39800 J = I(U) - N * (K - 1)
39900 W = I(U + 1) / N - (I(U + 1) / N > INT(I(U + 1) / N))
40000 T = I(U + 1) - N * (W - 1)
40100 PRINT !0, 23, "Locations (" RIGHT$(STR$(K), LEN(STR$(K)) - 1)"," RIGHT$(ST
      R$(J), LEN(STR$(J)) - 1)") and (" RIGHT$(STR$(W), LEN(STR$(W)) - 1)"," RIG
      HT$(STR$(T), LEN(STR$(T)) - 1)") are both"M(I(U));
40200 EXIT "Column"
40300 LABEL "END"
40400 ON STOP GOTO 0
40500 CLS
```