MULTIDOS

A Versatile Disk Operating System for TRS-80 Model I, III, and 4

AlphaBit Communications, Inc.

13349 MICHIGAN AVE., DEARBORN, MI 48126 (313) 581-2896

NOTICE TO MULTIDOS USERS

This software is sold on an "as is" basis. Neither Cosmopolitan Electronics Corporation nor AlphaBit Communications, Inc. shall not be liable or responsible to the purchaser with respect to liability, loss, or damage caused or alleged to be caused directly or indirectly by the use of this software, which includes but is not limited to any interruption of service, loss of business, c anticipatory profits, or consequential damage resulting from use of this software.

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

DBLDOS (tm)	Percom Data Co.
DOSPLUS (tm)	Micro Systems Software.
LDOS (tm)	Logical Systems Inc.
NEWDOS (tm)	Apparat, Inc.
TRS-80 (tm)	Radio Shack, a division of Tandy Corp.
TRSDOS (tm)	Radio Shack, a division of Tandy Corp.
Z80 (tm)	Zilog, Inc.

PROGRAM AND MANUAL BY VERNON B. HESTER

MULTIDOS and all Cosmopolitan Electronics Corporation products are distributed exclusively by AlphaBit Communications, Inc. Alphabit personnel will answer questions about Multidos to the best of their ability. Requests for detailed technical information will be passed on to the author of MULTIDOS, Vernon Hester, to be dealt with on an as-time-permits basis. Any comments or requests for information should be directed to:

AlphaBit Communications, Inc.
13349 Michigan Ave.
Dearborn, MI 48126
(313) 581-2896

MAKING A BACKUP

The first thing you should do with the MULTIDOS master diskette is make a backup. You will need a blank diskette to hold the backup copy. Be sure you do not have a write protect tab on the backup diskette.

LEAVE THE WRITE PROTECT TAB ON THE MULTIDOS MASTER DISKETTE.

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

You will see the MULTIDOS banner, and the date prompt. Enter the date and press <ENTER>.

Type the word BACKUP, then press the <ENTER> key. You'll get a series of prompts.

'WHICH DRIVE CONTAINS THE SOURCE DISKETTE?'
Answer with 0.

'WHICH DRIVE FOR THE DESTINATION DISKETTE?'
Answer with 1 if two drives are available (each having the same number of tracks), otherwise answer with 0.

'PRESS "ENTER" WHEN THE SOURCE DISKETTE IS IN DRIVE 0'.
Press <ENTER>.

At this point MULTIDOS will examine the source diskette for track count and density. The track count will be 35 for the MODEL I, and 40 for the MODEL III, MAX-80, or MODEL 4. The density will be the one ordered for the MODEL I, and double for the MODEL III, MAX-80, or MODEL 4.

Answer 'TRACK COUNT FOR THE DESTINATION DISKETTE (35 to 96)?' with <ENTER>

MULTIDOS will format the blank diskette, then duplicate the contents of the MULTIDOS master diskette.

If only one drive is available, BACKUP will tell you when to insert the blank (destination) diskette, and when to re-insert the MULTIDOS master (source) diskette. During a one drive BACKUP, you will have to change diskettes from time to time.

When BACKUP has completely duplicated the MULTIDOS master diskette,
'COMPLETED
will be displayed. Press <ENTER>.

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

CONTENTS

BACKING UP MULTIDOS
WHAT IS MULTIDOS 1
MANUAL NOTATION 2
MULTIDOS COMMANDS 2
SPECIAL UNIVERSAL COMMANDS 6
MULTIDOS SYSTEM DISKETTES 9

LIBRARY COMMANDS

APPEND	12	HELP	36
ATTRIB	13	KEYBD	37
AUTO	14	KILL	37
BLINK	16	LIB	37
BOOT	16	LINK	38
BUILD	16	LIST	38
CLEAR	19	LOAD	39
CLOCK	19	PATCH	39
CLRDSK	19	PRINT	40
CLS	19	PROT	40
CONFIG	20	RENAME	41
DATE	26	RESET	41
DDAM	26	RESTOR	41
DEAD	26	ROUTE	41
DEBUG	27	SCREEN	42
DEVICE	31	SETCOM	42
DIR	31	SKIP	42
DO	34	TIME	42
DUMP	34	TOPMEM	43
FORMS	35	V64	43
FREE	36	V80	43

SYSTEM UTILITIES

BACKUP/CMD	45	HELP/CMD	53
CAT/CMD	47	LO/CMD	54
CDIR/CMD	48	MEM/CMD	55
CONVERT/CMD	48	MEMDISK/CMD	55
COPY/CMD	48	PRT/CMD	56
DBLFIX	50	RS/CMD	57
DDT/CMD	50	SPOOL/CMD	58
FMAP/CMD	50	TAPE/CMD	59
FORMAT/CMD	51	VFU/CMD	60
GR/CMD	53	ZAP/CMD	64

DIRECTORY STRUCTURE 67 ERROR CODES 73

SUPERBASIC

BASIC	75	CMD"P"	84
BASIC *	75	CMD"Q"	85
BASIC !	76	CMD"R"	86
BASIC #	76	CMD"S"	86
BBASIC	76	CMD"T"	86
SINGLE STEPPING	77	CMD"U"	86
BREAK POINTS	78	CMD"V"	86
SELECTING VARIABLES	79	CMD"W"	86
REVIEWING VARIABLES	80	CMD"X"	87
STACKING PROGRAMS	80	CMD"uuuuu"	87
SINGLE KEYSTROKE	81	DEF FN	87
SINGLE LETTER	82	DEFUSR	88
&H and &O	83	INSTR	88
CMD"C"	83	LINE INPUT	89
CMD"D"	83	LIST	89
CMD"E"	83	MID$=	89
CMD"K"	83	TIME$	89
CMD"L"	84	USRn	90

SUPERBASIC OVERLAY FUNCTIONS

FIND	90	REFERENCE	95
GLOBAL EDIT	91	RENUMBER	96

SUPERBASIC FILE MANIPULATION

KILL	98	NAME	98
LOAD	98	RUN	99
MERGE	98	SAVE	99

SUPERBASIC FILE ACCESS

OPEN	100	CVS	102
CLOSE	100	CVD	102
INPUT#	100	LSET	102
LINEINPUT#	100	RSET	102
PRINT#	101	PUT	103
FIELD	101	GET	104
MKI$	102	EOF	104
MKS$	102	LOC	104
MKD$	102	LOF	104
CVI	102	ERROR CODES	105

TECHNICAL

RAM ADDRESSES	107	DRIVE CONTROL TABLE	115
FILE CONTROL BLOCK	110	OPTIONAL ZAPS	119

MULTIDOS

MULTIDOS is a TRS-80 disk operating system developed to provide a means of communicating with the other disk operating systems currently on the market. In addition, MULTIDOS contains major enhancements which make the program easy to use.

The word "TYPE" in the chart below classifies all known 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.
2) The use of psuedo-logical tracks. "P" indicated psuedo-logical tracks. "T" indicates true logical tracks.
3) Track zero formatted in a different density than the balance of the diskette. "Z" indicates the same density. "X" indicates a different density.
4) The lowest sector number assigned to each track. "0" = Numbered zero. "1" = Numbered one.

				MULTIDOS MODEL I		MODEL III/4 MAX-80	
MOD.	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	NEWDOS80-1	SINGLE	UTZ0	OK	NOTE 1	NOTE 2	NOTE 2
I	DOSPLUS	SINGLE	UTZ0	OK	OK	NOTE 2	NOTE 2
I	NEWDOS80-2	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	NEWDOS80-1	DOUBLE	DPX0	OK	OK	OK	OK
I	NEWDOS80-2	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	DTZ1	VFU	NO	VFU	NO
III	DOSPLUS	DOUBLE	DTZ0	OK	OK	OK	OK
III	LDOS	DOUBLE	DTZ0	OK	OK	OK	OK
III	NEWDOS80-2	DOUBLE	DPZ0	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 can copy a file FROM these types.
NOTE 1: These types will have their address marks changed.
NOTE 2 = Requires CONVERT/CMD to alter the directory address marks.

MULTIDOS MANUAL NOTATION

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

Symbol	Meaning
<ENTER>	Press the "ENTER" key.
<BREAK>	Press the "BREAK" key.
<SPACE>	Press the "SPACE-BAR".
<COMMA>	Press the "," key.
<LF>	A linefeed character, the down-arrow.
Filespec	A valid MULTIDOS file.
Drivespec	A particular logical disk drive number (0 to 7)
[ ]	Brackets enclose optional parameters. The "[" and "]" are not keyed in.
{ }	Braces enclose manual remarks.
A space	At least one mandatory space.
Punctuation	Must be entered as shown.
Single quote '	Encloses MULTIDOS responses shown on the video monitor.
...	Ellipses represent the repetition of an optional parameter.

COMMANDS

Whenever the command prompt,

```
MULTIDOS
_
```

Is displayed, the user may enter a command followed by <ENTER>.

EXAMPLE:

```
FREE<ENTER>
```

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

The maximum length of a command is 63 characters for the MODEL I and MODEL III, and 79 characters for the MAX-80 and MODEL 4. All commands must terminate with <ENTER>.

Commands may be entered in any combination of upper and lower case. The command interpreter performs a case conversion to all upper case unless a special bit is set in a control byte, (SM0), described in the technical section.

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

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

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 or numeric (A-Z, and/or 0-9).

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

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

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

NOTE: A filespec does NOT have any spaces on it. Spaces are treated as separators by MULTIDOS. If a filespec is spaced, only the contents preceding the space will be interpreted as the filespec.

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

WONDER.BOY WONDER.WOMAN WONDER.FULL

however, the following are unique filespecs:

WONDER:1 WONDER:2 WONDER:3

because the drivespecs are different.

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.
/CIM	A file created by the DUMP library command.
/CMD	A machine language file executable directly from DOS.
/REL	A relocatable object file.
/SYS	An operating system's overlay.
/TXT	An ASCII text file. (Typically sequential files)

MULTIDOS

MULTIDOS will insert the extension /CMD, if a filespec is entered without an extension.

EXAMPLE:

BACKUP<ENTER>

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

To execute a file without an extension, append just the "/" and MULTIDOS will load and execute the file without adding the default extension "/CMD".

EXAMPLE:

CREATE/<ENTER>

MULTIDOS will load and execute the file CREATE.

If a drivespec is not assigned, MULTIDOS will search 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 will search for the first active drive with the disk write enabled to initialize the file.

EXAMPLE:

PROG:2<ENTER>

MULTIDOS will look for the program PROG/CMD on logical drive two.

EXAMPLE:

PROG/<ENTER>

MULTIDOS will search the drives, in logical ascending order, then load and execute 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 will be loaded and executed.

EXAMPLE:

PROG/RAM<ENTER>

MULTIDOS will search the drives, in logical ascending order, then load and execute 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 will be loaded and executed. If the user wants to load and execute the file PROG/RAM on logical drive 4, then "PROG/RAM:4<ENTER>" must be entered on the command line.

More on COMMANDS

Multiple DOS commands will be executed if they are separated by commas immediately behind the commands.

EXAMPLE:

DIR 1, BASIC<ENTER>

1. Displays the directory on logical drive one.
2. Loads and executes SUPERBASIC.

The library command HELP (MAX-80, and MODEL 4), or the file HELP/CMD (MODEL I, and MODEL III) overrides any inquiry when a comma follows the command help is requesting.

EXAMPLE:

HELP FORMS,DIR<ENTER>

This will be interpreted as if only HELP<ENTER> was entered.

MULTIDOS will repeat the last DOS command if <ENTER> is the first response to the command prompt.

EXAMPLE:

DIR :1<ENTER>

After the command prompt is displayed, pressing <ENTER> as the first keystroke will again display the directory of the disk in logical drive 1. The diskettes can be changed between the <ENTER>'s.
To erase the last DOS command press the <BREAK> key.

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.

The use of the RESET library command will disable any multiple DOS command(s) following RESET and erase the last DOS command.

EXAMPLE:

RESET,DIR 1<ENTER>

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

MULTIDOS provides an easy method to obtain a directory by keying in the numbers 0 to 7 as the first character in the command promt. (The MODEL I and MODEL III do not allow backspacing. - Press <BREAK> to restart.)

SPECIAL UNIVERSAL COMMANDS

I - minidos

This feature will permit the user to interrupt most programs while executing and execute several selected commands then return to the main program without any changes in ıe main program's state during this interruption. Minidos will be activated if the following conditions are met:

1. Interrupt service is active.
2. Simultaneous depression of the ":" and ";" keys. - MODEL I/MODEL III
   Simultaneous depression of the "D", "F", and "G" keys. - MAX-80/MODEL 4.

To exit minidos:

1. Press <BREAK> or
2. Key in "!" As the first input character.

However, upon exit an extraneous character might be present.

Minidos commands require only one letter and do not require a space terminator. However, spaces may be keyed in after the command letter.

EXAMPLE:

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

The commands available are:

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

"D", DIR - Display the directory of the specified drivespec.

"K", KILL - Remove the specified filespec.

"L", LIST - List the specified filespec.

"M", - DEBUG. ***** MUST NOT BE INVOKED UNTIL minidos IS EXITED.

"P", - Send specified decimal character to the printer. (MODEL I/III)

"V", - Flip the video format between 64X16 and 80X24. (MAX-80/MODEL 4)

I.1 - minidos COPY - Copies files, <C>.

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

I.2 - minidos DIRECTORY - Display disk directory, <D>.

Dd, DId, DSd, D d, DI:d, or DS:d<ENTER>

D = 0 to 7 for the logical drive number.
If "d" is not entered, then zero is used.
I = display files with an invisible attribute.
S = display files with an invisible or system attribute.

I.3 - minidos KILL - Delete file, <K>.

K filespec<ENTER>

I.4 - minidos LIST - List a file to the display, <L>.

L filespec<ENTER>

The listing will be suspended if the <SPACE-BAR> is held down, or terminated if <BREAK> is pressed.

I.5 - Activate DEBUG, <M>.

M<ENTER>

minidos does not flag the user anything has happened. However, DEBUG should not be invoked until an exit from minidos is made. Minidos and DEBUG have totally different exit routes. These different exit routes correctly return to the command in process prior to minidos or DEBUG being invoked. If DEBUG is invoked during minidos, an exit from DEBUG will attempt to use minidos' exit route and will hang the system.

I.6 - minidos PRINT - Send a decimal byte to the printer DCB, <P>.

Pnn<ENTER>

nn must be in the range of 0 to 99. The main purpose of this command is send control bytes, 0 to 31, to the printer without leaving the main program in progress. (This command is not on the MAX-80 or MODEL 4.)

I.7 - Flip video format; <V>. (MAX-80/MODEL 4)

V<ENTER>

This command will clear the display then flip the video between the 64X16 format and the 80X24 format. If the current video format is 80X24, then "V<ENTER" will change the display to 64X16. If the current display format is 64X16, then "V<ENTER" will change the display to 80X24.

II - <JKL> Dump the screen display to the printer.

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

This function sends the current contents of the display to the device defined in the printer DCB. Graphic blocks are converted to the "." character. If graphic blocks are nc to be converted to the "." character then the <HJK> function, described below, should be used.

III - <HJK> Dump the screen display to the printer, including graphics.

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

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

The ASCII Codes for 0 to 95

Dec.	Hex	Char.	Dec.	Hex	Char.	Dec.	Hex	Char.
0	00	NUL	32	20	SPACE	64	40	@
1	01	SOH	33	21	!	65	41	A
2	02	STX	34	22	"	66	42	B
3	03	ETX	35	23	#	67	43	C
4	04	EOT	36	24	$	68	44	D
5	05	ENQ	37	25	%	69	45	E
6	06	ACK	38	26	&	70	46	F
7	07	BEL	39	27	'	71	47	G
8	08	BS	40	28	(	72	48	H
9	09	HT	41	29	)	73	49	I
10	0A	LF	42	2A	*	74	4A	J
11	0B	VT	43	2B	+	75	4B	K
12	0C	FF	44	2C	,	76	4C	L
13	0D	CR	45	2D	-	77	4D	M
14	0E	SO	46	2E	.	78	4E	N
15	0s	SI	47	2F	/	79	4F	O
16	10	DLE	48	30	0	80	50	P
17	11	DC1	49	31	1	81	51	Q
18	12	DC2	50	32	2	82	52	R
19	13	DC3	51	33	3	83	53	S
20	14	DC4	52	34	4	84	54	T
21	15	NAK	53	35	5	85	55	U
22	16	SYN	54	36	6	86	56	V
23	17	ETB	55	37	7	87	57	W
24	18	CAN	56	38	8	88	58	X
25	19	EM	57	39	9	89	59	Y
26	1A	SUB	58	3A	:	90	5A	Z
27	1B	ESC	59	3B	;	91	5B	[
28	1C	FS	60	3C	<	92	5C	\
29	1D	GS	61	3D	=	93	5D	]
30	1E	RS	62	3E	>	94	5E	^
31	1F	US	63	3F	?	95	5F	

MULTIDOS SYSTEM DISKETTES

MULTIDOS requires specific DOS overlay files (/DOL) present on a MULTIDOS disk, mounted in logical drive 0, for proper operation.

A MULTIDOS minimum system disk contains the following six files:

SYSRES/SYS - This is the executive program loaded during initialization. SYSRES/SYS contains the necessary routines to interface with the hardware being used. SYSRES/SYS also contains system scratchpad, vectors to internal routines, drive control table (DCT), floppy disk I/O routines, program loader, and file positioning.

DIR/SYS - This file contains all of the data on every file on the disk.

Command/DOL - The command interpreter. Command/DOL also contains the code for the library commands BLINK, BOOT, CLEAR, CLOCK, CLS, DEAD (MODEL I/III), LIB, LOAD, SCREEN, and VERIFY. The MAX-80/Model 4 version also contain the code for V64 and V80.

Open/DOL - This file contains the code to OPEN and INITialize a file.

Allocate/DOL - This file contains the code to allocate granules and file positioning to a record in the fifth or greater segment.

Close/DOL - This file contains the code to CLOSE and KILL files.

Other MULTIDOS files are required to execute certain commands and perform certain functions. The other MULTIDOS system files are:

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. This system command is not the same as the utility CAT/CMD.

Library1/EXT - Executor for the library commands APPEND, AUTO, BUILD, DATE, DEVICE, DIR, DO, DUMP, FREE, KILL, LINK, LIST, PRINT, ROUTE, TIME, and TOPMEM.

Library2/EXT - Executor for the library commands: ATTRIB, CLRDSK, CONFIG, DDAM (MODEL I), FORMS, KEYBRD, PATCH, PROT, RENAME, RESTOR, SETCOM (MODEL III), and SKIP.

Help/EXT - Executor of the library command HELP. (MAX-80/MODEL 4)

MULTIDOS requires a MULTIDOS system disk to be in logical drive 0 to load and execute a /CMD file from a MULTIDOS disk, a data disk, or an "alien" system diskette. MULTIDOS will not execute programs specifically written for an "alien" DOS. i.e. PROFILE III+.

Proper operation of certain MULTIDOS utilities do not require a MULTIDOS system disk in logical drive 0. Specific system restraints are covered in the description for the particular utility in the SYSTEM UTILITIES section of the MULTIDOS manual. MULTIDOS requires that a system disk be in logical drive 0 When any MULTIDOS utility is loaded and executed.

To remove system files, use the utility VFU/CMD with the "%" parameter.

EXAMPLE:

VFU %<ENTER>

The "%" permits VFU/CMD to access the "/DOL" and "/SYS" files.

DO NOT USE VFU/CMD TO COPY A DIR/SYS FROM ONE DISK TO ANOTHER!!

The removable system files are:

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. This system command is not the same as the utility CAT/CMD.

Library1/EXT - Executor for the library commands APPEND, AUTO, BUILD, DATE, DEVICE, DIR, DO, DUMP, FREE, KILL, LINK, LIST, PRINT, ROUTE, TIME, and TOPMEM.

Library2/EXT - Executor for the library commands: ATTRIB, CLRDSK, CONFIG, DDAM (MODEL I), FORMS, KEYBRD, PATCH, PROT, RENAME, RESTOR, SETCOM (MODEL III), and SKIP.

Help/EXT - Executor of the library command HELP. (MAX-80/MODEL 4)

LIBRARY COMMANDS

MULTIDOS library commands are usually entered after the MULTIDOS prompt has been received. However, the library commands can be executed from SUPERBASIC, or an assembly language routine.

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

EXAMPLE:

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

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

DATE 11/08/84<ENTER> {At least one space after DATE}

This establishes November 8, 1984 as the current RAM date.

Multiple library commands will be 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 will display 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 will rename "NEW/CMD" on the disk in logical drive 2 to "OLD/CMD", then display the directory of the disk in logical drive 2.

Whenever the term "switch" is part of a MULTIDOS library command line, "switch" represents an ON/YES or OFF/NO condition. If the "switch" is omitted, then ON/YES will be used.

MULTIDOS accepts 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 "(" and ")" surrounding the switch parameter.

EXAMPLE:

(1) CLOCK (N)<ENTER>
(2) CLOCK (Y)<ENTER>
(3) CLOCK<ENTER>

Example (1) will suppress the clock display, and examples (2) and (3) will display the clock.

APPEND Append a file to the end of another file.

```
APPEND filespec1 [to] filespec2<ENTER>
```

APPEND will add 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. APPEND requires both filespecs to have the same logical record length.

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.

EXAMPLE:

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


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


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

The file CLASS1/TXT will have the contents of file CLASS2/TXT added to the end of the original contents of CLASS1/TXT, and the file CLASS2/TXT will be unchanged. Here is the results of the APPEND.

CLASS1/TXT after CLASS2/TXT is appended:

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

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

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

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

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

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

ATTRIB Assign a filespec's attributes.

ATTRIB filespec (param[,param...])<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.
U=pw	pw = The new UPDATE password.
P=level	level = The protection level. (Detailed below)

The protection level is entered as words. I.E. READ, EXEC, etc.

Level	ACCESS password privilege
KILL	TOTAL ACCESS (lowest level)
RENAME	RENAME, WRITE, READ, EXECUTE
WRITE	WRITE, READ, EXECUTE
READ	READ, EXECUTE
EXEC	EXECUTE ONLY
NONE	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 higher than WRITE, I.E. READ, EXECUTE, with the use of the access password. The file cannot be RENAMEd or KILLed without the update password.

The access attribution does not require an update or protection level attribute. Files with access attribution require the ACCESS password for any access to the file.

All files have an ACCESS and UPDATE password. The default password is 8 spaces. With a password set to 8 spaces (blank), a password is not required to access the file. The library command DIR will display a "P" after the filename, if a protection level is set and the file has a non-blank UPDATE password.

EXAMPLE:

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

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

ATTRIB FILEB/BAS (V)<ENTER>

FILEB/BAS is visible and will be displayed when the library command DIR is executed.

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

AUTO[[!][#]DOSCMD][<LF>DOSCMD]<ENTER>

DOSCMD is any valid MULTIDOS library command or a filespec for an executable command file. The AUTO command will store up to 31 characters, after the mandatory space following "AUTO", on the GAT sector of the directory. If more than 31 characters are entered, no error message will be given; however, only the first 31 characters are stored. Since the "AUTO" command writes to the directory, the diskette must not be write protected when the "AUTO" command is set up or removed.

EXAMPLE:

AUTO DIR<ENTER>

On future power-ups the library command DIR will be executed after 'DIR' appears on the display.

AUTO provides for automatic operation of one or more MULTIDOS commands or executable command files upon power-up/re-boot. Multiple AUTO commands require a <LF> between each command 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 will display 'DIR', execute the library command DIR, display 'FREE', execute the library command FREE, display 'BASIC', then load and execute the file BASIC/CMD.

MULTIDOS will accept a multiple command for an AUTO command. The multiple command will be displayed once.

EXAMPLE:

AUTO DIR,FREE,BASIC<ENTER>

On subsequent power-up/re-boots, MULTIDOS will display 'DIR,FREE,BASIC', execute the library command DIR, execute the library command FREE, then load and execute the file BASIC/CMD.

An AUTO command will be removed if only <ENTER> follows AUTO.

EXAMPLE:

AUTO<ENTER>

This will remove 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, the programmer may want an AUTO command to execute without the ability of the user to suppress the AUTO command. To inhibit the suppression of an AUTO command, key in an "!" (exclamation mark) as the first character after the mandatory space following "AUTO".

EXAMPLE:

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

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

The programmer can make an AUTO command invisible if a "#" (pound sign) is placed in front of the first command.

EXAMPLE:

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

On subsequent power-ups, the file BASIC/CMD will load and execute, but "BASIC" will not displayed on the screen.

When setting up AUTO commands, the programmer 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 in:

```
AUTO %<ENTER>
```

A multiple AUTO command will be converted to a multiple command AUTO.

To query an AUTO command without re-booting the computer, key in:

```
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" will not execute. The programmer should include "DIR 1" in the "DO" file STARTUP, or rearrange the AUTO command to:

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

BLINK Disable or enable the blinking cursor.

BLINK[switch]<ENTER>

This command will enable the blinking cursor when "switch" is ON/YES, or disable the blinking cursor when "switch" is OFF/NO.

EXAMPLE:

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

BOOT Reset the computer.

BOOT<ENTER>

This command will cause a software reset. BOOT requires no arguments or parameters.

BUILD Create a "DO" file.

BUILD filespec [(A)]<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 will overwrite the filespec unless the "A" option is specified. If the "A" option is specified, then BUILD will add the input to the end of filespec.

EXAMPLE:

```
BUILD STARTUP<ENTER>        {Build input will overwrite STARTUP.}
BUILD STARTUP (A)<ENTER>    {Build input will ADD to STARTUP.}
```

To BUILD a "DO" file, follow these steps:

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

Although BUILD will accept 255 characters per build line, the command buffer can only hold 63 characters (MODEL I/MODEL III), or 79 characters (MAX-80/MODEL 4). The 255 byte line was incorporated into BUILD to accept long commands (see the "%" explanation on the next page.)

BUILD has three special characters "#", "$", and "%", to pause a "DO" file or suppress video output. The three special "BUILD" characters are only recognized if they are the first character in a "BUILD" line.

The "#" character is used to pause execution of a "DO" file. The "#" character can be the only input for a build line or the "#" character can precede a message to display before the pause occurs.

EXAMPLE:

```
Build input
#<ENTER>
```

Pauses "DO" and waits until a "SHIFT" key is pressed.

```
#INSERT DISK NUMBER 77<ENTER>
```

Pauses "DO", after displaying the message 'INSERT DISK NUMBER 77', and waits until a "SHIFT" key is pressed.

The "$" character will suppress video output and continue "DO" execution if the "$" character is the only input for a build line. If text follows the "$" character, then the text is printed, the video is suppressed, and the "DO" file pauses.

EXAMPLE:

```
Build input
$<ENTER>
```

Suppresses video output and continues "DO" file.

```
$This is the beginning of darkness!<ENTER>
```

Will display the message 'This is the beginning of darkness!', Suppress video output, then wait for a <SHIFT> to continue.

The video display is turned on whenever a "DO" file is completed. Since MULTIDOS can nest "DO" files, video output is enabled when ANY "DO" file completes.

The "%" character is BUILD's comment character. The "%" character must have additional text following or an error will occur during "DO" execution. The "%" character is required for each comment line terminated with <ENTER>.

EXAMPLE:

```
%We the people of the United States, in Order to form a more<LF>
Perfect Union, establish Justice, insure domestic Tranquility,<LF>
Provide for the common defence, 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>
```

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 will clear. [CLS]
The message 'LOADING BASIC' will appear. [%LOADING BASIC]
BASIC will load with memory set to 61440. [BASIC 61440]
The screen will clear. [CLS]
The message 'GET READY FOR A BLAST !' Will appear. [%GET READY FOR A BLAST]
The program "BLAST/BAS" will load and execute. [RUN"BLAST/BAS"]

Special BUILD characters are summarized in the following table:

BUILD Character	Action description.
#	pause and waits for <SHIFT> press prior to continuing.
#zzzzz	pause after displaying zzzzz and wait for <SHIFT> press prior to continuing.
$	suppress video output.
$zzzzz	suppress video output after displaying message zzzzz and wait for a <SHIFT> press to continue.
%zzzzz	display zzzzz and continue.

To display a message and continue:
%Message<ENTER>
To display a message and pause:
#Message<ENTER>
To display a message, suppress video output, and continue:
%Message<ENTER>
$<ENTER>
To display a message, suppress video output and pause:
$Message<ENTER>
To pause:
#<ENTER>
To suppress video output and pause:
$<ENTER>
#<ENTER>

CLEAR Zero Random Access Memory (RAM).

CLEAR<ENTER>

CLEAR zeroes RAM from 5200H to TOPMEM. If the contents of TOPMEM is FDFFH, CLEAR will set all bytes from 5200H through FDFFH to 00.

CLOCK Enable or disable the real time clock display.

CLOCK[switch]<ENTER>

The ON switch will cause the real time clock to be displayed at the top right of the video display. The OFF switch will suppress the display of the real time clock. To set the clock, use the library command TIME.

EXAMPLE:

```
CLOCK (ON)<ENTER      {The real-time clock is displayed}
CLOCK (OFF)<ENTER>    {The real-time clock is not displayed}
```

CLRDSK Zero a disk's unused granules.

CLRDSK[[:]d]<ENTER>

CLRDSK will erase unassigned granules on the disk in the specified logical drive by placing a continuous pattern of E5E5 bytes on each unassigned granule for single density media, and a 6DB6 byte pattern on each unassigned granule for double density media. This is the same pattern placed on the media by FORMAT/CMD, and the format function of BACKUP/CMD.

EXAMPLE:

CLRDSK :3<ENTER> {erase all unassigned granules on logical drive 3}

CLS Clear the screen.

CLS<ENTER>

CLS clears the screen and sets the cursor to the upper left hand corner. CLS can be used in conjunction with other commands or programs to erase the previous screen data, and present only the new data to the user.

EXAMPLE:

CLS, FREE<ENTER>

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

CONFIG[[[:]d] (P=p[,w][,N][,SI=k][,V=v][,ST=s][,m][,X])]<ENTER>

d = logical drive number, 0 to 7.
p = physical drive number. (Floppies can only 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.
v = volume, 1 or 2.
s = stepping rate, 6, 10, 12, or 30.
m = media type, 5 = 5 1/4" floppy, 8 = 3" floppy, or H = hard disk.
X = write current RAM configuration to logical drive zero.

CONFIG<ENTER> displays current RAM configuration, including the density of the media, the status of SKIP, and MEMDISK assignment.

CONFIG establishes, for the system, the hardware characteristics 1) size/type of media - 5 1/4" floppy, 8" floppy, or hard disk; 2) number of sides 1 or 2; 3) volume 1 or 2; and 4) track to track access speed.

EXAMPLE:

```
CONFIG 1(ST=6)<ENTER> {set logical drive 1's stepping rate to 6ms}
```

The CONFIG information is stored in the Drive Control Table (DCT) located in RAM from 4500H to 457FH. When a CONFIG parameter is changed, only the DCT is RAM is modified. If the system is re-booted, the power-up configuration will be loaded into RAM. The X parameter is required to write out the DCT to logical drive 0. If a logical drive has been assigned to MEMDISK, then CONFIG (X) is ignored.

CONFIG 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:

```
PROG:3           {logical drive 3}
VALUE/BAS:1      {logical drive 1}
DIR :2           {logical drive 2}
BACKUP :1 to :2  {logical drive 1 to logical drive 2}
```

If access is made to a file without a drivespec, MULTIDOS will search for the file in ascending LOGICAL drive order.

EXAMPLE:

```
ZAP<ENTER>
```

The file ZAP/CMD is searched from logical drive 0 to logical drive 7. If the file exists on logical drive 2, and logical drive 4, the version on logical drive 2 is loaded and executed.

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

EXAMPLE:

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

However, the user 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 logical 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, Physical = 1, 5" Floppy, double density,
    one sided, one volume, step rate = 06 mS.
:1, Physical = 0, 5" Floppy, double density,
    one sided, one volume, step rate = 06 mS.
:2, Physical = 2, 5" Floppy, double density,
    one sided, one volume, step rate = 06 mS.
:3, Physical = 3, 5" 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 wanted to copy information between several 40 track diskettes. Sometimes the data is going in one direction, and sometimes it is going in the other direction. 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 continuously 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. Therefore, our solution would be (1) assign logical drive 2 to physical drive 0, (2) assign logical drive 0 to physical drive 2, then (3) SKIP logical drive 0.

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

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

(3) SKIP 0<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. The user can change the write protect condition of a diskette without removing the diskette and applying/removing the write protect tab (write enable tab for 8" 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 will remove a logical drive from the system. It is useful for a setup with only two drives, and the user doesn't want MULTIDOS to contiuously 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, Physical = 0, 5" Floppy, double density,
    one sided, one volume, step rate = 06 mS.
:1, Physical = 1, 5" 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, Physical = 0, 5" Floppy, double density,
    one sided, one volume, step rate = 06 mS.
:1, Physical = 1, 5" Floppy, double density,
    one sided, one volume, step rate = 06 mS.
:2, Physical = 2, 5" 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, Physical = 0, 5" Floppy, double density,
    one sided, one volume, step rate = 06 mS.
:1, Physical = 1, 5" 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 will be 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, Physical = 0, 5" Floppy, double density,
    one sided, one volume, step rate = 06 mS.
:1, Physical = 1, 5" 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, should not be set to a faster rate than the manufacturer's specification. The values accepted for floppies are 6, 12, 20, and 30. The actual stepping rates will be set as indicated by the table below:

for 5 1/4" floppies

	MODEL I single density 1771 FDC	MODEL III/4 MAX-80 MODEL I double density 179X FDC
CONFIG (ST=6)<ENTER>	12 ms	6 ms
CONFIG (ST=12)<ENTER>	12 ms	12 ms
CONFIG (ST=20)<ENTER>	20 ms	20 ms
CONFIG (ST=30)<ENTER>	40 ms	30 ms

For 8" floppies, the indicated values and actual stepping rate will be half the 5 1/4" values, although the only CONFIG input values accepted are 6, 12, 20, and 30.

m = media type. MULTIDOS can have the configuration set for 5 1/4" floppies (5) or 8" floppies (8). In order to use the hard disk configuration (H), a hard disk driver will be needed. MULTIDOS does not have the needed driver, but the technical section should provide the necessary information to interface a hard disk driver with the DCT.

X = write the current RAM configuration to logical drive 0. The "X" parameter is used to save the current RAM configuration for future power-up/reboots. Changing a CONFIG parameter value does not record the change to the diskette in logical drive 0 unless the "X" parameter is also specified.

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

This sets up logical drive 1 for two sided operation, and saves this configuration to logical drive 0 for future power-up/re-boots.

DATE Set or display the date stored in RAM.

DATE[mm/dd/yy]<ENTER>

DATE will set 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 12/25/84   {Set RAM date to December 25, 1984}
DATE 11/18/83   {Set RAM date to November 18, 1983}
DATE<ENTER>     {Display current date stored in RAM}
```

DDAM Modify single density directory Data Address Marks. (Model I only)

DDAM[[:]d] (U or D)<ENTER>

d = logical drive number
U = USER DEFINED (used by TRSDOS, NEWDOS/21, etc)
D = DELETED (created/used by DOUBLE density controllers)

DDAM alters the Data Address Marks on a single density diskette's directory sectors. The address marks placed on a single density diskette by MULTIDOS cannot be easily read by TRSDOS, NEWDOS/21, and ULTRADOS. If a write is necessary to a single density diskette, use DDAM to convert the address marks after writing, if this diskette is to be read by TRSDOS, NEWDOS/21, or ULTRADOS, etc.

The DELETED DATA MARK is used for MODEL I to/from MODEL III, MAX-80, and MODEL 4 compatibility. If MODEL III, MAX-80, or MODEL 4 compatibility is not necessary, and "alien" system compatible address marks are desired; then patch a backup system diskette (MODEL I ONLY!!!) as follows:

PATCH SYSRES/SYS (REC=8) T=230.2>62.2<ENTER>

Once this patch is applied, the DDAM (D) function is defeated.

To reverse this patch:

PATCH SYSRES/SYS (REC=8) T=62.2>230.2<ENTER>

DEAD Software power-up. (MODEL I/MODEL III only)

DEAD<ENTER>

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

DEBUG Real time debugger.

DEBUG[switch]<ENTER>

DEBUG is a real time debugging package to use with machine language programs. DEBUG will examine and alter the contents of RAM or the Z-80 registers, jump to a specified address and begin execution with optional breakpoints, and single step Z-80 instructions in RAM or execute Z-80 CALLs from RAM. Since DEBUG loads into the overlay are of RAM (4D00H to 51FFH), DEBUG cannot be use with MULTIDOS overlays or utilities which load below 5200H. (BACKUP/CMD, CAT/CMD, FORMAT/CMD, RS/CMD, VFU/CMD, and ZAP/CMD)

Once the debugging facility is enabled - via DEBUG (ON) - it does not load and execute until an executable program, with protection level of less than EXEC, is loaded and before the first instruction is executed, or <SHIFT><BREAK>, <LEFT><SHIFT><BREAK> MODEL III/MODEL 4, is pressed. When DEBUG is initially loaded, if the display is blank, press <0> to enable one of three display formats:

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

```
{Debug pokes VIDEO refresh RAM and does use the}        AF 019B
{display routine to place data into the VIDEO. }        S  H  NC
{Therefore, while debugging a program with out-}        BC 0020
{put directed to the display, the user can use }        DE 0180
{format [1] to keep abreast of display output  }        HL 4009
{while debugging a program.                    }        AF'0044
{                                                        Z  P
{The <CLEAR> key is routed through the display }        BC'0160
{routine to reset the VIDEO from a WIDE display}        DE'0001
{or erase any extraneous data present on the   }        HL'7000
{display when Debug is entered with the display}        IX 4015
{set to format [1].                            }        IY 5180
{                                              }        SP 41EA
{Debug will function with the display disabled.}        PC 0083
```

[2] Register display with indirect RAM plus any 64-byte "page" of RAM.

```
AF 019B S  H  NC
BC 0020 C300 00C5 0604 1842 C368 4311 1540 18E3 .......B.hC..@..
DE 0180 0EFF 0C0F 30FC 7D07 0707 A957 DD4E 0528 ....0.}....W.N.(
HL 4009 0C00 1040 2000 0000 0080 0000 010E 4B00 ...@ .........K.
AF'0044  Z  P
BC'0160 00E6 81EE 8120 08DD 7E05 EE10 DD77 05AF ..... ..~....w..
DE'0001 2E06 1740 C32D 407E E3BE 23E3 C200 00C3 ...@.-@~..#.....
HL'7000 C4C5 C2D5 C720 2020 5265 616C 2074 696D .....   Real tim
IX 4015 010E 4B00 0020 8F00 07E5 0200 3C00 0000 ..K.. ......<...
IY 5180 43FF C343 C7C4 63FF CD63 C706 02F7 D302 C..C..c..c......
SP 41EA 0B07 8051 1843 1540 4F4F 4C00 9105 1843 ...Q.C.@OOL....C
PC 0083 DDE1 FDE1 E1D1 C1C9 F1D5 E5FD E5DD E5D5 ................
   0083 DDE1 FDE1 E1D1 C1C9 F1D5 E5FD E5DD E5D5 ................
   0093 DD21 8300 DDE3 F5AF C93A E837 E6F0 EE30 .!.......:.7...0
   00A3 C90F 380F 0F38 090F 3803 C3E0 00C3 E000 ..8..8..8.......
   00B3 C3E0 00C3 E000 0F30 E80F 380B 0F38 0E0F .......0..8..8..
```

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

```
4400 3E10 EF00 003E 40EF 56F5 3619 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 formats ([1] or [2]), DEBUG displays all the Z-80 registers, organized for interpretation either as two 8-bit registers or as 16-bit register pairs. Since most programs use several sets of register pairs as indirect pointers or indexing registers, 16 bytes of indirect data are presented with each register pair in format [2]. Each of the flag registers is shown with an ASCII representation of its flag bits. For the flag registers, the hex contents is displayed, along with a bit-by-bit alphabetic code which makes it easier to interpret the flag status. For example, bit 0 (rightmost bit) is the carry flag, the alphabetic code shows an C in that position whenever the carry flag is "set".

Table of codes for all the flag bits:

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

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 must be hexadecimal without the "H" suffix.

Command	Terminator Required	Operation Performed
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 display formats. (Current screen contents is unaffected)
G[jjjj[,kkkk]]	<ENTER>	Place jjjj in PC register and executes with optional breakpoint at kkkk.
I	none	Single-steps next RAM instruction.
M[cccc]	<SPACE>	Sets the current modification address to cccc. Modification information will be displayed in the lower left of the screen. If cccc is currently in the display, it will be overlayed by a transparent cursor. If cccc is omitted, the last modification address will be used for cccc.
N	none	Produces continuous "I" commands until <SHIFT><SPACE> is pressed.
O	none	Enables the display formats.
P	none	Set display to register only - format [1].*
Rrp aaaa	<SPACE>	Loads register pair rp with the value aaaa.
Q	none	Sets display to previous format [2] or [3].*
S	none	Sets display to full screen - format [3].*
U	none	Dynamic display update until <SHIFT><SPACE> is pressed.
X	none	Sets display to register format [2].*

* DISPLAY FORMAT IF DIPLAY IS ENABLED VIA COMMAND "O". IF COMMAND "F" WAS ISSUED, DISPLAY FORMATS MAY BE CHANGED PRIOR TO A COMMAND "O".

up-arrow	none	Decrements memory display by 16d, 10H bytes.
<SH>up-arrow	none	Decrements memory display by 4096d, 1000H bytes.
down-arrow	none	Increments memory display by 16d, 10H bytes.
<SH>down-arrow	none	Increments memory display by 4096d, 1000H bytes.
right-arrow	none	Increments memory display by 1 byte.
<SH>right-arrow	none	Increments memory display by 256d, 100H bytes.
left-arrow	none	Decrements memory display by 1 byte.
<SH>left-arrow	none	Decrements memory display by 256d, 100H bytes.

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

EXAMPLE:

DFAE944<SPACE>

Will display the page of memory starting with address E944H.

The "M" command detail....

Any time you wish to alter the contents of a memory location, key Mcccc then <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 will be updated, then DEBUG will increment the modification address by one.

To increment the modification address and leave the current address unchanged, key <SPACE>, <COMMA> or right-arrow. To decrement the modification address and leave the current address unchanged, press the left-arrow. Similarly the up-arrow will decrement the modification address by 16d, 10H, and the down-arrow will increment the modification address by 16d, 10H. Pressing any non-hex key will exit the modify memory mode.

To disable DEBUG, key in:

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

DEVICE List current I/O devices.

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 (Model III, MAX-80, and MODEL 4 only), and SO = RS 232-C output (Model III, MAX-80, and MODEL 4 only).

EXAMPLE:

```
DEVICE<ENTER>

KI = I at X'4BC4'
DO = O at X'4275'
PR = O at X'4275', routed to DO
SI = I at X'4C7C'
SO = O at X'FF00', routed to :1
```

The keyboard is an input device with the routine starting at RAM address 4BC4H. The display is an output device with the routine starting at 4275H. The printer is routed to the display. The RS 232-C input routine starts at 4C7CH, and the RS 232-C output is routed to a file on logical drive 1.

DIR Display a specified disk's directory.

```
DIR[mask][ [[:]d]([A][,I][,P][,R][,S])]<ENTER>
```

The DIR command displays the file directory for a single drive, including the drive number, disk title, disk date, the number of logical tracks, the number of physical tracks, the free space in granules, the free space in kilobytes (1024d), and names of all visible and non-system files on the disk in ascending alphabetical order.

The "mask" is used to limit the display to selected files which match the "mask". The "mask" is optional, and has two wildcards. The "?" character is used as a substitute for any character, and the "*" character is used as a filler substitute for any character. The "mask" option does not override the "I" or "S" parameters discussed on the next page.

EXAMPLE:

USER INPUT	FILES DISPLAYED
DIR */CMD<ENTER>	with an extension of "/CMD".
DIR */ASM<ENTER>	with an extension of "/ASM".
DIR B*/*<ENTER>	with a "B" in the first position.
DIR ??A*/*<ENTER>	with an "A" in the third position.
DIR */<ENTER>	without an extension.
DIR ???/*<ENTER>	with 1, 2, or 3 characters in the filename.

At the MULTIDOS command level, DIR can be invoked with one keystroke. The key pressed represents the logical drive desired. The keystrokes are:

```
        "0" = DIR :0
        "1" = DIR :1
        "2" = DIR :2
        "3" = DIR :3
  <SHIFT>0 = DIR :0'        {MAX-80 and MODEL 4 only}
  <SHIFT>1 = DIR :1'
  <SHIFT>2 = DIR :2'
  <SHIFT>3 = DIR :3'
```

The key must be the first keystroke after <ENTER> or <BREAK> for the MODEL I/III or in the first position for the MAX-80 and MODEL 4.

The "A" parameter will display, for each file, the date of the last update (Date), the physical allocation, including the end of file sector and the last byte within the end of file sector (Eof), the logical record length (Lrl), the number of logical records (Lrec), the number of segments - contiguous granules (Seg), and the file size in granules (Grans). A maximum of 11 files (64X16 screen) or 19 files (80X24 screen) will be displayed at one time. To display the next file, press <SPACE>; or to display another "page" of files, press <ENTER>.

EXAMPLE:

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

0  MULTIDOS 12/01/84  35 log  35 phy cyls   32 grans    48.00 K
Filename          Date        Eof      Lrl   Lrec    Seg  Grans

ZAP/CMD P       11/26/84     31/117    256     32      1      6
```

The file ZAP/CMD was last updated on 11/26/84, uses 31 full sectors and 117 bytes in the 32nd sector, has a logical record length of 256, consumes 32 sectors in one segment consisting of 6 contiguous granules.

The "I" parameter will display the files with the "I" attribute as well as visible files. The files with an "I" attribute are indicated with an "I" after the filename.

EXAMPLE:

```
DIR :2(I)<ENTER>

2  MULTIDOS 12/01/84  35 log  35 phy cyls   32 grans    48.00 K

BACKUP/CMD I    BASIC/CMD I     BBASIC/CMD I    CAT/CMD P
COPY/CMD I      DBLFIX/CMD      DDT/CMD         FORMAT/CMD I
GR/CMD          MEM/CMD P       MEMDISK/CMD     PRT/CMD
RS/CMD P        SPOOL/CMD       TAPE/CMD        VFU/CMD P
ZAP/CMD P
```

The "P" parameter will direct the output to the printer as well as the display, and the directory listing will scroll until the last file is displayed.

EXAMPLE:

```
DIR 3(A,P)<ENTER>
```

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

```
3 MULTIDOS 12/01/84  35 log  35 phy cyls   32 grans     48.00 K
Filename          Date        Eof      Lrl   Lrec    Seg  Grans

CAT/CMD P       07/31/84      9/137    256     10      1      2
CDIR/CMD        10/20/84      2/117    256      3      1      1
DBLFIX/CMD      06/30/84      2/64     256      3      1      1
DDT/CMD         07/13/84      4/32     256      5      1      1
FMAP/CMD        10/14/84      4/176    256      5      1      1
GR/CMD          07/25/84      1/3      256      2      1      1
HELP/CMD        09/24/84     23/95     256     24      1      4
LO/CMD          11/17/84      8/209    256      9      1      2
MEM/CMD P       07/13/84      1/195    256      2      1      1
MEMDISK/CMD     10/29/84      3/176    256      4      1      1
PRT/CMD         08/05/84      3/231    256      4      1      1
RS/CMD P        06/30/84      4/227    256      5      1      1
SPOOL/CMD       08/04/84      4/172    256      5      1      1
TAPE/CMD        07/13/84      7/68     256      8      1      2
VFU/CMD P       09/02/84     17/102    256     18      1      3
ZAP/CMD P       11/26/84     31/117    256     32      1      6
```

The "R" parameter will display removed files, provided the directory entry location was not overwritten. (CDIR/CMD clears out a removed file's directory entry). Removed files are indicated by having a "R" after the filename

The "S" parameter will display the system files as well as visible files. System files are indicated with a "S" after the filename.

EXAMPLE:

```
DIR */?OL 1(S)

1 MULTIDOS 12/01/84  35 log  35 phy cyls   32 grans     48.00 K

Allocate/DOL S  CREF/BOL S      Close/DOL S     Command/DOL S
Debug/DOL S     EDIT/BOL S      ERROR/BOL S     Error/DOL S
Minidos/DOL S   Open/DOL S      PACK/BOL S      RENUM/BOL S
UNPACK/BOL S    UTIL/BOL S
```

DO Substitute disk file for keyboard input.

DO filespec<ENTER>

DO will execute the filespec entered. Usually the filespec has been previously created with the BUILD command. The DO command will add 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, then resets TOPMEM after the "DO" file is complete. 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 the multiple command AUTO, 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>

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

DUMP will transfer the contents of memory starting at "ssss" and ending at "eeee" to the filespec, and will add the extension "/CMD" to the filespec if none was entered and the "CIM" parameter is not specified. The transfer address, "tttt", will default to 402DH. The transfer address 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 will be transferred to a disk file named BUG/CMD. When the command BUG<ENTER> is entered, MULTIDOS will load memory from FD00H to FFFFH and begin execution at FE00H.

The "TITLE" parameter will write 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 will be transfered to a disk file named DCT/CIM.

FORMS Set or display printout format parameter values.

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

I = intialize 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 (value=0) sent after each line feed (decimal 10)
L = issue a line feed (decimal 10) after each carriage return (decimal 13)
C = route formatted printer output to RS-232-C (not MODEL I)
OFF = sets m to 0, w to 255, t to 60, p to 60, n to 0, L off, C off, and the line and character counters to 0. (OFF,X removes PRT "clone")
X = save current forms setting to the system diskette for default parameter values on subsequent power-up/re-boots. The diskette in drive zero must not be write protected. (Saves a "clone" of PRT/CMD - MODEL I/III)

The maximum line width is determined by the "m" and "w" values. When a line exceeds the sum of these two values, FORMS will automatically insert a carriage return (decimal 13) to start a new line. If the "w" parameter is set to 255, then FORMS will not insert carriage returns. Most printers have a paper width of 80 characters. If you want to punch holes on the left side and not punch into your printed text, set the "m" parameter to 5, and the "w" parameter to 75 (80 - 5 = 75).

EXAMPLE:

FORMS (M=5,W=75)<ENTER>

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.

FORMS will interpret a decimal 11 as a line feed without a carriage return (vertical tab one line). Decimal 11 can be used to send line feeds to certain printers which do not respond to multiple line feed characters. When ESC, decimal 27, is encountered, FORMS will not count this or the next character in the page width. i.e. LPRINT CHR$(27);CHR$(12) will send both codes directly to the printer without generating a form feed, and leave the character counter where it was before this ESC pair was sent to the printer.

The RAM address for the FORMS parameter values are:

Text length.	(T)	4028H
Line counter.		4029H
Width of text.	(W)	402AH
Character counter.		402BH
Print lines per page.	(P)	402CH

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

To list the current parameter values, key:

FORMS<ENTER>

NOTE: MODEL I/III MULTIDOS requires PRT/CMD executed before FORMS can be used. However, if a FORMS (.....,X) is performed, then FORMS will be available upon power-up/re-boot.

FREE Display the free space on all mounted disks.

FREE<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 will total all the free file spaces and available disk space in kilobytes.
****** For you HARD disk users, FREE will total up to 999 Meg!! ******

HELP Provide a brief explanation of a MULTIDOS LIBRARY command.

HELP[command]<ENTER>

HELP will assist you in using the MULTIDOS LIBRARY commands along with the format required to execute each command. If the command is omitted or incorrect, HELP will list all of the available commands in the HELP file. This is a library command for the MAX-80 and MODEL 4 only. The MODEL I and MODEL III version has a command file HELP/CMD which performs the same function as the MAX-80 and MODEL 4 library command HELP.

KEYBRD Set or display keyboard attributes.

```
(1) KEYBRD[ ([C=cc][,L=sw])]<ENTER>
(2) KEYBRD[ ([C=cc][,L=sw][,W=sw][,E=sw])]<ENTER>

cc = cursor character value (0 to 255 or X'00' to X'FF')
L  = UPPER/lower case at power-up/re-boot
W  = 80 column display at power-up/re-boot
E  = graphic character conversion for use with certain printers
sw = Y, YES, or ON if the function is desired
sw = N, NO, or OFF if the function is not desired
```

(1) KEYBRD will establish the cursor character, and the character case on power-up/re-boot for the MODEL I and MODEL III. In addition, (2) KEYBRD will establish the video width, or graphic conversion [for EPSON (tm) printers capable of reproducing the TRS-80 (tm) video graphic characters] on power-up/re-boot for the MAX-80 and MODEL 4.

If a KEYBRD attributes is changed, it is updated on the system disk, but does not go into affect until a power-up/re-boot occurs.

EXAMPLE:

```
KEYBRD (L=N,C=176)<ENTER>
```

This command will set the case to UPPER, and change the cursor character to 176d on future power-up/re-boots by overwriting the current values on the system disk in logical drive zero.

To list the current KEYBRD attributes, key:

```
KEYBRD<ENTER>
```

KILL Remove a filespec.

```
KILL filespec<ENTER>
```

KILL will reset the directory in use bit and deallocate 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, MULTIDOS will search for the file in ascending logical drive order and remove it.

LIB Display the MULTIDOS library commands.

```
LIB<ENTER>
```

This command lists the available library command for a full system disk. The library commands BLINK, BOOT, CLEAR, CLOCK, CLS, DEAD (MODEL I/III), DEBUG, LIB, LOAD, VERIFY, V64 (MAX-80/MODEL 4), and V80 (MAX-80/MODEL 4) are contained in Command/DOL which uses RAM from 4D00H to 51FFH. The rest of the MULTIDOS library commands operate in RAM from 5200H to 68FFH.

LINK Simultaneous output of output devices.

LINK[dvl [to] dv2]<ENTER>

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

EXAMPLE:

LINK DO to PR<ENTER>

This will echo to the display anything going to the printer.

EXAMPLE:

LINK PR DO<ENTER>

This will send to the printer every byte going to the video DCB, including control characters (decimal 1 to 31). The PRT/CMD file will fi%ter out specific control characters if you do not want them to do all kinds of things to the printout such as: start underlining, compress print, expanded print, form feeds, etc. (See PRT/CMD for details.)

To un-LINK any or all LINKed devices, key:

LINK<ENTER>

If dvl is LINKed to dv2, then dv2 cannot be LINKed to dvl, dvl cannot be ROUTEd to dv2, and dv2 cannot be ROUTEd to dvl. However if dvl is LINKed to dv2, and dv2 is LINKed to dv3, and dv3 is LINKed to dvl, the system will crash.

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 will also go to the filespec. A subsequent LINK<ENTER> will stop 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 will also go to the printer. To filter out control codes going to the filespec key:

FORMS (C)<ENTER>
ROUTE SO to filespec<ENTER>
LINK PR to DO<ENTER>

The codes to filter out must be entered when PRT/CMD is executed.

If the RS-232-C is linked to the display, several of MULTIDOS utilities will be severely hampered. BACKUP/CMD, FORMAT/CMD, ZAP/CMD. Although these utilities will operate very slowly, the other functions will operate flawlessly.

LIST Display disk file.

LIST filespec[([G][,C][,D])]<ENTER>

LIST will display the filespec's contents to the display. If the file contains control codes (decimal 1 to 31), LIST will suppressed the control codes unless the "C" parameter is specified. Graphic characters are suppressed unless the "G" parameter is specified. LIST can be terminated by pressing <BREAK> or suspended by pressing <SHIFT>@. To continue from a listing pause, press any other key except <SHIFT>@.

The "D" parameter overrides the "C" and "G" 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 will transfer the filespec into RAM then return control to MULTIDOS. If the filespec loads between 4D00H and 51FFH, Command/DOL will load this area when control returns MULTIDOS. If the file loads between 5200H and 68FFH, most library commands, if executed, will load in this area.

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>

nn = physical record of filespec (NOTE: 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.

PATCH will change any disk file, regardless of the password protection level. The target or byte separator can be either a semicolon ";" or a period ".".

EXAMPLE:

PATCH CREF/BOL (REC=1) T=2;245.32>3.245;48<ENTER>

This will patch the file CREF/BOL to show the total references during a full reference listing. 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". PATCH does not recognize logical record lengths, REC=nn must be the PHYSICAL record.

PRINT Print a message or the contents of a file to the printer.

(1) PRINT"message<ENTER>
or (2) PRINT filespec<ENTER>

PRINT can print a message to the printer from the MULTIDOS command mode by placing a double quote immediately after PRINT is keyed in.

EXAMPLE:

PRINT"The directory of my special operating system.<ENTER>

Will print:
The directory of my special operating system.

EXAMPLE:

PRINT""TEST NUMBER 1"<ENTER>

Will print:
"TEST NUMBER 1"

PRINT will print the contents of the filespec to the printer. If the file contains control codes (decimal 1 to 31), PRINT will suppress the control codes unless the "C" parameter is specified. Graphic characters are suppressed unless the "G" parameter is specified.

PROT

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

LOCK = assign MASTER PASSWORD to all user files
UNLOCK = remove all passwords from user files
PW = change the MASTER PASSWORD
DATE = date all files to current RAM date used with LOCK/UNLOCK

PROT can change the disk MASTER PASSWORD, lock - add master password to, or unlock - remove all passwords form all visible and non-system files on the disk. The drivespec referenced must not be write protected.

EXAMPLE:

PROT (LOCK,DATE)<ENTER>

This will assign the MASTER PASSWORD to all user files, and redate all user files to the current RAM date. If no parameters are entered with PROT, you will be prompted to change the disk's name and the date. There is NO checking for a valid name or date; use what you wish! If you only want to change one of the name/date pair, simply <ENTER> for the one not to be changed. To prematurely exit this command, press <BREAK>.

NOTE: The master PASSWORD on your MULTIDOS disk is PASSWORD.

RENAME Change filename.

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

This command will change the name of filespec1 to filespec2. If the optional "#" is specified, then the renamed file's date will be changed to the current RAM date. Filespec2 will contain the protection level, password, and directory attributes of filespec1. RENAME will not duplicate a existing filename on the same disk, and will only change the primary HIT (Hash Index Table - see ZAP) byte. ZAP will complain if the renamed file is in more than four segments - at least two HIT entries.

RESET Reset I/O devices and restore TOPMEM.

RESET<ENTER>

RESET will un-LINK, un-ROUTE, remove all interrupt tasks, and restore TOPMEM to the power-up/re-boot value. If a ROUTE to filespec is active, RESET will close the file, then un-ROUTE the device.

RESTOR Recover a KILLed filespec.

RESTOR filespec:d<ENTER>

RESTOR will attempt to recover a removed file. If the file space on the disk has been re-assigned you will be notified of this condition and RESTOR will abort. The drive number is mandatory for RESTOR to work properly.

ROUTE Redirect one device to another.

(1) ROUTE[dv1 [to] dv2]<ENTER>
(2) ROUTE dv1 [to] filespec<ENTER>

ROUTE will change the routine entry point of dv1 to dv2's entry point. If format (2) is used, ROUTE will create a buffer in high memory and direct the output destined for dv1 to the specified filespec.

EXAMPLE:

ROUTE PR to DO<ENTER>

This will cause all bytes normally going to the printer to go to the display instead.

To un-ROUTE all devices and close the filespec for format (2), key:

ROUTE<ENTER>

SCREEN Dump the screen contents to the printer.

SCREEN<ENTER>

SCREEN will transfer the screen contents to the printer.

SETCOM Set or display RS-232-C, parameter values. (MODEL III only)

SETCOM[([BAUD=r][,sel][,WORD=w][,STOP=b][,psw][,tsw][,DTR][,RTS])]<ENTER>

r   = BAUD rate. Accepted are: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, or 19200.
sel = parity selection, ODD or EVEN
w   = Word length. Accepted are: 5, 6, 7, or 8.
bb = Stop bits. Accepted are: 1 or 2.
psw = Parity Switch. PE for parity enabled, and PD for parity disabled.
tsw = Transmit switch. TE = Transmitter enabled, TD = Transmitter disabled.
DTR if entered sets DTR high, otherwise DTR will be set low.
RTS if entered sets RTS high, otherwise RTS will be set low.

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

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

SKIP[[:]d]<ENTER>

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

SKIP will enable a 80 track drive to READ a 40 track diskette. If no drive number is specified, then all SKIP's are removed.

TIME Set or display time.

TIME[hh:mm:ss]<ENTER>

hh = hour, mm = minute, ss = second

EXAMPLE:

TIME 22:21:00<ENTER>

This will set RAM's time to 10:21 pm.

TOPMEM Set or display upper MULTIDOS system memory.

TOPMEM[xxxxx]<ENTER>

xxxxx = a decimal number from 28671 to 65535,
or a hexadecimal number from 6FFFH to FFFFH.

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

EXAMPLE:

TOPMEM BFFFH<ENTER>

This will set the upper memory usage to BFFFH.

EXAMPLE:

TOPMEM 61439<ENTER>

This sets the upper memory usage for MULTIDOS to 61439.
TOPMEM without a value will display the current TOPMEM value in both hexadecimal and decimal.

EXAMPLE:

TOPMEM<ENTER>
EFFFH, 61439

VERIFY Reread a written sector.

VERIFY[switch]<ENTER>

VERIFY will cause all disk writes to be reread for parity. All directory writes are automatically verified.

V64 Set video width to 64 characters. (MAX-80 and MODEL 4)

V64<ENTER>

V64 will set the video width to 64 characters then clear the display.

V80 Set video width to 80 characters. (MAX-80 and MODEL 4)

V80<ENTER>

V80 will set the video width to 80 characters then clear the disaply.

BACKUP/CMD Duplicate a disk.

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

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

BACKUP will duplicate all files from one disk to another. The "A" option, if specified, will cause an absolute format and does not check or warn the user if the destination disk contains data. Source and destination logical drive numbers can be specified on the BACKUP command line. The source and destination logical drive numbers can be the same. If the logical drives are the same, BACKUP will prompt the user 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 will take you through the easy procedure to duplicate a disk. After BACKUP/CMD is loaded and executes, the screen will clear and

´MULTIDOS Disk Duplicator Program - Version X.X

Q1 ´Which drive contains the source diskette? _´

will appear. BACKUP 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 will be bypassed.

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

Q2 ´Which drive for the destination diskette? _´

will appear. Again BACKUP 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 will be bypassed.

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

Q3 ´Press "ENTER" when the source disk is in drive X.´

will appear. 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 <ENTER>.

MULTIDOS will analyze the source disk for track count and density, then
´The source diskette has YY tracks, in ZZZZZZ density.

Q4 Track count for the destination diskette (18 to 96)? _´

will appear. Where "YY" is the number of tracks on the source disk and "ZZZZZZ" is the density of the source disk. The destination disk will be formatted in "ZZZZZZ" density; however, you may specify the track count for the destination disk. If <ENTER> is the response for the track count, BACKUP will use "YY" for the track count.

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

'Insufficient tracks to copy all files!'

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

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

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

```
'Disk contains data.
 The disk name is DISKNAME, dated mm/dd/yy.
Q5   Re-format this diskette (Y/N/Q)? _'
```

will appear. If you want to reformat this disk, press <ENTER> or input a <Y>. If you want to bypass the formatting, input <N> - BACKUP will skip the formatting but verify the destination disk. If you want to abort BACKUP, press <BREAK> or enter <Q>.

BACKUP does not check the destination disk 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).

If the source drive and destination drive are the same, BACKUP will prompt the user to insert the source disk and destination disk as required. The swapping will continue as necessary until all the files are copied to the destination disk.

After BACKUP has copied all files:

'Completed.

will be displayed.

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

'Insert SYSTEM <ENTER>'

will also be displayed. This prompts you to insert a MULTIDOS disk to return to the state prior to entering BACKUP.

The format pattern placed on the disks by BACKUP/CMD may be changed by zapping one to four bytes on BACKUP/CMD's first sector. Key ZAP<ENTER>, press <F>. Enter BACKUP/CMD to the Filespec prompt. When ZAP/CMD requests "Relative sector in file BACKUP/CMD (0 to 0016/010H)....", press <ENTER>. Relative sector number zero of BACKUP/CMD will be 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.

Relative sector zero of BACKUP/CMD

```
`B HEX00 057A 4241 434B 5550 2020 2031 3138 347E .zBACKUP   1184~
A    10 2A20 2020 204E 4F54 4943 4520 2020 202A *    NOTICE    *
C DR 20 2A20 2028 6329 2020 2031 3938 3320 202A * (c)  1983  *
K 1  30 2A20 436F 736D 6F70 6F6C 6974 616E 202A * Cosmopolitan *
U    40 2A20 2045 6C65 6374 726F 6E69 6373 202A *  Electronics *
P TR 50 2A20 436F 7270 6F72 6174 696F 6E2E 202A * Corporation. *
/ 03 60 2A20 2020 204E 4F54 4943 4520 2020 202A *    NOTICE    *
C 03 70 2A2A 2A2A 2A2A 2A2A 2A2A 2A2A 0102 0034 ************...4
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 5072 6573 7320 2245 4E54 4552 2220 7768 Press "ENTER" wh
   00 B0 656E 2074 6865 2003 0A43 6F6D 706C 6574 en the ..Complet
      C0 6564 2E0D 736F 7572 6365 2064 6973 6B20 ed..source disk
FILE D0 6973 2069 6E20 6472 6976 6520 302E 0364 is in drive 0..d
0000 E0 6573 7469 6E61 7469 6F6E 2064 6973 6B20 estination disk
000H F0 6973 2069 6E20 6472 6976 6520 302E 03CD is in drive 0...
```

Here are several byte patterns with the sensitivity indicated.

00 00 = least sensitive

E5 E5 = mildly sensitive (IBM single density)

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

5B 5B = intermediate (TRSDOS (tm) MODEL III)

6D B6 = most sensitive (MULTIDOS)

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.

CAT/CMD Display a directory of a TRS-80 (tm) disk.

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

A = disk map of files
I = display invisible files
M = wait for "alien" disk mount in logical drive 0
S = display system files

CAT/CMD will display the directory on practically any TRS-80 (tm) disk, including MODEL I, MODEL III, MODEL 4, and MAX-80 disks, regardless of the address marks, density, or sector/granule format. See library command DIR for the explanation of "mask".

CDIR Zero unused directory entries.

CDIR[[:]d]<ENTER>

CDIR will zero unassigned directory entries. When a file is KILLed, MULTIDOS simply resets a bit to indicate the directory entry is available. The library command RESTOR, simply sets this bit. CDIR will completely zero the entire 32 bytes of the directory entry, thus preventing any recovery of a KILLed file.

CONVERT/CMD Change address marks on single density disks. (not MODEL I)

CONVERT[[:]d]<ENTER>

CONVERT will change the USER DEFINED address marks on an single density diskette's directory sectors to DELETED address marks. CONVERT is required for the MODEL III, MODEL 4, and MAX-80 versions of MULTIDOS to read certain single density diskettes. 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 will change 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.'

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

COPY/CMD Duplicate a single file.

```
COPY [#]filespec1:d1 [to] [filespec:]d2
COPY [:]d [#]filespec1 [to] filespec2
COPY [:]d [#]filespec
COPY $ [#]filespec:d1 [to] [filespec:]d2   (d1 or d2 = 0)
COPY $[:]0 [#]filespec1 [to] filespec2
COPY $[:]0 [#]filespec
```

COPY/CMD will duplicate a file from one disk to another.

The disk containing the file will be referred to as the source disk. The disk to receive the file will be referred to as the destination disk. 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 will prompt 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 will duplicate the contents of SHIFT/TXT to MURK/ABC in logical drive number three.

Whenever logical drive zero is a drivespec, and the source OR destination disk is not a system disk, a "$" MUST precede the source filespec. A MULTIDOS system disk is a disks 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
COPY $ MANUAL/TXT:3 0
```

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 system disk into logical drive 0, then press <ENTER>.

COPY normally assigns the source filespec's date to the destination file. However, if 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 will not prompt you to mount a disk when the source and destination drivespecs are different, and neither is logical drive 0.

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

Does NOT prompt the user to mount any disks.

The colon, ":", is optional for stand alone drivespecs.

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

DBLFIX/CMD Fix boot sector on DBLDOS (tm) diskettes.

DBLFIX [:]d

DBLFIX/CMD will modify DBLDOS (tm) data and system diskettes to be read and written to by MULTIDOS. The DBLDOS (tm) will perform the same after DBLFIX/CMD has modified the DBLDOS (tm) diskette. The diskette to be modified must not be write protected.

DBLFIX/CMD requires at least two drives. The drive number specified, "d", must be other than zero.

EXAMPLE: To fix a DBLDOS (tm) diskette in drive 1.

DBLFIX :1

DDT/CMD 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.

NOTE: The MODEL I version of DDT/CMD executes an OUT 254,0, and the MODEL III version executes an OUT 95,0 to reset any high speed modification. Hardwired highspeed modifications must be reset to the normal CPU speed for DDT/CMD to indicate the correct rotation speed.

FMAP/CMD Allocation map.

FMAP[[:]d][P][D]<ENTER>

d = logical drive
P = send to printer
D = double space (used with P)

FMAP/CMD will display and/or print a map of allocated granules, directory granules, and any flawed granules.

EXAMPLE:

FMAP 1 PD<ENTER>

This will printout an allocation map of the disk in logical drive one, and double space each line. FMAP uses these symbols to indicate how a disk is allocated.

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

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
- = to signify a disk name to follow

FORMAT/CMD will prepare a disk for file storage only.

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

´MULTIDOS Formatter Program - Version X.X

Which drive contains the diskette to be formatted? _´

will be displayed. Reply with the target diskette´s logical drive number. If the drive number was specified in the FORMAT command line, then this query will be 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) ´Track count for this diskette (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".

´Which track for the DIRECTORY (1 to XX - default 17)? _´

XX = One less than the track count entered in (Q3). Press <ENTER> to use 17, or enter the desired directory track.

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

```
'Disk contains data.
 The disk name is DISKNAME, dated mm/dd/yy.
 Re-format this diskette (Y/N/Q)? _'
```

will appear. If you want to reformat this disk, press <ENTER> or input a <Y>. A <N>, <Q>, or <BREAK> response will to abort FORMAT.

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 0014/00EH)....", press <ENTER>. Relative sector number zero of FORMAT/CMD will be 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 464F 524D 4154 2020 2031 3138 347E .zFORMAT   1184~
0    10 2A20 2020 204E 4F54 4943 4520 2020 202A *    NOTICE    *
R DR 20 2A20 2028 6329 2020 2031 3938 3420 202A * (c)   1984  *
M 0  30 2A20 436F 736D 6F70 6F6C 6974 616E 202A * Cosmopolitan *
A    40 2A20 2045 6C65 6374 726F 6E69 6373 202A *  Electronics *
T TR 50 2A20 436F 7270 6F72 6174 696F 6E2E 202A * Corporation. *
/ 02 60 2A20 2020 204E 4F54 4943 4520 2020 202A *    NOTICE    *
C 02 70 2A2A 2A2A 2A2A 2A2A 2A2A 2A2A 0102 0034 ************...4
M    80 4442 4C20 4445 4E53 4954 5920 3D20 6DB6 DBL DENSITY = m.
D SE 90 534E 4720 4445 4E53 4954 5920 3D20 E5E5 SNG DENSITY = ..
   00 A0 4446 4C54 2046 4D54 2054 524B 203D 2028 DFLT FMT TRK = (
   00 B0 3E80 CD64 3918 1C3E 00CD 9644 3AEC 370F >..d9..>...D:.7.
      C0 30FA FDCB 0056 280B 3E00 CD96 443A EC37 0....V(.>...D:.7
FILE D0 0F30 FA3A 303A CD8A 3922 B136 219F 36CD .0.:0:..9".6!.6.
0000 E0 6744 1600 CD1D 3A0E 00CD 0039 C27B 36FD gD....:....9.{6.
000H F0 CB02 7E28 1432 4439 3A63 3457 C5CD 1D3A ..~(.2D9:c4W...:
```

00 00 = least sensitive

E5 E5 = mildly sensitive (IBM single density)

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

5B 5B = intermediate (TRSDOS (tm) MODEL III)

6D B6 = most sensitive (MULTIDOS)

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.

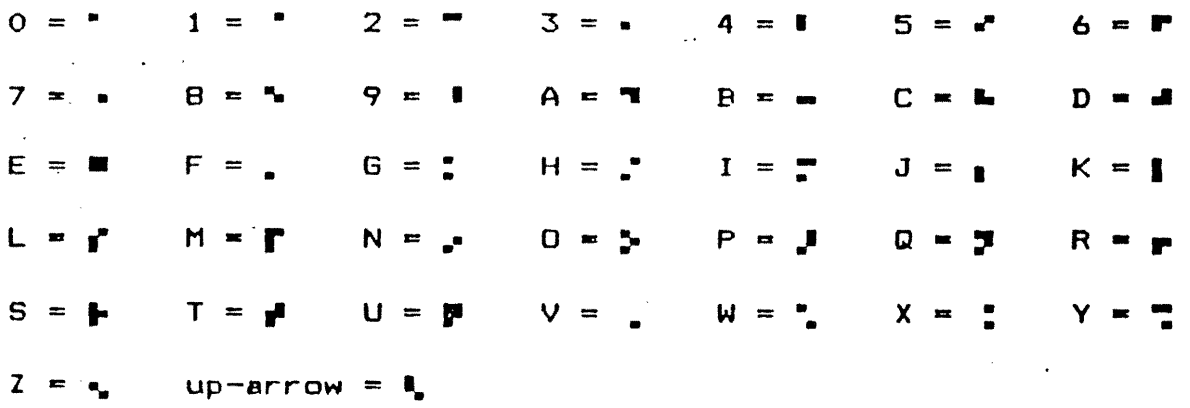
GR/CMD Configure keyboard for graphics. (MODEL I/III only)

GR<ENTER>

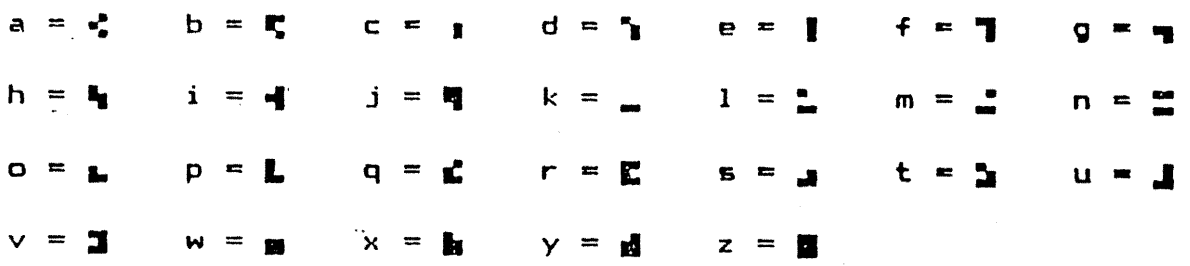
GR/CMD will modify the keyboard to produce graphic characters directly from the keyboard. GR/CMD loads into high memory and resets the TOPMEM address. To produce keyboard graphics press <SHIFT><CLEAR>. To return to normal keyboard characters press <SHIFT><CLEAR> again.

Keyboard graphic character set

(pixels lit)



lowercase (or SHIFTED without lowercase keyboard)



HELP/CMD (see HELP under LIBRARY COMMANDS)

LO/CMD Executable object file offset.

LO[filespec]<ENTER>

LO/CMD will change where a file loads or put a image of the loaded file in a filespec or place the image on disk sectors. LO/CMD does not require a "/CMD" extension. However, if you want to access a file without an extension, place a "/" after the filename. Lets do this one together.

LO LO<ENTER>

The screen clears and

```
'Load Offsetter - Version 1.0 (c) C.E.C. 1984.
 Source Filespec? LO
 Relative or fixed offset (R/F)? _'
```

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

```
'Program loads:
 5200 - 522C
 5300 - 5AFB
 Entry point = 5200
 Offset appendage (Y/N)? _'
```

is displayed. The information on the display tells us that the file LO/CMD loads from 5200H to 522CH, skips over 522DH thru 52FFH, loads 5300H to 5AFBH, and has an entry point of 5200H. A <Y> or <ENTER> response will tell LO/CMD to add an offset appendage to a file we would specify later. The offset appendage will move the file to 5200H before the file executes. Respond <ENTER> (default "Y"). If we had entered an N, we would bypass this query.

```
'Disabled or enabled interrupts (D/E)? _'
```

will be displayed. Press <ENTER> (default = first choice, "D" disabled). Then

```
'New base address in HEX? _'
```

is displayed. An <ENTER> here will default to the lowest load address displayed (5200H). Key in 7000<ENTER>. Now

```
'Destination filespec? _ '
```

is displayed. Here is where we enter the filename (auto "/CMD" appended). Instead we will start over by entering an "*" character to this query. The "*" character will start us at the "Source Filespec? _" prompt, if entered to "Offset appendage (Y/N)? _" or queries to follow. Key in LO<ENTER>. Now, lets respond F<ENTER>.

```
'Load base starting address HEX (5B00 minimum)? _'
```

is displayed. Press <ENTER> (default = 5B00H).

```
'Program loads:
 5200 - 522C
 5300 - 5AFB
 Entry point = 5200
 Load marks or direct dump (L/D)? _'
```

is displayed. Press <ENTER> (default = the first choice, "L" - load marks). "Offset appendage (Y/N)? _" appears again. Key <ENTER> and key <ENTER> to the next query also. Now

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

is displayed. An <ENTER> here will default to the appendage entry point, any other address will be taken verbatim. Key in 402D<ENTER>. The file we could name would load then exit to MULTIDOS. Lets go back to the begining 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 sectord (F/D)? _'
```

appears on the display. An <F> or <ENTER> response will take us to the "Destination filespec? _" prompt, but a <D><ENTER> will ask us for drive, track and sector. A direct dump file has no load marks. All of the BOOT/SYS files on the other operating systems are direct dumped files. LO/CMD is a complex utility, but it is also powerful. SYSRES/SYS on your MULTIDOS diskette was created with this utility. Applications unlimited!

MEM/CMD Random Access Memory test.

(1)MEM<ENTER>	(MODEL I/III)
(2)MEM[B]<ENTER>	(MAX-80/MODEL 4)

(1) MEM/CMD will test random access memory from 4000H to TOPMEM.
(2) MEM/CMD will test random access memory from 0 to TOPMEM. If "B" is specified, then the 64K of expansion RAM is distructively tested.

If a memory bit fails, the byte and bit that failed will be displayed.

MEMDISK/CMD Random access psuedo drive.

(1)MEMDISK[t]<ENTER> (MODEL I/III)

t = number of tracks (2 to 16)

(2)MEMDISK[[:]d]<ENTER> (MAX-80/MODEL 4)

d = drive number (0 to 7)

(1) MEMDISK/CMD will let you use part of high memory as a data disk. MEMDISK/CMD will consume 2K of RAM for each track, "t", specified. If you do not specify a track count, MEMDISK/CMD will default to 16 tracks (32K).

EXAMPLE:

MEMDISK 8<ENTER>

This creates a MEMDISK with 8 tracks (16K). The logical drive assigned to MEMDISK will be the first available logical slot in the DCT. MEMDISK/CMD will adjust TOPMEM to protect itself. However, if you use an application program which does not recognize TOPMEM, you may clobber the MEMDISK. MEMDISK can be used as you would use any other drive, but remember MEMDISK does not maintain data if the power is removed or the computer is reset.

(2) MEMDISK/CMD will use the 64K of expansion RAM as a data or system disk. If drive zero or no drive number is specified, then MEMDISK/CMD will copy all of the necessary system overlays to MEMDISK, creating a system MEMDISK.

To remove a MEMDISK, key:

MEMDISK X<ENTER>

MEMDISK/CMD will remove MEMDISK from the DCT, then restore TOPMEM if TOPMEM wasn't changed after MEMDISK is loaded. If MEMDISK was a system disk, then MEMDISK/CMD will prompt you to insert a system disk into the prior logical drive 0, containing the system disk which loaded MEMDISK.

PRT/CMD Forms filter.

(1) PRT<ENTER> (MODEL I/III)
(2) PRT b1[,b2][,b3][...b10]<ENTER> (MAX-80/MODEL 4)

(1) PRT/CMD is a high memory, menu driven forms filter and graphic character converter for the MODEL I and MODEL III. PRT/CMD is required to enable the library command FORMS. After PRT<ENTER> is executed, the prompt:

'Enter decimal code to be ignored. _'

will be displayed. Enter the decimal numbers (1 to 255), one at a time - for up to ten numbers, you do not want to go to the printer. If you do not want to filter any values, or have only entered a few, respond with just <ENTER> to the prompt, then

'Graphic conversion for EPSON printer (Y/N)? _'

Enter <Y> if you have an EPSON printer capable of printing the TRS-80 (tm) video graphic characters, otherwise enter <N>.

(2) PRT/CMD is a high memory sequential filter for the MAX-80 and MODEL 4. PRT/CMD will accept the first 10 values entered on the command line. If additional values are desired, re-issue the "PRT b1,b1,b2, etc." command until you have loaded all the desired values to filter.

RS/CMD Machine Memory Scanner.

. RS<ENTER>

RS/CMD scans memory from 0000H to FFFFH and attempts to locate an 8 bit byte or 16 bit word specified by the user. The first query will be:

'START? _'

Enter the starting address, in 4 hex digits followed by <ENTER>, then

'STOP? _'

will be displayed to the right of the starting address. Enter the ending address, in 4 digit hex followed by <ENTER>. Next RS/CMD will ask:

'Byte, or word search (B/W)? .'

For a byte, search press <B>. For a word search, press <W>. Next RS/CMD will ask you to enter the search target. Enter two (for byte search) or four (for word search) hexadecimal characters.

Next, for word scans only, the prompt:

'Enter auxiliary mnemonic _'

will be 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. If no auxiliary mnemonic is desired press <ENTER>. For the "L" (LOAD) command, the following question

'Immediate, or Direct (I/D) .'

will be displayed. If the response is "D", then:

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

will be displayed. Answer as desired.

Next the query:

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

will be displayed. Enter "A" or the desired register pair.

RS/CMD will display the hexadecimal locations where the specified byte or word is found.

SPOOL/CMD Variable RAM printer buffer. (MODEL I/III)

SPOOL<ENTER>

SPOOL/CMD has two questions.

'How many 256 byte BLOCKS for SPOOLING (1-99)? _'

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

'Enter MEMORY SIZE (decimal) you want to protect. _'

Use this to protect a high memory routine which does not protect itself by setting TOPMEM. Press <ENTER> to use TOPMEM.

The spooler will now commence operation and control of any printed output until <RIGHT-SHIFT><BREAK> is pressed. This will suspend output and the query:

'Buffer? _

will appear. Enter <Y> to save the buffer contents, or <N> to reset the pointers and send a carriage return character to the printer.
The next query to appear will be:

'SPOOL? _'

Enter <Y> if you want to continue spooling, or <N> to remove SPOOL/CMD.
when SPOOL/CMD is removed and nothing changed TOPMEM, the TOPMEM value previous to SPOOL/CMD loading will be restored.

TAPE/CMD Tape to disk transfer utility

TAPE<ENTER>

TAPE/CMD will transfer a 500 baud contiguous system program from tape to disk. The utility will prompt:

'ENTER "T" to read TAPE, or "D" to return to DOS? _'

Ready the tape deck and position tape to start of file. Enter <T> to begin loading the system tape. If the file on tape does not load properly or is not contiguous, an error message will appear and the tape load will exit. If the file loads properly, indicated by START, END and ENTRY points of the program being displayed, it is ready to be transferred to disk. The prompt:

'Is the PROGRAM "xxxxxx" to be modified (Y or N)? _'

will be displayed. This prompt wants to know if you want the program to load at another address, then shift to the START address.

If <Y> is entered, then

'Are interrrupts to be enabled or disabled (E or D)? _'

will be displayed. Enter <E> if the interrupts are to remain enabled, or enter <D> if the interrupts are to be disabled before the program executes.

'Enter new base address in HEX? _'

Enter a new base address greater than 4CFFH, and GREATER than the programs start address, enabling MULTIDOS to load the program to a nonconflicting area with DOS. An offset appendage is added to the disk file to transfer the program to the START address. The next prompt:

'Initialize LEVEL II type DCB and RST vectors (Y or N)? _'

will be displayed. Respond <Y> if the program requires a LEVEL II environment, or <N> if the program does not require a LEVEL II environment. Whether the program is to be modified or not the following prompt will appear:

'Is a new FILESPEC required (Y or N)? _'

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

'Filename please? _'

will appear if you choose to give the program a new filespec.
Once all prompts are answered the program will be transferred to disk.

VFU/CMD Versatile file utility.

VFU<ENTER>

VFU/CMD provides for five frequently needed disk operations:

(1) copy a file(s) from one disk to another,
(2) execute a file,
(3) directory printout,
(4) move a file(s) from one disk to another, and
(5) purge a file(s).

VFU/CMD will calculate the amount of free memory upon entry. If enough memory is available, VFU/CMD will load Open/DOL, Close/DOL, and Allocate/DOL into high memory, enabling the user to copy, move, or purge without a system disk in drive zero. If insufficient memory is available, then the message:

```
'Insufficient MEMORY for C/M/P
 without a SYSTEM disk in drive zero.
 Press any key to continue.'
```

will appear, warning the user to keep a system disk in drive zero.

VFU/CMD is ready for your command when:

```
'Versatile File Utility - 8.x (c) C. E. C. 1984

            Press          Action
             "C"           Copy
             "E"           Execute
             "H"           Hard copy
             "M"           Move
             "P"           Purge
 Choice _'
```

is displayed. Key in your choice.

The <CLEAR> key will exit VFU, whenever the winking cursor is displayed.

VFU - COPY COMMAND - <C>

The copy command can be used to copy files from TRSDOS (tm) Model III double density diskettes to a non-TRSDOS (tm) disk.

To copy files press <C>. VFU/CMD will respond with:

```
'Condition (A/C/D/E/O/S)? _'
```

This is a means to select the conditions to meet before file copying can take place.

A = any condition.
C = create only. Only copy file(s) if the file(name) do not exist on the destination disk. Useful to avoid overwriting a file with the same name.
D = different date. Only copy file(s) if the existing file(name) on the destination disk has a different date.
E = earlier date. Only copy file(s) if the existing file(name) on the destination disk has an earlier date. Useful to avoid overwriting a later version of the same file(name).
O = overwrite. Only copy file(s) if the file(name) already exist on the destination disk.
S = size difference. Only copy file(s) if the file(s) on the destination disk are different in file size. (different EOF)

The next prompt will be:

´ Press "S" for selective, or "T" for total. _´

To select the files to be copied, press <S>. To copy all files, press <T>. The next prompt will be:

´Include "INVISIBLE" files? _´

Press <N> or <Y> as appropriate.

´Include "SYSTEM" files? _´

Press <N> or <Y> as appropriate. Finally, VFU/CMD will request 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, then press the number for the logical destination drive.

Press <BREAK> at any of the above prompts to return to the "Choice" prompt.

If the selective option was chosen, the directory will be displayed with the winking cursor next to the first filspec name. The arrow keys will move the winking cursor in the arrow direction. If you want to copy that file press <Y>. A "+" symbol will appear in front of the filename (file marked) indicating this file is to be copied, and the winking cursor will move to the next file. If you do not want to copy a file, press <N>, or the <SPACE-BAR>, and the cursor will move to the next file. During the file selection process, the shift left arrow will reposition the cursor to the first filename removing all "+" symbols. To remove a "+" symbol from a single file, use the arrow keys to position the cursor over the "+" symbol and press <N> or <SPACE-BAR>. If you move beyond the last file, you can press the left arrow key to continue file selection. If the total option was selected, the directory will be displayed with a "+" symbol in front of every filename. Use the left arrow to "un-mark" files, or the shift left arrow to "mark" all files.

After all selected files have been marked for copying, press <ENTER> to go to the prompt:

´Press "A" to abort, "ENTER" to execute, or "R" to repeat. _´

Press <ENTER> to start the copy function. To terminate the copy function, before all files are copied, hold down a shift key until VFU/CMD completes copying the current file.

After the copy process is complete, the directory of the destination disk will be displayed, indicating this command is complete.

VFU - EXECUTE COMMAND - <E>

Use the arrow keys to position the winking cursor in front of the file to be executed, then press <Y>. The EXECUTE command will immediately run the program when the <Y> is pressed. If the selected filespec does not have a "/CMD" extension, the EXECUTE command will load BASIC and attempt to run the filespec.

VFU - PRINT DIRECTORY COMMAND - <H>

To print a directory, press <H>. The next prompt will be:

´Include "INVISIBLE" files? _´

Press <N> or <Y> as appropriate.

´Include "SYSTEM" files? _´

Press <N> or <Y> as appropriate. Finally, VFU/CMD will request the drive with the following prompt:

´Drive number? _´

Press <BREAK> at any of the above prompts to return to the "Choice" prompt.

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 will be displayed on the screen. Finally, the utility will prompt with the following message:

´Press "A" to abort, "ENTER" to execute, or "R" to repeat. _´

A reply of <ENTER> will cause the directory to be printed. If a 10 character per inch printer is used, the printout width will be sized to fit inside a 5 1/4" diskette jacket.

VFU - MOVE COMMAND - <M>

The MOVE command performs the same as the COPY command except the file on the source drive will be purged after each file is copied to the destination drive.

VFU - PURGE COMMAND - <P>

To purge files from a disk, press <P>. The next prompt will be:

'Press "S" for selective, or "T" for total. _'

To select the files to be purged, press <S>. To purge all files, press <T>. The next prompt will be:

'Include "INVISIBLE" files? _'

Press <N> or <Y> as appropriate.

'Include "SYSTEM" files? _'

Press <N> or <Y> as appropriate. Finally, VFU/CMD will request the drive with the following prompt:

'Drive number? _'

Press the desired logical drive for file purging.

Press <BREAK> at any of the above prompts to return to the "Choice" prompt.

If the selective option was chosen, the directory will be displayed with the winking cursor next to the first file name. The arrow keys will move the winking cursor in the respective direction. If you want to purge that file press <Y>. A "+" symbol will appear in front of the filename (file marked) indicating this file is to be purged, and the winking cursor will move to the next file. If you do not want to purge a file, press <N>, or the <SPACE-BAR>, and the cursor will move to the next file. During the file selection process, the shift left arrow will reposition the cursor to the first filename removing all "+" symbols. To remove a "+" symbol from a file, use the arrow keys to position the cursor over the "+" symbol and press <N> or <SPACE-BAR>. If you move beyond the last file, you can press the left arrow key to continue file selection. If the total option was selected, the directory will be displayed with a "+" symbol in front of every filename. Use the left arrow to "un-mark" files, or shift left arrow to "mark" all files.

After all the selected files have been marked for purging, press <ENTER> to go to the prompt:

'Press "A" to abort, "ENTER" to execute, or "R" to repeat. _'

Press <ENTER> to start the purge function. To terminate purging, before all files are purged, hold down a shift key until VFU/CMD completes purging the current file.

After the purge process is completed, the revised directory of the disk will be displayed, indicating this command is complete.

ZAP/CMD Disk sector/memory 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 track, verify sectors, and fix MULTIDOS directories.

General ZAP/CMD information.

A. DEFAULTS
When ZAP/CMD is waiting for a variable response, ZAP/CMD will default to the first choice or lowest value if <ENTER> is pressed. Whenever an input field is complete, ZAP/CMD will automatically insert an <ENTER> for you. i.e. if the query is "DENSITY (S/D) ." (one character input required), pressing <ENTER> or <S> will terminate the query an continue.

B. 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.

C. BREAK KEY
Pressing <BREAK> will return the user to the previous query. When the "Choice" prompt is displayed, pressing <BREAK> will exit ZAP/CMD.

D. MODIFICATION
Pressing <M> after a displaying a page of MEMORY or a diskette SECTOR, will place the user 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 will move both cursors in the arrow direction. The "@" key will toggle the user between the HEX and ASCII modification mode, as indicated by the presence of "HEX" or "ASC" in the upper left hand corner for sector displays. Key in the new data for the bytes to be changed. Two keystrokes are required for hexadecimal changes, or one keystroke for ASCII changes. 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.

Copy sectors <C>

The "Copy sectors" option is obtained by pressing <C>. This option will copy sectors from any disk MULTIDOS will read to any other disk MULTIDOS will read. However, the user is cautioned to make sure the configuration byte for the source and destination disks have be set properly. These bytes are set correctly after a successful "DIR", or "File sectors" to BOTH disks. This option is menu driven, and won't overwrite some of the data if 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 will step 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 "DIR", or "File sectors"). TRSDOS (tm) double density diskettes (MODEL III - versions 1.1, 1.2, and 1.3; MODEL I - version 2,7 and 2.8) do not have a configuration in MULTIDOS, therefore; they are not supported. Use of this function requires user discipline in selecting the track number. ZAP unihibitedly accepts track and sector numbers from 0 to 255. The selection of the first sector is menu driven, and additional sectors are displayed by pressing:

Key pressed	Action
UP ARROW	Increase track count by one
DOWN ARROW	Decrease track count by one
RIGHT ARROW	Increase sector by one
<SHIFT> RIGHT ARROW	Increase sector by one
LEFT ARROW	Decrease sector by one
<SHIFT> LEFT ARROW	Decrease sector by one
"T"	Reselect track/sector
"S"	Reselect sector

File sectors <F>

The primary function of the "File sectors" option is to access files, regardless of protection, regardless of density, regardless of the operating system {except TRSDOS (tm) MODEL I double density}.

The "File sectors" option is obtained by pressing <F>. This option requires the filename to be entered exactly as the file name appears in the directory (every character and character case significant). Although ZAP will search all drives for a given filespec, it is suggested that the drive number be appended to the filename. i.e. FILENAME/EXT:D. This will greatly decrease the access time, because ZAP performs an analysis on each diskette prior to the HIT search for the filespec's hash code.

Enter the target file name at the

'Filespec .....'

prompt.

If ZAP/CMD finds the file, then

'Relative sector in xxxxxxxx/xxx:x (0 to dddd/hhhH) ....'

will be displayed. The "xxxxxxxx/xxx:x" will be replaced with the target file name, and "dddd/hhhH" will be the last relative sector in the target filename in decimal (dddd), and hexadecimal (hhhH). Enter the desired relative sector. (A file's first sector is relative sector zero [0]). <ENTER> will display the first sector of the file, and a number greater than dddd/hhhH will display the dddd/hhhH relative sector. Additional sectors in the target file can be accessed by use of the arrow keys.

Key pressed	Action
UP ARROW	Increase file sector by one
DOWN ARROW	Decrease file sector by one
[<SHIFT>] RIGHT ARROW	Increase file sector by one
[<SHIFT>] LEFT ARROW	Decrease file sector by one

Memory <M>

The "Memory" option is obtained by pressing <M>. when the "Address ....." prompt appears, enter the desired address. The memory address is changed by pressing the arrow keys.

Key pressed	Action
UP ARROW	Increase address by 256/100H
DOWN ARROW	Decrease address by 256/100H
RIGHT ARROW	Increase address by one
<SHIFT> RIGHT ARROW	Increase address by 16/10H
LEFT ARROW	Decrease address by one
<SHIFT> LEFT ARROW	Decrease address by 16/10H

Track format <T>

The "Track format" option is obtained by pressing <T>. This option will format and verify a single track on a floppy diskette. Don't be quick on the draw. The format density requires only one keystroke!

Verify <V>

The "Verify" option is obtained by pressing <V>. This option will verify that the target tracks - sector detailed - a readable. If the target diskette has mixed density tracks, "Verify" will switch density and keep getting up. "Verify" uses "logical tracks" when verifying a double density or double sided NEWDOS/80 (tm) diskette.

Fix directory <X>

The "Fix directory" option is obtained by pressing <X>. This option is designed to fix the directories for MULTIDOS - version 1.7 diskettes. "Fix directory" will also fix 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 "Fix directory" changes the GAT. The following information supplements the "Fix directory" option.

The directory, usually filenamed DIR/SYS, has information which 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 intergral sections referred to as tables:

1. The Granule Allocation Table (GAT) has the data on free and allocated space on the disk.
2. The Hash Index Table (HIT) has the hash codes for each active file on the disk.
3. The balance of the directory sectors contain the directory records (DIREC) for each file. Each directory record 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 which consists of a contiguous group of whole sectors. A PHYSICAL track is a one of many concentric circular recording surfaces on a side of a diskette 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 will vary with the hardware configuration of the disk drive. However, the number of granules per LOGICAL track will be fixed in accordance to the table below.

Diskette	Granules/Track	Granules/Cylinder	Sectors/Granule
5" SD/SS	2	2	5
5" SD/DS	2	4	5
5" DD/SS	3	3	6
5" DD/DS	3	6	6
8" SD/SS	2	2	8
8" SD/DS	2	4	8
8" DD/SS	3	3	10
8" DD/DS	3	6	10

SD=single density, DD=double density, SS=single sided, DS=double sided

TABLE 1

The GAT is further divided into two additional tables. Bytes X'00' through X'5F' are the assignment/free table, which 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.

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, which 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.

Byte X'CC' contains the number of logical cylinders, and byte X'CD' contains the number of physical cylinders for the disk. (MULTIDOS only!)

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.

```
    HEX 00  FFFF FFFF FFFF FFF8 F8FF FFFF FBFF FFFF ................
        10  FFFF FFFF FFFF FFF8 F8F8 F8F8 F8F8 FAF8 ................
        20  F8F8 F8FF FFFF FFFF 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 ................
    TRK 75  F8F8 F8F8 F8F8 F8F8 F8F8 F8F8 F8F8 F8F8 ................
    017 80  F8F8 F8FF FFFF FFFF FFFF FFFF FFFF FFFF ................
    11H 90  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
        A0  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
    SEC B0  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF ................
    000 C0  FFFF FFFF FFFF FFFF FFFF FFFF 2323 E042 ..... B......##.B
    00H D0  4D55 4C54 4944 4F53 3132 2F30 312F 3834 MULTIDOS12/01/84
        E0  0D20 2020 2020 2020 2020 2020 2020 2020 .
    RPT F0  2020 2020 2020 2020 2020 2020 2020 2020
```

Figure 1

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 which 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 extention (left justified and padded with spaces). This work space is processed through a hashing routine which 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 eliminiting 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 eight 32 byte directory records (DIREC). The maximum number of directory records is 256, regardless of the number of sectors alocated to the directory. (The HIT storage is limited to 256.)

Relative byte in the DIREC		Function
0	bit 7	
	0	File primary directory entry (FPDE).
	1	File extended directory entry (FXDE).
0	bit 6	If set indicates a system file.
0	bit 5	If set indicates a non-VFU file.
0	bit 4	If set indicates an active file.
0	bit 3	If set indicates an invisible file.
0	bit 2,1,0	Contain the files protection level.
1	bit 7,6,5,4	Unused by MULTIDOS.
1	bit 3,2,1,0	The month the file was created/updated.
2	bit 7,6,5,4,3	The day the file was created/updated.
2	bit 2,1,0	The year (offset by 1980) the file was created/updated.
3		The end of file byte.
4		The logical record length (LRL).

5 through 12	The filename padded to the right with spaces.
13 through 15	The extention padded to the right with spaces.
16 and 17	Update password hash code.
18 and 19	Access password hash code.
20 and 21	Last physical record.
22 through 29	Contain four segment fields.
30 through 31	Pointer to FXDE.

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 and/or the file is larger than 128 granule.

Bit 6 - SYStem files must have this bit set.

Bit 5 - If this bit is set, then VFU will not access the file.

Bit 4 - This is the bit to indicate if a file is active or not, regardless of the other bits in DIREC+00, or bytes in the DIREC. If this bit is set, then the file is active.

Bit 3 - If the bit is one, then the filename is hidden to directory display or print unless the invisible parameter is specified.

Bits 2-0 - Contain the protection level of the file.

DIREC+01

In a FPDE, this byte contains two zones. The first zone (bits 7,6,5,4) are not altered or used by MULTIDOS. The remaining bits, 3-0 are used to store the month portion of the date of the last file update.

In a FXDE, this byte is the reverse pointer to the previous FXDE or FPDE.

DIREC+01

In a FPDE, this byte contains two zones. The first zone (bits 7,6,5,4) are not altered or used by MULTIDOS. The remaining bits, 3-0 are used to store the month portion of the date of the last file update.

In a FXDE, this byte is the reverse pointer to the previous FXDE or FPDE.

DIREC+02

This byte contains the balance of the file's date information. Bits 3-7 contain the day of the month, and bis 0-2 contain the year offset by 1980.

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 bytes contain the filename. The filename is left justified and padded with spaces.

DIREC+13 through DIREC+15

These bytes contain the extension. The extension is left justified and padded with spaces.

DIREC+16 and DIREC+17

These bytes contain the hash code for the update password.

DIREC+18 ans DIREC+19

These bytes contain the hash code for the access password.

DIREC+20 and DIREC+21

These two bytes contain the total number of sectors occupied by the file.

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 in bits 7-5, and the number of contiguous granules less one in bits 4-0, for the cylinder in the first byte.

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'.

DIREC+02 through DIREC+21 are not used for FXDE's.

OTHER BYTES IN THE GAT SECTOR FOR EARLIER VERSIONS OF MULTIDOS

Bytes X'C0' through X'CB' GAT usage has varied with the MODEL as well as versions of MULTIDOS. These bytes were used for system and forms storage. Version 1.7 of MULTIDOS does not use these bytes.

BYTE

C0 Configuration pattern for DRIVE# 0

C1 Configuration pattern for DRIVE# 1

C2 Configuration pattern for DRIVE# 2

C3 Configuration pattern for DRIVE# 3

C4 Bits 7-5 are the power-up bits for SMO. Bits 4-0 is the number of disk I/O tries less one. (Version 1.6)
This byte is the OUT 254,(value) for speed-up mods on MODEL I version 1.4/1.5 or the byte for OUT 95,(value) for MODEL III version 1.3.

C6 The FORMS line width.

C7 The FORMS "S" parameter. (If pagination active the high bit is set.)

C8 AUTO invincible byte. (X'21' if AUTO command is invincible.)

C9 Keyboard defaults. The bits vary between versions and models.

CA Default cursor character.

CB The BCD representation of MODEL I version 1.5 or prior, MODEL III version 1.3/1.2/1.1/1.0.
Version 1.6 uses this byte for FORMS parameters.

MULTIDOS ERROR MESSAGES

HEX	DECIMAL	ERROR MESSAGE
00	0	No error.
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.
08	8	Drive not available.
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 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	Program not found.
20	32	Improper drive number specified.
22	34	Load file format error.
23	35	Memory fault.
25	37	Access attempted to protected file.
26	38	File has not been opened.
27	39	Drive configuration one volume.

When an error occurs involving the directory, MULTIDOS will generate two error messages. For example, if the disk in drive zero is write protected, and an AUTO command is issued, MULTIDOS will respond with:

```
Granule allocation table write error.
Write protected media.
```

Use the second error message in resolving the problem.

ERROR MESSAGE BREIF DEFINITION

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 thru the MULTIDOS directory read routine.

Data record not found during read/write. - The track, side, or sector number is not found. This error can be caused by a flawed disk or a diskette formatted with a different TPI 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 for DIREC detail.

Disk space full. - All granules have been assigned.

Drive not available. - Diskette not spinning in selected drive.

Drive not configured double sided two volume. - Side one selected for a logical drive with a single volume configuration.

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 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 given to MULTIDOS is not in the range of 0 thru 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 to 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 more than one unit past the end of file.

Parity error during read/write. - A sector read/write operation failed.

Program not found. - Filespec is not on any mounted diskette.

Write fault on disk drive. - Definitely a hardware problem.

Write protected media. - A write attempt to a write protected disk.

SUPERBASIC 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 addresses.

command = Any valid BASIC command.

A mandatory space is required after "BASIC" if any of the optional parameters are specified. A comma will indicate a multiple command.

EXAMPLES:

BASIC<ENTER>

SUPERBASIC will load with 3 file buffer areas allocated (289 bytes for each file buffer area), and all RAM up to TOPMEM available for use.

BASIC 4<ENTER>

SUPERBASIC will load with 4 file buffer areas allocated, and all RAM up to TOPMEM available for use.

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

SUPERBASIC will load with 2 expanded file buffer areas allocated (545 bytes each), capable of handling user defined record lengths (1 to 256), RAM from 60000 up will not be available to SUPERBASIC, and the program "PROG1/BAS" will be loaded by SUPERBASIC and executed starting with the first program line.

BASIC * Recover BASIC program.

BASIC *<ENTER>

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

CAUTION: You cannot go from SUPERBASIC to MULTIDOS, execute commands such as "DIR" and then use "BASIC *" to return. In such cases you should use the CMD"uuuuu" function from within SUPERBASIC.

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

BASIC ! Capture a BASIC program from RAM.

BASIC !<ENTER>

This unique command will transfer a BASIC program, previously in RAM, to SUPERBASIC. With a BASIC program in RAM, insert your MULTIDOS disk into drive 0 and press reset. When the MULTIDOS prompt appears, enter BASIC !.

BASIC # Recover a LEVEL II BASIC program.

BASIC #<ENTER>

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

MULTIDOS uses SUPERBASIC's CMD "W" function to enter LEVEL II BASIC. You may use CMD "W" with or without a program in RAM. If a program is in RAM, it will be transferred to LEVEL II BASIC with text retained. To exit from LEVEL II BASIC to SUPERBASIC with the program intact, key SYSTEM<ENTER>, answer the "*?" With "/16480", hold down the enter key and wait for the MULTIDOS prompt. Enter "BASIC #" as shown above.

BBASIC Enhanced SUPERBASIC.

BBASIC has all of the features of SUPERBASIC with the addition of single step, trace, variable review, and program pushing functions. Upon entering BBASIC, the <@> key becomes a control key. The "@" character is printed by pressing <SHIFT><SPACE-BAR>.

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

The functions will be described with an "@" symbol preceding a character. This represents pressing the <@> key along with the key following the "@".

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

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

@3 TRACE ON - PRINTER
This function will direct the trace to the printer in the format space, 10k digit, 1k digit, 100's digit, 10's digit, and unit digit. Zero suppression is used and each line requires six character positions. The trace information will show the complete program flow, but will only be printed when an entire print line is available, unless a character at a time printer is being used. e.g. TELETYPE.

SINGLE STEPPING

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

@4 SINGLE STEP OFF
This function will turn off the single step function and allow your program to run as normal. If the trace function was in use, it will continue to function until turned off via <@1>.

@5 SINGLE STEP TO THE END OF LINE
This function will execute one program line then wait until any key is pressed. The trace to video display mode will also be initiated to show the line numbers being executed. The tracing function can be disabled by a <@1>, while the single step mode continues.

@6 SINGLE STEP INSTRUCTION
This function will execute one BASIC instruction then wait until any key is pressed. The trace to video display mode will be initiated to show the line numbers being executed. This function can be useful, but be wary of using it if a program contains lines such as:

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

To single step through this loop would require 300 presses of a key. Instead use SINGLE STEP TO END OF LINE for this type of line.

@7 SINGLE STEP WITH TIMED WAIT
This function will execute one program line or instruction, pause for a predetermined amount of time, then continue. The trace to video will be initiated to show the line numbers being executed. <@5> and <@6> become sub-functions after <@7> is initiated. Pressing <@6> after <@7> is initiated will cause the delay to occur at an instruction separator, in addition to the end of a line. Pressing <@5> will cause the delay to occur at

the end of a line only. To speed up execution (decrease delay) press <@up-arrow> (halves delay). To slow down execution - increase delay - press <@down-arrow> (doubles delay). The delay has nine settings from approximately 4 milliseconds to 0.9 seconds at normal CPU speeds.

BREAK POINTS

The trace and single step functions previously described can be invoked while a program is running by inserting as many POKE instructions in the program where the functions will initiates. The following codes are used:

FUNCTION	POKE 16667,	KEY equivalent
TRACE OFF	1	@1
TRACE ON - VIDEO	2	@2
TRACE ON - PRINTER	3	@3
SINGLE STEP OFF	4	@4
SINGLE STEP TO END OF LINE	5	@5
SINGLE STEP INSTRUCTION	6	@6
SINGLE STEP WITH TIMED WAIT	7	@7

EXAMPLES:

If normal program execution is desired until line 1540, then single step with trace to the screen required, insert, just prior to line 1540, the instruction "POKE 16667,5".

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

Multiple POKE's are permitted.

```
POKE16667,7:POKE16667,6:POKE16667,1
```

This will invoke "SINGLE STEP WITH TIMED WAIT" between instructions with the trace disabled.

SELECTING/REVIEWING VARIABLES

BBASIC will suspend program execution to select or review variables then resume execution and restore the display. If there is insufficient memory to save the contents of the video display, a graphic block will appear in the upper right hand portion of the screen - to the left of the line number trace region, and the program continues.

SELECTING VARIABLES

.@N = select variables for review

Pressing <@N> selects the variables to review during program execution. This command can be entered at any time, before the program is ran or during program execution. After invoking <@N>, the screen will clear and the query "Length? " will be displayed. Respond from the following choices:

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

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

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

NOTE: The name length includes all characters. The variable name A$(21,5) is considered to have a length of eight. F(R(3,8)) is nine characters in length. After successfully entering a variable length, the message "Input." will be displayed and all previously entered variable choices will be erased. <@N> accepts simple as well as complex variables. Such as:

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

Any number of parentheses are allowed, and algebraic hierarchy is maintained. Although erroneous variable names such as A$3 or A(3H) are not rejected, they will cause errors later when a review of the variables is attempted.

Once all of the variables to review are entered, press <BREAK> to continue with the review. If the maximum number of variables allowed are entered,, BBASIC will automatically proceed with the review. At this point BBASIC invokes an <@O> as described below.

REVIEWING VARIABLES

@O = review the selected variables

Pressing <@O> during program execution will immediately save the contents of the video display and the message:

´<C>hange <D>elete <I>nsert.´

will appear along with the first variable for review and its value. Variables are displayed in the order entered with the <@N> function.

Pressing <CLEAR> will cause the message ´End.´ to appear. If the reviewing of the variables is complete, press <CLEAR> again to resume program execution and return the original video display. Any other key will start at the beginning of the variable rewiew.

Pressing <C> will erase the last variable displayed, and put the user in an input mode. The new variable selected and its value will be displayed immediately after entered. Remember, the variable name is limited in length per the original choice when <@N> was selected.

Pressing <D> will delete the last displayed variable.

Pressing <I> will insert a variable PRIOR to the last displayed variable.

Pressing any key other than <CLEAR>, <C>, <D>, or <I> will advance the display to the next variable selected.

If an attempt to review a variable whose subscript is out of range or with an illegal name, the message ´Redo!´ will be displayed and the <C> command will be invoked. A valid variable 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, this array will be dimensioned for eleven elements (0-10). If the program subsequently attempts to dimension this array via the "DIM" instruction, an error will occur. Dimension all used arrays before review.

STACKING BASIC PROGRAMS

BASIC programs may 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 current BASIC program in high memory
@: = Recall the last saved program from memory
@8 = Append the last saved program to the current program
@9 = Append the next to last saved program to the current program
@0 = Recall the next to last saved program

Pressing <@-> will copy the resident program into high memory and adjust memory size to the beginning of the saved program, thus protecting it from BASIC. The original program will be left available in BASIC RAM, if memory permits. Since memory size is adjusted, subsequent saves can be made as desired. When a program is saved, a tall vertical bar will appear in the upper right hand corner of the video display - to the right of the line number trace region, indicating a program has been saved into high memory and a program is in BASIC memory for user execution or modification. If insufficient memory is available to save a program and also maintain it in BASIC RAM (i.e. saving a 30K program in a 48K machine), the program will be saved and a "NEW" invoked. This condition will be indicated by a clear screen with a small block in the upper right of the video display.

Pressing <@:> will recall the last saved program from high memory, overlaying the resident program and adjusting memory size. If no saved program remains, the error message "NOTHING TO POP" will be displayed.

Pressing <@0> will recall the next to the last saved program from high memory, overlaying the resident program and adjusting memory size. <@0> provides a means to switch the resident program with the last saved program by saving the current program via the <@-> command and then recall the next to the last program via the <@0> command.

Pressing <@8> will recall the last saved program and append it to the resident program.

Pressing <@9> will recall the next to last saved program and append it to the resident program.

Line number sequence is mandatory for proper execution of the appending commands. The recalled program should have its lowest line number greater than the highest line number in the current program in BASIC RAM. The <@8> nad <@9> commands append, they do not merge.

SINGLE KEYSTROKE COMMANDS

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

The single keystroke commands are recognized only if they are the first keystroke after the BASIC prompt, '>', appears. If some other key is pressed, press <BREAK> to enter a single keystroke command.

SINGLE LETTER COMMANDS

```
                A = AUTO
                C = CONT
                D = DELETE
                E = EDIT
                I = AUTO(current line + 1),1 [INSERT]
          K"filespec = KILL"filespec"
                L = LIST
          L"filespec = LOAD"filespec"
           Mln1,ln2 = move ln1 to ln2
             Mln1,. = move ln1 to current line
             M.,ln2 = move current line to ln2
           Nln1,ln2 = duplicate ln1 as ln2
             Nln1,. = duplicate ln1 as current line
             N.,ln2 = duplicate current line as ln2
                P = list page from current line
              Pln = list page from line ln
                R = run program
          R"filespec = RUN"filespec"
          S"filespec = SAVE"filespec"
```

The above commands are terminated with <ENTER>.

A line number followed by <ENTER> will list the line. To delete a line you must use D or DELETE, followed by the line number (optional) then <ENTER>.

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

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

EXAMPLE: This BASIC program is in memory and the current line is 20.

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

If L<ENTER>
Then Line 20 will be listed.

If L-.<ENTER>
Then Lines 10 and 20 will be listed.

If L.-<ENTER>
Then Lines 20, 30, and 40 will be listed.

If .
Then Line 20 will be listed.

&H and &O Hex and octal constants.

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

This command will permit the use of hexadecimal (base 16) or octal (base 8) constants within a program. The "H" is optional in SUPERBASIC.

EXAMPLE:

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

CMD"C" Space compression.

```
CMD"C"<ENTER>
```

This command eliminates unnecessary spaces and linefeeds from the resident BASIC program, resulting in more memory and faster execution.

CMD"D" Load and execute DEBUG.

```
CMD"D"<ENTER>
```

This command loads and executes DEBUG. (See library command DEBUG)

CMD"E" Disk I/O error.

```
CMD"E"<ENTER>
```

This command displays the last disk I/O error occurring in BASIC.

CMD"K" Zero numeric array. Null string array.

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

vv = Any valid variable name
 0 = dummy argument
```

This command zeroes the array vv(dim[,dim...]).

EXAMPLE:

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

A% is dimensioned to a 280 element array. After line 700 is executed, A% is still a 280 element array, but each element has the value of zero.

CMD"L" Delete array.

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

vv = Any valid variable name
0 = dummy argument

This command deletes the array vv(dim[dim...]. After using this command the array can be redimensioned.

EXAMPLE:

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

A% is dimensioned to a 280 element array. After line 700 is executed, the array A% no longer exists and memory consumed by the array is available.

CMD"O" String sort. (See CMD"Q")

CMD"P" Pack program lines

CMD"P"<ENTER>

This command packs BASIC program lines together and maintains program logic. Upon invoking the command the prompt:

'Maximum line length _'

will appear. Enter the maximum number of characters to be packed into one program line. Enter any value between 0 and 65535. A 0 packs lines to a maximum of 65536 characters. The default value is 240. After the line length is selected, a second prompt will appear:

'First _'

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

'Last _'

Enter the last to be packed. The default line is the last program line.

If no errors occur, the program will be packed using the following rules:

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.

CMD"Q" String sort.

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

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

This command sorts of a string array in accordance with the sign of n1. If n1 is positive, the sort will be in ascending or alphabetical order. If n1 is negative, the sort will be in descending or reverse alphabetical order. VERSION (1) is used to sort a single dimensioned array. The array will be sorted up to the n1th element, including the 0th element. Version (2) is used to sort a two dimensional array with n2 indicating which column of the array to use as the sort key.

EXAMPLE:

```
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 will be:

```
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" will 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.

CMD"R" Enable interrupts (MODEL I only).

CMD"R"<ENTER>

This command enables the interrupts.

QCMD"S" Return to MULTIDOS.

CMD"S"<ENTER>

This command returns you to the MULTIDOS operating system.

CMD"T" Disable interrupts (MODEL I only).

CMD"T"<ENTER>

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

CMD"U" Unpack program lines.

CMD"U"<ENTER>

This function unpacks a program into as many single statements as possible, inserts spaces around each key word, and renumbers the program 10, 20, etc.

CMD"V" Scalar variables.

CMD"V"<ENTER>

This command displays all assigned scalar variables and string equivalents in the order they were created. The screen will clear and up to 12 variables will be displayed. Press any key to display an additional variable or <ENTER> to display up to 12 more. This function may be embed in a BASIC program, however, user intervention will be required to continue.

CMD"W" Transfer to LEVEL II.

CMD"W"<ENTER>

This function transfers the resident BASIC program to LEVEL II BASIC while retaining memory protection. Refer to "BASIC #" for direction on re-entry to SUPERBASIC from LEVEL II BASIC.

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

CMD"X" Remove REMark statements

CMD"X"<ENTER>

This command removes REMark statements from a BASIC program in RAM. If a line consists of only a remark statement, then the entire line will be removed. After the remark statements are removed, the renumber function is called to check for undefined lines.

CMD"uuuuu" Execute a MULTIDOS function from SUPERBASIC.

CMD"uuuuu"<ENTER>

uuuuu = Any valid MULTIDOS command

This command executes any valid MULTIDOS command, including BASIC, from within the SUPERBASIC environment. The command can be used in the direct mode or as a statement within a BASIC program. The CMD"uuuuu" function will temporarily set TOPMEM to a value to protect the BASIC program, and requires approximately 6300 free bytes to execute.

EXAMPLES:

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

(2) CMD"BACKUP":GOTO 230
BACKUP/CMD will be loaded and executed. Upon completion, they BASIC program will resume execution at the next instruction.

DEF FN Define function. {The space between DEF and FN is optional}

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

xxx = Name of the function (any valid variable name).
uuu = Variables to be used to define the function.
www = An expression or formula involving the variables 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 will be the same as the type of variable used to name the function. In addition, the variables used in the DEFFN statement, are local variables and do not effect the variables used elsewhere in the program.

EXAMPLE:

```
10 DEF FN 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, but the variables in line 20 and 30 are N and D. K and L may be used in the program and have no effect on FN Q, nor does FN Q definition using K and L have any effect on the variables K and L.

DEFUSR Define entry address of USR routine.

```
DEFUSRn = aaaaa

    n = The digit 0-9. If n is omitted 0 is used.
aaaaa = The entry address to a machine language routine.
        aaaaa may be any numerical expression, including
        constants, variables, or functions.
```

EXAMPLE:

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

Defines 32000 decimal to as the entry point to a USR 5 call.

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 for which you are searching,
        or (2) The actual substring for which you are searching
```

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

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

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

LINEINPUT Input a string from keyboard.

```
LINEINPUT["message";]vv$

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

This BASIC statement is the method to input a complete line from the keyboard, including leading spaces, punctuation, and line feeds. This statement nulls the variable, and does not print a question mark. A space is permitted between LINE and INPUT (LINE INPUT).

LIST Display program text. (Model I enhancement)

```
LIST<ENTER>
```

This command has been modified to show graphic characters included in quoted text. Editing graphic lines will maintain the graphics, as long as the "A" command is not used. (If you accidentally enter an "A" command, use the "Q" command to abort.)

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$ will be 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"	1234567Z

TIME$ Get current RAM date and time.

```
T$ = TIME$
```

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

USRn Execute a machine code routine.

USR[n](val)

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

This function transfers control to a machine language subroutine which was previously defined with the DEFUSRn statement. When a USR function is encountered in a statement, SUPERBASIC transfers the program counter (PC register) to the address specified by the corresponding DEFUSR statement. When the specified USR function is complete, via a RET or JP 0A9A instruction, the BASIC program will continue at the next statement following the USR function.

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

The last USR statement executed has the LSB and the MSB entry points in 408EH [16526 dec] and 408F [16527 dec]. You can circumvent the extended USR function by poking C9H [201 dec], into RAM location 41A9H [16809 dec], and poke the subroutine's address into 16526 and 16527 decimal to execute a LEVEL II program with USR calls in SUPERBASIC. Add the instruction: POKE 16809, 201 prior to executing the USR subroutine. This modification will enable SUPERBASIC to execute a USR function developed for LEVEL II.

SUPERBASIC OVERLAY FUNCTIONS

FIND Find ASCII characters.

Ftar<ENTER>

This command will find all occurrences of "tar" in a BASIC program. The display will indicate the line the "tar" is found on, and if more than one occurrences are on this line a "/" is printed followed by the number of occurrences. Both leading nd 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 in line 90 three times.

EXAMPLE:

F in <ENTER>

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

GLOBAL EDITING Mass editing BASIC program.

-<ENTER> {a minus sign followed by <ENTER>}

GEDIT makes mass changes to a BASIC program, by performing the following:

Change all or part of variable names.
Change all or part of constants, data list items, or strings.
Change graphic codes as "CHR$(x)" into packed strings (x = 128-191).
Change space compressio codes as "CHR$(y)" into packed strings (y > 192).
Merge adjacent line numbers into one long line.
Split one long line into two shorter ones.
Change key words.

GEDIT - General changes

-<ENTER>

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

```
          'L              T

    T = _'
```

This is the target entry mode. The rules for the target will be discussed after the replacement is defined. After a target is entered, you will be 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 will limit a change 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 will return 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 will prompt 'Use it?'. This the last chance to correct an error in the entries. An <N> response will put the GEDIT in the replacement mode, an a <Y> response will start GEDIT to search the program for the target and make the changes. The screen will show 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 will be displayed. To exit from GEDIT, at this point, press <BREAK>.

If the replacement is larger than the target, the program will be increased in size by that difference for each occurence. If you run out of memory ,an "Out of MEM in XX" error message will be displayed, and all changes will be made up to the point (somewhere in line XX) where memory was insufficient.

Rules for target:

To change a constant, a variable name, or item in a data list which is not enclosed in quotes, respond with the item when the target is requested.

(In all of the following examples, T is the target, R is the replacement, and "->" means changes 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->CA

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

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)

To change only the first character of a multi character variable, enter the target character followed by a single quote (X´).

EXAMPLE: (Change all occurrences of the first "A" to "H")

T = A´ and R = H

AA->HA, AB->HB, AC->HC, QD->QD, AE->HE, AA$->HA$, BA$->BA$

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 (´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->A, AX->AX

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->TE

GEDIT - Changes to key words:

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

GEDIT - Building compressed strings

Program containg many "CHR$(x)+CHR$(x)" statements, will be shorten by building compressed string. To build a compressed graphics string in place of a CHR$(x) type lines, enter the "+" character for the target. To build a compressed string with space compression codes, enter the "*" character for the target. 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$(x) OR CHR$(y) cannot contain blanks within the parentheses. i.e. CHR$(191) is not acceptable and will not be changed.

EXAMPLES:

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

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

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

GEDIT - Merging 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 contains lines 1,2,3,4, & 5, then you merge line 3 to line 2, an error would result at run time if line 5 originally contained "GOTO3", because line 3 no longer exists. Do not append to a line which contains an open quote (10 A$="THIS IS). Close the quote first, then merge (10 A$="THIS IS").

GEDIT's merge can create lines greater than 255 bytes in length, which will execute properly, but can neither be properly listed on the screen nor edited. If necessary, use CMD"U" to seperate the line. To merge, respond with the "/" character and the line number to be merged as the target.

EXAMPLE:

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

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

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

GEDIT - Splitting lines

GEDIT will 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, & 60, and you split line 20 into lines 20 and 45, lines 30 and 40 will not be found by any reference instructions such as "GOTO 30", "GOSUB 40", etc. However, if the program flow is such that BASIC would normally process the next statement in RAM after the new lines 20 and 45, lines 30 and 40 will be processed. BASIC would, in the absence of any branching instructions, process lines 20, 45, 30, 40, and 50 in that order.

The split point must be directly behind a colon (:) in the program line. To 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, the line will be 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$="FGH"
35 PRINT B$
```

REFERENCE Cross reference variables, and integers up to 9999999.

;[p][[#]xxx]<ENTER>

p = listing directive. If p = "*" the reference listing is to the display. If p = "$" the the reference listing is to the printer and the display.

xxx = Reference target:
(1) A one or two character variable name without a type suffix.
(2) An integer number which may be a line number or a value used in the program.
(3) A 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 will be displayed sequentially, with each additional press of <;> alone.

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

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

EXAMPLES:

;*<ENTER>

All integer and variable references are displayed on the screen.

;K<ENTER>

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

;$G<ENTER>

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

;#PRINT<ENTER>

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

;$#<ENTER>

All key words will have their references directed to the video and line printer.

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

```
/n  where n = the number of references to the target 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.

    the "n" will not show if it has a value of 1.
```

RENUMBER Renumber a BASIC program.

```
:[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. eee must be >= sss and has a
      default value of 65529. This is the last line renumbered.
```

RENUMBER checks for missing or invalid line numbers in a program, recovers a "NEWED" program, and renumbers all or part of a program. To recover a program immediately after a "NEW", key in :<ENTER>.

Error checking only - for undefined lines, etc. - is accomplished by leaving out all arguments. e.g. :<ENTER>.

Error types are:

```
11111/nnnnn/U   Line number 11111 has line nnnnn undefined.
                (line number nnnnn does not exist)
11111/S         Line number 11111 contains a syntax error.
11111/O         Line number 11111 contains an overflow line number.
                (line number > 65529)
```

EXAMPLE:

20/290/U means line 20 has a reference to line 290 and there is no line numbered 290 in the resident BASIC program. The reference may be in the form of (1) GOTO 290 (2) GOSUB 290 (3) ...THEN 290 (4) ....ELSE 290 (5) RESUME 290 (6) ON n GOTO 280,290 (7) IF ERL = 290, etc.

RENUMBER will check your program for errors before renumbering. If any errors are found they will be displayed and the program will be 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
```

An attempt to renumber this program, will produce:

```
20/290/U   40/S    80/O
Function completed
READY
>_
```

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 fixing the above errors, the command ":50,1,50,69<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. Lets renumber this program in order for the INPUT statement in line 30 to get processed. This will be accomplished by MOVING lines 30 to 70 inclusive to less than 20. Lets key in ":11,1,30,70<ENTER>", and the results will be:

```
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.

SUPERBASIC FILE MANIPULATION

KILL — Delete a file from a diskette.

KILLvar$ or KILL"filespec"

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

This command deletes a file from the directory. If the statement is within the program, the file should be closed first. If KILL is executed in the command mode, KILL will close all files before it attempts to remove the targel filespec.

LOAD — Load a BASIC program to RAM.

LOADvar$[,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 will load and run the program without closing any files opened via an OPEN statement.

MERGE — Combine two programs in RAM.

MERGEvar$ or MERGE"filespec"

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

This command combines the program currently in RAM with the program lines in var$. If a line number in var$ coincides with a line number in the resident program, the resident line will be overwritten.

NAME — Load an execute a program keeping variable values.

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

This command chains programs which are too large to fit in memory. This command load and execute the file var$ or filespec beginning with the lowest line number as if it were a new BASIC program. The previous variables will remain intact, except for DEFFN, strings created via READ, or string assignments directly in a BASIC program. The ",B" option will load and run the program without closing any files opened via an OPEN statement.

RUN Load a BASIC program and execute.

RUNvar$[,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 will load and run the program without closing any files opened via an OPEN statement.

SAVE Save a BASIC program to disk.

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

A = save in ASCII

This command transfers the current BASIC program to the diskette.

SUPERBASIC FILE ACCESS

OPEN Set the mode and assign a file buffer area to a filespec.

OPEN mode, buf, var$

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

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

buf = the file buffer area, in the range of 1 to "ff", to be assigned to var$". "ff" is the number of file buffer areas made available during BASIC initialization. Only the "I" mode can have more than 1 file buffer area assiged to a filespec.
var$ = a string defined as a filespec.

OPEN mode, buf, var$, udl

udl = user defined record length.

Mode can be "R" or "D". If mode = "D", udl is obtained from the filespec.

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

OPENQ$,N,P$

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.

CLOSE Unassign a file buffer area by buffer number.

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

buf = an expression with a value of 1 to 15. If buf is omited, all open file buffer areas will be 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".
```

Any statement or command generating a CLEAR will close all files. For example the following will close all files: NEW, LOAD/RUN (without ",R"), MERGE, DELETE, EDIT, and CLEAR.

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 from a disk file. Each data item is read sequentially from the beginning of the file. To INPUT# data sucessfully, you need to know how the data was placed on the disk. If the data is going to a variable, BASIC ignores leading spaces, and begins the INPUT# with the first non-space character. If the data is going to a numeric variable, a comma, trailing space or <ENTER> in the file will terminate the variable assignment. If the data is going to a string variable, a comma (not enclosed in quotes), or <ENTER> (not preceded with a line feed) will terminate the input assignmen after the first non-space is encountered. If non-quoted text is encountered, followed by quoted text, then the qouted text also becomes part of the variable assignment.

LINEINPUT# OPEN"I" read a line of text command.

LINEINPUT#buf,var$

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

LINEINPUT# will read all characters from the current position up to and including an (ENTER> (not preceded with a line feed), up to the end of file, or 255 characters, whichever comes first. LINEINPUT# differs from INPUT# in that <ENTER>, not preceded with a line feed, is the terminator.

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 express on to be evaluated and written to the diskette.
```

The delimiter, "m", can be a semi-colon ";" or comma ",". The use of these delimiters will determine the format on the diskette. If the comma is used, the "items" will be zoned in 16 byte areas on the diskette, 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:

```
X$ = "JONES, TOM"
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$[,len2 AS str2$...]

  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$2 = the variable name of the second field.
  ... = subsequent "len AS str$" pairs for the balance of the buffer
        area size. The size is determined by the user for created
        files or by the file itself for existing files.
```

More than one set of variables can point to the same buffer area.

EXAMPLE:

The data is stored in the following format. The first 64 bytes contain the cliens name and address. The next 14 bytes contain the phone number.

```
FIELD 1, 56 AS AD$
FIELD 1, 78 AS AP$, 122 AS HG$
```

These statements assigns the first 56 bytes to AD$, the first 78 bytes to AP$, and bytes 79 thru 200 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. The first field statement could be

```
FIELD 1, 56 AS AD$, 14 AS PN$
```

to capture the phone number.

MKI$, MKS$, and MKD$ Data converters. Numbers to string.

Since all of the data items for RANDOM I/O are defined as strings, numeric data must be converted to strings.

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.

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 will change the variable type flag from integer to string in the variable PAY$, creating a 2 byte string.

```
LSET TAX$ = MKD$(K%*10)
```

This statement will produce an 8 byte string, TAX$.

CVI, CVS, and CVD Data converts. String to numbers.

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

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 variables WF$ and FR$ were assigned to a buffer area via the FIELD statement.

LSET, RSET Place data in a RANDOM buffer area.

```
LSETstr$ = exp$          RSETstr$ = exp$
```

str$ = variable assigned to the buffer area via FIELD
exp$ = the assignment for str$

EXAMPLE:

```
LSET WF$ = MKI$(J%)
RSET DR$ = B$ + C$
```

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

PUT RANDOM write statement.

PUT[#]buf[,rec]

buf = a file buffer area 1 to 15.
rec = the record number (1 to 32767, and -32768 to -1). If rec is omitted, the current record is used. The current record is one unit greater than the last record written.

The PUT statement will move data from the file buffer area to a specified record in a file. Each PUT statement will write to a file if the record length is 256 bytes. However, if record length other than 256 is used, (only possible using the V parameter at BASIC initilization), then BASIC will wait until the I/O buffer area 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 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 will increase the file by the necessary length to accommodate the record.

EXAMPLE:

```
100 OPEN"R",1,"TEST/DAT",40
110 FIELD 1, 2 AS FD$, 8 AS MD$, 30 AS XR$
120 INPUT "What is your age";AG
130 INPUT "What is yout name";NM$
140 PRINT "OK ";NM$;", How many bytes can a Z-80 address";
150 INPUT;BT!
160 PRINT "I will keep a record of this data"
170 F$=MKI$(AG)
180 LSET FD$ = F$
190 LSET MD$ = MKD$(BT!)
200 PUT 1,1
```

GET RANDOM record reader.

GET[#]buf[,rec]

buf = file buffer area 1 to 15.
rec = the record number (1 to 32767, and -32768 to -1). If rec is omitted, 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 or OPEN"D",buf,filespec)
(2) FIELD the buffer area (FIELD buf, 1 AS X$, 233 AS Y$, ...)
(3) GET the record
(4) CVI, CVS, CVD as necessary.

EXAMPLE:

```
100 OPEN"D",1,"TEST/DAT"
110 FIELD 1, 4 AS JK$, 8 AS KL$, 30 AS LM$
120 GET 1, 1
130 BYTES! = CVD(KL$)
140 AGE%  = CVI(JK$)
150 COMM$ = LM$
160 ...
```

EOF End of file detector.

EOF(buf)

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

LOC File location indicator.

LOC(buf)

This function returns 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).

SUPERBASIC ERROR MESSAGES AND ERR CODES.

ERR	ERR/2+1	MESSAGE
100	51	Field organization exceeded the logical record length.
102	52	Internal error, use CMD"E" for specific.
104	53	The buffer number is not available or was used improperly.
106	54	File is not in specified directory.
108	55	Incorrect file mode.
110	56	File previously opened.
112	57	Disk read error, use CMD"E" for specific.
114	58	Disk write error, use CMD"E" for specific.
116	59	File already exists.
118	60	End of file encountered.
120	61	Drive not available.
122	62	Disk space full.
124	63	"EOF" reached before any characters read.
126	64	Bad "PUT" or "GET" parameter.
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	Incorrect password used to access file.
140	71	Full directory. File cannot be extended.
142	72	The file has not been opened.
144	73	Undefined function.
146	74	File not found.

ERROR MESSAGE BREIF DEFINITIONS

Access mode differs from open mode. -

Bad "PUT" or "GET" parameter. -

Directory space full. - All DIREC slots are used. See ZAP for DIREC detail.

Disk read/write error, use CMD"E" for specific. - Use CMD"E", then refer to the DOS error message definitions.

Disk space full. - All granules have been assigned.

Drive not available. - Diskette not rotating in specified logical drive.

End of file encountered. - An attempt is made to read 1 unit past the EOF.

"EOF" reached before any characters read. -

Field organization exceeded the logical record length. - The sum of the "len" for the "len AS str" pairs in a FIELD statement is greater than the logical record length.

File already exists. -

COMMON - MODEL I and MODEL III (All numbers are HEXADECIMAL!)

4015 thru 401C [KBDCB] - KEYBOARD DCB.

401D thru 4024 [VODCB] - VIDEO 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.
DCB+6 VIDEO cursor character.
DCB+7 Space compression if 0. Special characters if 1. (MODEL III only)

4025 thru 402C [PRDCB] - PRINTER 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 thru 402F [TODOS] - Vector to indicate the completion of a DOS command, error or no error.

4030 thru 4032 [DESS] - Vectors to 402D after resetting Stack Pointer to the DOS stack, 41E5, and prompting the user "Insert SYSTEM <ENTER>". If in BASIC, vectors to function similar to 6CC on MOD I's.

4053 [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

4054 [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
6	PR output to SO	PR output to parallel port
7	Linefeed generated after each Carriage return	Carriage return alone

4055 [ROUTE] - This byte is described as follows:

BIT	IF SET	IF RESET
0	PR routed to SO	PR not routed to SO
1	PR routed to DO	PR not routed to DO
2	DO routed to PR	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	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

4056 thru 4057 [MEMTP] - Previous TOPMEM prior to executing the current "DO" file.

4058 thru 4059 [DOBUF] - Pointer to current "DO" buffer.

405A [NULLS] - Bits 7-4 contain the left margin, bits 3-4 contain the number of NULLS sent after a linefeed.

405B [DBLER] - This byte contains the primary disk I/O error code, if any. This byte is reset to zero when the error is displayed or when the user is in the DOS command mode.

405C [TAW] - This byte is set to a non-zero value when a drive is coming up to speed.

405D thru 407F is used as DEBUG scratchpad when DEBUG is active.

4080 thru 4151 is used as scratchpad and several DOS functions. CMD "uuuuu" saves BASIC's use of 4080 thru 41E5.

41E5 [STKLD] this is the DOS stack address.

4400 thru 4402 [DOS] - Vector to load Command/DOL and enter the DOS command level. The Stack Pointer is loaded with the system stack, 41E5, and "DO" is checked. If "DO" is active, key characters are obtained from the "DO" file. If a "DO" is not active, the contents of KIM, 4047/4265 (MODEL I/III) is examined to see if the terminator was a comma, indicating a multiple dos command. If the contents of KIM, is not a comma, "MULTIDOS" is printed and the user can input a DOS command.

4403 thru 4404 [DATA] - Storage area for extension data.

4405 thru 4407 [EXEC] - Vector to execute the command @ (HL).

4408 [SADOS] - Storage byte for repeat DOS commands.

4409 thru 440C [ERROR] - Entry point to DOS error library. Error message is the lower 5 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.

4419 thru 441B [CAT] - This is a method of obtaining DIRectory data 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.

Temporary exit: (if any of bits 3,4, 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 = ?

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.

Function:
Bit pattern of B register.

BIT	SET	RESET
0	Place files in RAM.	Display to VIDEO.
1	Include I files.	
2	Include S files.	
3	Bits 3 - 5 are the	
4	lower protect area	
5	in lines + 3.	
6	override bits 0-5, and return with free grans 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.

441C thru 441E [FILK] - DE => FCB (20 bytes), HL => filespec. FILK transfers the filespec from (HL) to (DE). Case conversion will be in accordance to the bits in SMO.

441F [VERB] - This byte contains the LSB of the WRITE routine.

File control block (FCB). 20 contiguous bytes of RAM designated by the user. Before open, the FCB has the filespec left justified, terminated with an 0D or 03. After open the FCB is defined as follows:

FCB+00:
  BIT 7 Set to indicate an open file.
  BIT 0 Set forces CLOSE to write a new DIREC. This bit is set whenever a write is made to the file.

FBC+01:
  BIT 7 Set if the file opened has a LRL <> 0, or byte I/O performed.
  BIT 6 Set by POSN (i.e. RANDOM I/O in BASIC). Having this bit set will retain 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 2-0 Access password level.

FCB+02: Only used by BASIC during OPEN"O", or OPEN"E".

FCB+03 and FCB+04: Buffer address for reads or writes. Initially set to HL when INIT or OPEN is executed.

FCB+05: PNR. Position in Next Record (NRN - FCB+0A and FCB+0B)

FCB+06: The drive number.

FCB+07: DIREC position code.

FCB+08: PER. Position in Ending Record. (ERN - FCB+0C and FCB+0D)

FCB+09: LRL. Logical Record Length.

FCB+0A and FCB+0B: NRN. Next record number. (actually one LESS than)

FCB+0C and FCB+0D: ERN. Ending record number. (true)

FCB+0E and FCB+0F: A copy of DIREC+16 and DIREC+17.

The balance of the FCB - FCB+10 thru FCB+1F - contain data in 4 byte clusters.

FCB+10 thru 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 thru FCB+17, FCB+18 thru FCB+1B, and FCB+1C thru FCB+1F: Are the same as FCB+10 thru FCB+13.

Several of the following entry points will return an error if something goes awry. This is indicated with "A = error". No error if "Z" flag set.

4420 thru 4422 [INIT] - DE => FCB (with filespec), HL => 100 byte buffer. INIT calls OPEN, 4424, to see if the file exists. If the file exists a return to the caller is made. 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.

4423 [NOW] - Drive selected scratchpad.

4424 thru 4426 [OPEN] - 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.

4427 [PORT] - A mirror image of the floppy diskette command. This byte is interrogated by MEMDISK, and HARD DISK drivers.

4428 thru 442A [CLOSE] - DE => FCB (opened). CLOSE completes the last write to the filespec, if necessary, 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 thru 442E [KILL] - DE => FCB (opened). KILL deallocates all granules assigned to the filespec. A = error.

442F [CDRN] - Drive number for logical drivespec during file OPENing.

4430 thru 4432 [LOAD] - DE =>FCB (filespec). LOAD places the filespec into RAM. A = error.

4433 thru 4435 [LRUN] - DE => FCB (filespec). LRUN calls LOAD, and pushes the entry point. A return is executed if the file has an attribute of EXEC or NONE else if DEBUG is active, LRUN will execute DEBUG. A = error.

4436 thru 4438 [READ] - 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 thru 443B [WRITE] - 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 thru 443E [VERF] - DE => FCB (opened). VERF calls WRITE then rereads the sector for parity, if a write to disk occurred. A = error.

443F thru 4441 [POBOF] - DE => FCB (opened). POBOF sets the FCB to the status of a just opened file - position zero. A = error.

4442 thru 4444 [POSN] -DE => FCB (opened). POSN sets the FCB to the logical record in the BC register. A = error.

4445 thru 4447 [BCKSP] - DE => FCB (opened). BCKSP sets the FCB one logical record less than the FCB has. A = error.

4448 thru 444A [POEOF] - DE => FCB (opened). POEOF sets the FCB at the end of file. POEOF should return the error 1C.

444B thru 444C [DEXt] - Jump relative to DEXT.

444D thru 444E [FUNC] - FUNC contains the current overlay entry point.

444F thru 4450 [FORCE] - FORCE points to the 4 byte test for floppy I/O error "record not found". If the first two bytes are changed to 18 and 02 respectively, then automatic density recognition is defeated.

4451 thru 4453 [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 the A is zero on entry, HL returns FFFF.

4454 thru 4455 [PARAm] - Jump relative to PARAM.

4456 thru 4458 [REVEC] - Jumps to a modified overlay loader entry point.

4459 thru 445B [ITSIN] - Exit point of modified overlay loader.

4461 thru 4463 [JKL] - Dumps entire VIDEO refresh RAM to the address in the PRDCB+1 and PRDCB+2, converting non-ASCII charactrers to periods.

4464 thru 4466 [CEOF] - 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.

4467 thru 4469 [VOD] - HL =>message. VOD is the general message display routine. VOD outputs a byte to PUTVO, 33, until it encounters a 03, or 0D terminator. Only the 0D terminator is sent to PUTVO. The HL register is positioned one byte after the terminator character.

446A thru 446C [PRT] - HL =>message. PRT is similar VOD, except the output is PUTPR, 3B, instead of PUTVO, 33.

446D thru 446F [GTIME] - 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, DE => [TIC], BC = 003A.

4470 thru 4472 [GDATE] - 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, DE => hour, BC = 002F.

4473 thru 4475 [DEXT] - DE => FCB (filespec). HL => extension. DEXT puts extension in filespec if filespec does not have an extension.

4476 thru 4478 [PARAM] - DE => parameter list. PARAM interrogates the parameter list and sets the addresses accordingly. A parameter list consists of six character phrases, padded to the right with spaces if necessary, followed by a two byte address, and terminated with a zero.

4479 thru 447B [DOTIT] - The FORMS routine will call this address when the software indicates the form is at the top of page. DOTIT points to a return instruction when MULTIDOS is loaded. The user can place titles on printouts by using this sample program.

```
00001 TODOS   EQU     402DH
00002 TOPMM   EQU     4049H   ;MODEL I ADDRESS, USE 4411H FOR MODEL III
00003 DOTIT   EQU     4479H
00004 PRT     EQU     446AH
00005         ORG     0FC00H          ;ARBITRARY ADDRESS
00006 START   LD      HL,MYCDE
00007         LD      (DOTIT+1),HL
00008         DEC     HL
00009         LD      (TOPMM),HL      ;PROTECTS ROUTINE
00010         JP      TODOS
00011 MYCDE   EXX                     ;MULTIDOS DOESN'T USE ALT REGS.
00012         LD      HL,MYTIT
00013         CALL    PRT             ;CALL ITSELF (SAFE BECAUSE OF EXX)
00014         OR      A               ;MUST RETURN A NON ZERO
00015         EXX                     ;PUT 'EM BACK
00016         RET                     ;BACK TO CALLER
00017 MYTIT   DEFM    'THIS IS MY WORK'
00018         DEFB    0DH
00019         END     START
```

447C thru 4483 [ADRQ] - If floppy, ADRQ loads DCT+0 with the contents of the A register, then sets DCT+6, DCT+7, DCT+8, and DCT+9, in accordance with DCT+0.

4497 thru 44A0 [LWAIT] - This routine will load the floppy disk controller with the contents of the A register, then wait approx. 30 micro seconds.

44A1 thru 44AA [FPOS] - IX => FCB. FPOS returns the ERN in the HL register, and the PER in the A register.

44AB thru 44C8 [SEEK] - D = track, E = sector, C = logical drive. SEEK will position the floppy disk R/W head over the desired track.
44C9 thru 44CB [MOTON] - Floppy disk drive select, and motor turn on.

44CC thru 44CE [GRDUM] - Dumps entire VIDEO refresh RAM to the address in the PRDCB+1 and PRDCB+2, including graphic characters.

44CF thru 44D1 [ZZZZ] - A call to ZZZZ prints "Insert SYSTEM <ENTER>", and waits for the <ENTER> key pressed, then returns.

44D8 thru 44D9 [LNKTO] - Contains address for LINK control.

44DA [GTDCC] - Position IY register to the DCT in the C register.

44DB thru 44DD [GTDCT] - Position IY register to the DCT in the A register.

44DE thru 44E0 [READV] - D = track, E = sector, C = logical drive. READV checks the sector for readability. A = error.

44E1 thru 44E3 [READS] - D = track, E = sector, C = logical drive, HL = 100 byte buffer. READS transfers the sector data into (HL). A = error.

44E4 thru 44E7 [WRITS] - D - track, E = sector, C = logical drive, HL = 100 byte buffer. WRITS transfers the data in (HL) to the sector. A = error.

44E8 thru 44EA [DIRRD] - DIRRD issues a READS specifically looking for a DELETED DATA MARK (DDM). DIRRD returns if a DDM sector is read, otherwise DIRRD will read physical track zero, physical sector zero, relative byte two, and update DCT+3 with this byte (directory cylinder), then reads the new TRACK, same sector. If the new track, same sector does not return a DDM, DIRRD will try "P" density1, if "P" density doesn't return a DDM, DIRRD will try "P" density2. If still a DDM isn't found, DDIRD will return an error of 11. A = error.

44EB thru 44ED [DIRWT] - Same as WRITS, except issues DDM. A = error.

44EE thru 44F0 [RDIRP] - On entry, the B register contains the DIREC position code, and the C register contains the 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. i.e. CLOSE. This is why you should NEVER change diskettes after a file is opened, and will remain 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 will write 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 will return with the HL register pointing to the register B position code DIREC, in the I/O buffer. A = error.

44F1 thru 44F3 [WDIRP] - On entry, the B register contains the DIREC position code, the C register contains the logical drive number, and the information is in BUFF. WDIRP will exit with HL => BUFF. A = error.

44F4 thru 44F6 [USRF] - D = track, E = sector, C = logical drive, HL = buffer, and the A register contains the FCB code. USRF accepts any of the read sector, write sector, or write track commands. A = error.

44F7 thru 44F9 [GETDT] - On entry, C = logical drive. On exit the D register will contain the contents of DCT+3 for the C register DCT.

44FA thru 44FC [MODE] - Called by CLOSE to see if any data has not been written to the file.

44FD thru 44FF [OPESA] - Checks for DE => FCB opened. If open saves BC, DE, HL, IX, and IY registers, IX returned => FCB.

4500 thru 457F [DCTS] - Eight Drive Control Tables.

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 NEWDOS/80.
BIT	4	If set - Perform a 1.8 to 1 division to read NEWDOS/80 MOD III.
BIT	5	If set - Double sided diskettes are treated as two volume.
BIT	6	If set - 8" floppy, otherwise 5 1/4" floppy.
BIT	7	If set - Double density media, otherwise single density.

HARD disk bit pattern:			MEMDISK bit pattern:
BIT	0	Step rate LSB.	N/A
BIT	1	Step rate.	N/A
BIT	2	Step rate.	N/A
BIT	3	Step rate MSB.	N/A
BIT	4	SET TO ONE.	SET TO ONE
BIT	5	SET TO ZERO.	SET TO ZERO
BIT	6	SET TO ONE.	SET TO ONE
BIT	7	SET TO ZERO.	SET TO ONE.

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/MEMDISK. A zero indicates floppy.
BIT	5	Used for HARD drives to indicate format pattern is logged.
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. 1 to 127. Floppies will also have the high bit set to indicate a two head one volume format.

DCT+03: The directory cylinder. This byte is obtained from cylinder zero, sector zero, relative byte two.

DCT+04: 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: Dynamic head number selected.

DCT+06: The number of sectors per granule.

DCT+07: The number of granules per cylinder.

DCT+08: The number of sectors per track.

DCT+09: The number of directory file sectors on the first surface.

DCT+0A: The most significant byte for the number of sectors per cylinder.

DCT+0B: The least significant byte for the number of sectors per cylinder.

DCT+0C: The most significant byte for the cylinder offset.

DCT+0D: The least significant byte for the cylinder offset.

DCT+0E: The head offset.

DCT+0F: Not assigned.

DCT+0A thru DCT+0F are not used by floppies.

4580 thru 4CFF contain the code for the balance of DOS functions.

4D00 thru 51FF is the BASIC overlay area. Command/DOL, Debug/DOL, and Minidos/DOL also use this area.

4E00 thru 51FF is the DOS overlay area.

BACKUP/CMD, FORMAT/CMD, RS/CMD, VFU/CMD, ZAP/CMD, and the library command DIR, use 4D00 thru 51FF. If Allocate/DOL is called while using RS/CMD, ZAP/CMD, or the library command DIR, then a memory conflict will occur. This can only happen if a device is accessed while the device is routed to a file requiring more disk space or positioning beyond the fourth segment.

UNIQUE MODEL I

4040 [TIC] - This byte is incremented by one for each real time clock interrupt.

4041 - seconds.
4042 - minutes.
4043 - hour.
4044 - year.
4045 - day.
4046 - month.

4047 thru 4048 [KIM] - Points to the terminator byte after executing a DOS function.

4049 thru 404A [TOPMM] - Contain the address of the highest available memory location available for DOS, BASIC temporarily sets this word to a different value during CMD "uuuuu" execution.

4200 thru 42FF [BUFF] - System I/O buffer.

430E [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 while loading an overlay, and resets this byte to zero after completion of the overlay function.

430F [OLC] - Overlay mask storage for the overlay in the overlay area when Minidos is invoked. Minidos will reload this overlay upon exit.

4310 [OLB] - Overlay mask of the overlay in the overlay area when a new overlay is executed.

4311 [OLA] - Current overlay mask. The overlay mask is bits 5,3,2,1, and 0.

430C [SM0] - The following c-mments apply if the noted bit is set.

BIT 0. During file OPENing this bit will be set if the file has an attribute of EXEC or NONE. This bit is valid AFTER a file has been OPENed.
BIT 1. Set if LRUN, 4433, is entered.
BIT 2. Ignore PASSWORDS. Reset when file is OPENed.
BIT 3. Don't RAM date. Reset after file CLOSEd.
BIT 4. Set if EXEC, 4405, is entered.
BIT 5. Accept !"#$%&<>?@ in a filespec.
BIT 6. Accept numbers as the first character in a filespec or extension.
BIT 7. Do not convert lowercase to uppercase. (accepts > 3D in filespec.)

430D [SM1] - 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.
BIT 3. Library2/EXT RESTOR command executed.
BIT 4. Library1/EXT DIR command executed.
BIT 5. In BASIC. This bit is reset during CMD"uuuuu".
BIT 6. Type ahead active.
BIT 7. "DO" active.

4315 thru 4317 [EXDB] - Debug vector. If 4315 contains zero, DEBUG will not load when <SHIFT><BREAK> is pressed, regardless of bit 0 in SM1. The DOS command interpreter interrogates bit 0 in SM1 and sets 4315 accordingly. (If bit 0 is non-zero, 4315 is loaded with C3.)

4318 thru 4357 [COMBF] - DOS command mode input buffer.

4359 [SUBY] - Loads HL with (HL)+1.

435A thru 435E [SUBZ] - Loads HL with (HL).

440D thru 440F [DEBUG] - Debugger load/execute entry.

4410 thru 4412 [IUS] - Insert a task in 1 of 4 slots. (25ms each). The A register contains the slot number and the DE register contains the address of the routine. When a interrupt task is entered, all registers have been saved and the interrupts are disabled and MUST remain disabled. The HL register will be at the start of the routine upon entry. The current version of MULTIDOS uses SLOT 0 for the clock, and SLOT 3 for the spooler.

4413 thru 4415 [DUS] - Delete task in slot A. Uses AF, DE, and HL.

UNIQUE MODEL III

4215 [MMB] - Same as MMB, 430E, MODEL I.

4216 [TIC] - This byte circulates from 1E to 01 for each real time clock interrupt.

4217 - seconds.
4218 - minutes.
4219 - hour.
421A - year.
421B - day.
421C - month.

4225 thru 4264 [COMBF] - DOS command mode input buffer.

4265 thru 4266 [KIM] - Points to the terminator byte after executing a DOS function.

4267 thru 4269 [EXDB] - Same as EXDB, 4315, MODEL I.

426A [OLC] - Same as OLC, 430F, MODEL I.

426B [OLB] - Same as OLB, 4310, MODEL I.

426C [OLA] - SAME AS OLA, 4311, MODEL I.

426D [SMO] - SAME AS SMO, 430C, MODEL I.

426E [SM1] - 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.
BIT 3. Library2/EXT RESTOR or SETCOM command executed.
BIT 4. Library1/EXT DIR command executed.
BIT 5. In BASIC. This bit is reset during CMD"uuuuu".
BIT 6. Type ahead active.
BIT 7. "DO" active.

4300 thru 43FF [BUFF] - System I/O buffer.

4411 thru 4412 [TOPMM] - SAME AS TOPMEM, 4049 thru 404A, MODEL I.

4491 thru 4496 [RUPP] - The first byte is a F3 when TYPE-AHEAD is inactive, or a 0 when TYPE-AHEAD is active to keep the interrupts on during disk I/O.

44D2 thru 44D4 [IUS] - Insert a task in 1 of 4 slots. (33 1/3ms each). The A register contains the slot number and the DE register contains the address of the routine. When a interrupt task is entered, all registers have been saved and the interrupts are disabled and MUST remain disabled. The HL register will be at the start of the routine upon entry. The current version of MULTIDOS uses SLOT 0 for TYPE-AHEAD, and SLOT 3 for the spooler.

44D5 thru 44D7 [DUS] - Delete task in slot A. Uses AF, DE, and HL.

OPTIONAL ZAPS for MULTIDOS

(1) High speed modifications. (MULTIDOS can perform disk I/O in high speed.)

MODEL I

Most software high speed modifications for the MODEL I require an output to a 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 F0 to the PORT in relative position F1 located on track 0 sector 1.

```
STD F0  0CFE 0000 0504
```

The above partial ZAP screen of track one, sector zero, indicates an 0C will be output to port FE during initialization.

MODEL III

The MODEL III version of MULTIDOS will set bit 6 of 4210 and output a 1 to port 5F during initialization. The code to accomplish this is on track zero sector one from relative byte 0B thru relative byte 18.

```
HEX 00  F3ED 563A 0243 FE56 C200 0021 1042 CBF6
    10  CB6E 7E3D EC3E 01D3 5F
```

This expressed in assembly language is:

MNEMONICS		relative bytes
LD	HL,4210H	0B,0C,0D
SET	6,(HL)	0E,0F
SET	4,(HL)	10,11
LD	A,(HL)	12
OUT	(0ECH),A	13,14
LD	A,1	15,16
OUT	(5FH),A	17,18

Bytes 0E and 0F set a MODEL 4 to 4MHz. Zero these two bytes if you do not want the MODEL 4, while running in the MODEL III mode, to run at 4MHz. Bytes 15, 16, 17, and 18 set the HOLMES board to high speed. If you want to sent a byte to another port, modify these bytes as desired.

(2) Disk I/O tries. (distributed value = 05)

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 immediately. Increasing the number of I/O tries may permit you to read the file; however, eventually it will fail. If you want to change the number of I/O tries, modify the byte in relative position F4 on track zero sector one (MODEL I), or sector zero (MODEL III). DO NOT USE ONE!

(3) Head load delay. (distributed value = 04)

Whenever MULTIDOS selects a different drive, even if the motors are up to speed, a delay of 15ms occurs before an attempt is made to read/write to the newly selected drive. The amount of delay is in 3.75ms (MODEL I), or 3.79ms (MODEL III) for each unit in relative byte F5 on track zero sector one (MODEL I), or sector zero (MODEL III). DO NOT USE ZERO!

(4) OPEN error codes. (distributed values = 18, 18)

The error code most application programs expect when an attempt is made to access a file, and the "file is not found", is 18. 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 18 is correct for case 1, error code 08 is correct for case 2, and error code 1F is correct for case 3. So what! Nevertheless, for the purist, the correct error code can be returned. This is accomplished by zapping the last 2 bytes in Open/DOL. Now wait a minute. OK, the last 2 bytes to be loaded. The last four bytes in Open/DOL are 02, 02, 17, 4E. These four bytes are used by the system loader and are not part of the executed code. Therefore, the bytes to zap immediately precede these four bytes. Since the MODEL I and MODEL III versions of Open/DOL are different, you will have to find them yourself! Being somewhat a purist, the MULTIDOS files were created with an Assembler that zeroes all unused bytes in the last sector. Ah ha! They won't be that difficult to find after all. The first of the last two (six) bytes is the error code returned if a drive is not available. The second of the last two (six) bytes is the error returned if the file was searched for and not found. Purist zap them with 08, and 1F respectively.

(5) Default value for SMO. (distributed value = 0)

If any of the high three bits in SMO are set, the user can have unusual filenames. However, keep in mind not all operating systems recognize the new syntax. Relative byte F7 on track zero sector one (MODEL I), or sector zero (MODEL III) is loaded into SMO during initialization.

(6) Default response for ZAP/CMD menu. (distributed value = 43)

ZAP/CMD's first sector in relative byte 80, contain the menu default character. (This is the only user alterable default.) ZAP it, if you want!

(7) Default number of file buffer areas for BASIC (distributed value = 3)

BASIC/CMD's first sector in relative byte 84, contain the default number of file buffer areas allocated during BASIC initialization. Please use a number in the range of 00 and 0F. A number greater than 0F will clobber part of BASIC/CMD code.

MODEL 4 MULTIDOS

I. GENERAL (All numbers are hexadecimal)

<CAPS> is used to toggle between CAPS and upper/lower case. <SHIFT>0 does NOT toggle between CAPS and upper/lower case.

<LEFT><SHIFT><F1> cancels <LEFT><SHIFT><F2>, and <LEFT><SHIFT><F3>

<LEFT><SHIFT><F2> activates the <CLEAR> key as an ALT key. When the <CLEAR> key is held down and an alpha key, A thru Z, is pressed, the value returned will be the alpha key code plus 40. When the <CLEAR> key is held down and a shifted (either <SHIFT> key) alpha key is pressed, the value returned will be the alpha key code plus 60, i.e. <ALT><C> = 83, <ALT><SHIFT><D> = A4. The <LEFT><SHIFT><F2> function disregards the condition of the CAPS mode.

<LEFT><SHIFT><F3> activates the <CLEAR> key as an ALT CTRL key. When the <CLEAR> key is held down and another key is pressed (including <ENTER>, <BREAK>, and the arrow keys) the value returned will be the key pressed plus 80. The <LEFT><SHIFT><F3> function recognizes the condition of the CAPS mode, i.e. <ALTCTRL><C> = C3, <ALTCTRL><SHIFT><C> = C3 - in CAPS. <ALTCTRL><C> = E3, <ALTCTRL><SHIFT><C> = C3 - in upper/lower case.

<SHIFT><RIGHT-ARROW> only converts to the 32/40 character mode if these keys are pressed when the input mode is at the first position.

II. MEMORY MAP of Model 4 MULTIDOS (All numbers are hexadecimal)

0 thru 7FF inclusive is used for device I/O handling, KEYBOARD driver, INTERRUPT handling, VIDEO driver, PRINTER driver, and TIMEKEEPING.

Location	Function

0 (RST 0) [SUPV] CLS, VIDEO WIDTH, and FUNCTION KEYS

INPUT	USES	FUNCTION
A=0	AF	Clears display
A=1	AF	Set VIDEO to 64x16 and clears display.
A=2	AF	Set VIDEO to 80x16 and clears display.
A=3,B=0	AF	Set F1 thru F6 to null keys.
A=3,B=1	AF	Set F1 to routine @ DE.
A=3,B=2	AF	Set F2 to routine @ DE.
A=3,B=3	AF	Set F3 to routine @ DE.
A=3,B=4	AF	Set F4 to routine @ DE. <RT><SHIFT><F1>
A=3,B=5	AF	Set F5 to routine @ DE. <RT><SHIFT><F2>
A=3,B=6	AF	Set F6 to routine @ DE. <RT><SHIFT><F3>
A=3,B>6	AF	IGNORED.
A>3	AF	IGNORED.

The KEYBOARD scan routine will execute the function, then return with the character in the A register. If a particular key input is also desired, then LD A with the character just before the last RET instruction. Function key commands/characters do NOT repeat.

03 Contains 17 to indicate Version 1.7

04 Contains the VIDEO width under software control.

05 thru 07 Vector to special first position keystroke when 40 is called. If the desired key is pressed, and a special function is desired, then remove (at least) the two calls from the STACK:

```
POP   HL    ;call to 0005
POP   HL    ;restore register
POP   QQ    ;call to 0040
```

08 thru 0F Not used in DOS. Same function in BASIC.
10 thru 12 Vectors to PARSE under DOS, otherwise same BASIC function.
13 thru 17 [GETBT] Same as MOD I/III.
18 thru 1A Same as MOD I/III.
1B thru 1F [PUTBT] Same MOD I/III.
20 thru 22 Not used in DOS. Same function in BASIC.
23 thru 27 Same as MOD I/III.
28 thru 2A Overlay load vector. <BREAK> key does NOT vector here!
2B thru 2F [GETKI] Same MOD I/III.
30 thru 32 Vector to load/execute DEBUG. Same as MOD I/III.
33 thru 37 [PUTVO] Same as MOD I/III.
38 thru 3A Interrup mode I. Same as MOD I/III.
3B thru 3F [PUTPR] Same as MOD I/III.
40 thru 42 [LINE] Same as MOD I/III.
43 [SUBY] 43 INC HL ;This uses wasted
[SUBZ] 44 LD A,(HL) ; space in MOD I/III
45 INC HL
46 LD H,(HL)
47 LD L,A
48 RET
49 thru 4F [KWAIT] Same as MOD I/III.
50 thru 54 [GETSI] - Functions similar to MODEL III.
55 thru 59 [PUTSO] - Functions similar to MODEL III.
5A [IMAGE] - Contains contents of PORT 84.
5B 00 This byte must remain zero.
5C thru 5D [WHERE] - Same as 1B MOD I/III.
5E & 5F Extension of DELAY routine at 60.
LD BC F780 to obtain a one second delay.
60 thru 65 [DELAY] Same function as MOD I/III.
66 thru 68 NMI vector.
69 thru 7FF Totally different code here (SUBJECT TO CHANGE!)

800 thru 2FFF This area is loaded/used by BASIC and is expected to remain intact during a CMD"uuuuu". If BASIC is not being used, then this area if free system RAM.

3000 thru 37FF is used by DOS library command DIR, by CAT, BACKUP, FORMAT, VFU, ZAP, and BASIC. This area is handled as a psuedo-overlay area. This area can be used, provided bit 4 of SM1 (4222) is set. DOS and BASIC will check the status of this bit and reload this area, if necessary.

3800 thru 3BFF is psuedo RAM used for dectecting key presses.

3C00 thru 3FFF is used for VIDEO refresh RAM.

VIDEO codes.

0-6	Ignored.
7	Beep.
8	Backspace cursor one position and erase character under cursor.
9	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.
11	Reset reverse video high bit routine.
12	Set all VIDEO refresh RAM from cursor position to end of frame to the contents in A@ (404C).
13	Position cursor to HOME position.
14	Scroll VIDEO and cursor up one line.
15	Swap space compression code with special characters.
16	Swap alternate special characters with special characters.
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	HOME cursor, disable reverse video, and reset 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.

```
LD      A,code
CALL    PUTVO
```

On exit A = input and Z flag is set.

4000 thru 4009 [KEYCL] - These bytes are used by the keyboard driver for nKEY rollover and scrath pad for KEY repeat rate.

400A [BLDR] - Line input <CLEAR> key override if non-zero. Places character into buffer.

400B [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	SPOLLER buffer full	SPOOLER buffer not full
7	SPOOLER buffer has data	SPOOLER buffer empty

400C [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
6	PR output to SO	PR output to parallel port
7	Linefeed generated after each Carriage return	Carriage return alone

400D [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	DO not routed to PR
3	DO routed to SO	DO not routed to SO
4	SO routed to PR or file	SO not routed to PR
5	SO routed to DO	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

400E thru 400F [MEMTP] - Previous TOPMEM prior to executing the current "DO" file.

4010 thru 4011 [DOBUF] - Pointer to current "DO" buffer.

4012 [NULLS] - Bits 7-3 contain the left margin, bits 2-0 contain the number of NULLS sent after a linefeed.

4013 [DBLER] - This byte contains the primary disk I/O error code (second error message), if any. This byte is reset to zero when the error is displayed or when the user is in the DOS command mode.

4014 [TAW] - This byte goes non-zero when a drive is comming up to speed.

4015 thru 401C [KBDCB] - KEYBOARD 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 RESERVED.
1 RESERVED.
2 Space compression.
3 lowercase.
4 key beep.
5 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, protect 5 lines. (protect 1 to 8 lines.)

401D thru 4024 [VODCB] - VIDEO 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.
DCB+6 VIDEO column position.
DCB+7 VIDEO row position.

4025 thru 402C [PRDCB] - PRINTER 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 thru 402F [TODOS] - Vector to indicate the completion of a DOS command, error or no error.

4030 thru 4032 [DESS] - Vectors to 402D after resetting Stack Pointer to the DOS stack, 42FE, and prompting the user "Insert SYSTEM <ENTER>". If in BASIC, vectors to function similar to 6CC on MOD I's.

4033 [GRAF] - Sets the keyboard to <LEFT><SHIFT><F2> mode.

4034 thru 4035 [ULIN] - Sets the keyboard to <LEFT><SHIFT><F3> mode.

4036 thru 4039 [UFUN] - Sets the keyboard to <LEFT><SHIFT><F1> mode.

403A thru 403C [SIDCB] - DCB for the RS-232-C.

403D thru 403F [SODCB] - DCB for the RS-232-C.

4040 [TIC] This bytes increases by one for each real time interrupt.

4041 thru 4042 [BAUD] - RS-232-C default storage.

4043 thru 404B [TACKL] - TYPE-AHEAD key mask storage.

404C [A@] - Stores character for write to end of VIDEO when a 12 is sent to VIDEO routine.

404D [B@] - BREAK KEY status. If zero, the KEYBOARD will return a zero when <BREAK> is pressed. DOS command modes sets this byte to a non zero value.

404E thru 405C [PARSE] - Increments the HL register pair, interrogates the byte in HL, skips spaces and returns with the "C" flag set if numeric, "Z" flag set if ":" or 0.

405D thru 407F [DBST] is used as DEBUG scratchpad when DEBUG is active.

4080 thru 4151 is used as BASIC scratchpad and several DOS functions.

411C thru 4151 is reusable scratchpad. CMD "uuuuu" saves 4080 thru 411B.

4152 thru 4157 is used as a mirror copy of the time and date. This area is updated every second. Used to recover the time and date after re-boot.

4158 thru 41FF is used by the system to load overlays, has the LRUN routine, and time keeping tables.

4200 thru 4202 [EXDB] - Debug vector. If 4200 is zero, DEBUG will not load when <LEFT><SHIFT><BREAK> is pressed, regardless of bit 0 in SM1. The DOS command interpreter interrogates bit 0 in SM1 and sets 4200 accordingly. (If bit 0 is non-zero, 4200 is loaded with C3.)

4203 thru 4205 [HOMO] - VIDEO control.

4206 thru 4208 [XYRO] - VIDEO control.

4209 - RESERVED.

420A thru 420F [RUPP] - This byte is a F3 when TYPE-AHEAD is inactive. This byte is 0 when TYPE-AHEAD is active to keep the interrupts on during diskette I/O.

4210 [MODUT] - This is maintained in the same function as the MODEL III.

4211 thru 4213 [PAGX] - This vectors to the routine which corrects the HL register (HL register contains the relative cursor position offset by 3C00 - 3C00 to 437F) and updates the page bit for port 84.

4214 - Not used.

4215 thru 4216 [CLON] - Pointer to clock routine.

4217 - seconds.
4218 - minutes.
4219 - hour.
421A - year.
421B - day.
421C - month.

421D [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 while loading an overlay, and resets this byte to zero after completion of the overlay function.

421E [OLC] - Storage of overlay mask of the overlay in the overlay area when Minidos is invoked. Minidos will reload this overlay upon exit.

421F [OLB] - The overlay mask of the overlay in the overlay area when a new overlay is executed.

4220 [OLA] - Current overlay mask.
The overlay mask is bits 5,3,2,1, and 0.

4221 [SM0] - The following comments apply if the noted bit is set.

BIT 0. During file OPENing this bit will be set if the file has an attribute of EXEC or NONE. This bit is valid AFTER a file has been OPENed.
BIT 1. Set if LRUN, 4433, is entered.
BIT 2. Ignore PASSWORDS. Reset when file is OPENed.
BIT 3. Don't RAM date. Reset after file CLOSEd.
BIT 4. Set if EXEC, 4405, is entered.
BIT 5. Accept !"#$%&<>?@ in a filespec.
BIT 6. Accept numbers as the first character in a filespec or extention.
BIT 7. Do not convert lowercase to uppercase. (accepts > 3D in filespec.)

4222 [SM1] - 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.
BIT 3. Help/EXT is in RAM area 5200 to 68FF.
BIT 4. 3000 thru 37FF has been used.
BIT 5. In BASIC. This bit is reset during CMD"uuuuu".
BIT 6. Type ahead active.
BIT 7. "DO" active.

4223 thru 4224 [KIM] - Points to the terminator byte after executing a DOS function.

4225 thru 4274 - DOS command mode input buffer.

4275 thru 42FD is the system stack area.

42FE thru 42FF contain 4030. (Safety valve)

4300 thru 43FF is the disk I/O BUFFER.

4400 thru 4402 [DOS] - Vector to load Command/DOL and enter the DOS command level. The Stack Pointer is loaded with the system stack, 42FE, and "DO" is checked. If "DO" is active, key characters are obtained from the "DO" file. If a "DO" is not active, the contents of KIM, 4223 is examined to see if the terminator was a comma, indicating a multiple dos command. If the contents of KIM, 4223 is not a comma, "MULTIDOS" is printed and the user can input a DOS command.

4403 thru 4404 [DATA] - Storage area for extention data.

4405 thru 4407 [EXEC] - Vector to execute the command @ (HL).

4408 [SADOS] - Storage byte for repeat DOS commands.

4409 thru 440C [ERROR] - Entry point to DOS error library. Error message is the lower 5 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 will return to the caller. If bit 7 is not set, this routine will exit to 402D.

440D thru 4410 [DEBUG] - Debugger load/execute code.

4411 thru 4412 [TOPMM] - Contain the adrress of the highest available memory location available for DOS. BASIC temporarily sets this word to a different value during CMD "uuuuu" execution.

4413 thru 4415 [MXLIN] - VIDEO maximum line condition. Returns in 40X10 A=40, D=10; in 50X18 A=50, D=18.

4416 thru 4418 [FIXRR] - This routine will return the corresponding refresh RAM location in the HL register pair when entered with B = row, and C = column.

4419 thru 441B [CAT] - Obtaining DIRectory data on a mounted disk.
Entry:
A = not used.
B = function. (see below)
C = ;ogical drive spec.
DE = not used.
HL = RAM area, if function directs to RAM.

Temporary exit: (if any of bits 3,4, 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 OFF 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.

Function:
Bit pattern of B register.

BIT	SET	RESET
0	Place files in RAM.	Display to VIDEO.
1	Include I files.	
2	Include S files.	
3	Bits 3 - 5 are the	
4	lower protect area	
5	in lines + 3.	
6	override bits 0-5, and return with free grans 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.

441C thru 441E [FILK] - DE => FCB (20 bytes), HL => filespec. FILK transfers the filespec from (HL) to (DE). Case conversion will be in accordance to the bits in SM0.

441F [VERB] - This byte contains the LSB of the WRITE routine.

File control block (FCB). 20 contiguous bytes of RAM designated by the user. Before open, the FCB has the filespec left justified, terminated with an 0D or 03. After open the FCB is defined as follows:

FCB+00:
  BIT 7 Set to indicate an open file.
  BIT 0 Set forces CLOSE to write a new DIREC. This bit is set whenever a write is made to the file.

FBC+01:
  BIT 7 Set if the file opened has a LRL <> 0, or byte I/O performed.
  BIT 6 Set by POSN (i.e. RANDOM I/O in BASIC). Having this bit set will retain 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 2-0 Access password level.

FCB+02: Only used by BASIC during OPEN"O", or OPEN"E".

FCB+03 and FCB+04: Buffer address for reads or writes. Initially set to HL when INIT or OPEN is executed.

FCB+05: PNR. Position in Next Record (NRN - FCB+0A and FCB+0B)

FCB+06: The drive number.

FCB+07: DIREC position code.

FCB+08: PER. Position in Ending Record. (ERN - FCB+0C and FCB+0D)

FCB+09: LRL. Logical Record Length.

FCB+0A and FCB+0B: NRN. Next record number. (actually one LESS than)

FCB+0C and FCB+0D: ERN. Ending record number. (true)

FCB+0E and FCB+0F: A copy of DIREC+16 and DIREC+17.

The balance of the FCB - FCB+10 thru FCB+1F - contain data in 4 byte clusters.

FCB+10 thru 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 thru FCB+17, FCB+18 thru FCB+1B, and FCB+1C thru FCB+1F: Are the same as FCB+10 thru FCB+13.

Several of the following entry points will return an error if something goes awry. This is indicated with "A = error". No error if "Z" flag set.

4420 thru 4422 [INIT] - DE => FCB (with filespec), HL => 100 byte buffer.

INIT calls OPEN, 4424, to see if the file exists. If the file exists a return to the caller is made. 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.

4423 [NOW] - Drive selected scratchpad.

4424 thru 4426 [OPEN] - 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.

4427 [PORT] - A mirror image of the floppy diskette command. This byte is interrogated by MEMDISK, and HARD DISK drivers.

4428 thru 442A [CLOSE] - DE => FCB (opened). CLOSE completes the last write to the filespec, if necessary, 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 thru 442E [KILL] - DE => FCB (opened). KILL deallocates all granules assigned to the filespec. A = error.

442F [CDRN] - Drive number for logical drivespec during file OPENing.

4430 thru 4432 [LOAD] - DE =>FCB (filespec). LOAD places the filespec into RAM. A = error.

4433 thru 4435 [LRUN] - DE => FCB (filespec). LRUN calls LOAD, and pushes the entry point. A return is executed if the file has an attribute of EXEC or NONE else if DEBUG is active, LRUN will execute DEBUG. A = error.

4436 thru 4438 [READ] - 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 thru 443B [WRITE] - 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 thru 443E [VERF] - DE => FCB (opened). VERF calls WRITE then rereads the sector for parity, if a write to disk occurred. A = error.

443F thru 4441 [POBOF] - DE => FCB (opened). POBOF sets the FCB to the status of a just opened file - position zero. A = error.

4442 thru 4444 [POSN] -DE => FCB (opened). POSN sets the FCB to the logical record in the BC register. A = error.

4445 thru 4447 [BCKSP] - DE => FCB (opened). BCKSP sets the FCB one logical record less than the FCB has. A = error.

4448 thru 444A [POEOF] - DE => FCB (opened). POEOF sets the FCB at the end of file. POEOF should return the error 1C.

444B thru 444C [DEXt] - Jump relative to DEXT.

444D thru 444E [FUNC] - FUNC contains the current overlay entry point.

444F thru 4450 [FORCE] - FORCE points to the 4 byte test for floppy I/O error "record not found". If the first two bytes are changed to 18 and 02 respectively, then automatic density recognition is defeated.

4451 thru 4453 [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 the A is zero on entry, HL returns FFFF.

4454 thru 4455 [PARAm] - Jump relative to PARAM.

4456 thru 4458 [REVEC] - Jumps to a modified overlay loader entry point.

4459 thru 445B [ITSIN] - Exit point of modified overlay loader.

445C thru 445D [PUNCH] - Points to address of the 10 byte table which has the code returned for the "ENTER", "CLEAR", "BREAK", arrows, and "SPACE-BAR".

445E thru 4460 [DSTAT] - Returns the status of the logical drive in the C register. The "Z" flag will be reset if no disk, the "Z" flag will be set if disk mounted, and the "C" flag will be set if write protected.

4461 thru 4463 [JKL] - Dumps entire VIDEO refresh RAM to the address in the PRDCB+1 and PRDCB+2, converting non-ASCII charactrers to periods.

4464 thru 4466 [CEOF] - 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.

4467 thru 4469 [VOD] - HL =>message. VOD is the general message display routine. VOD outputs a byte to PUTVO, 33, until it encounters a 03, or 0D terminator. Only the 0D terminator is sent to PUTVO. The HL register is positioned one byte after the terminator character.

446A thru 446C [PRT] - HL =>message. PRT is similar VOD, except the output is PUTPR, 3B, instead of PUTVO, 33.

446D thru 446F [GTIME] - 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, DE => [TIC], BC = 003A.

4470 thru 4472 [GDATE] - 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, DE => hour, BC = 002F.

4473 thru 4475 [DEXT] - DE => FCB (filespec). HL => extention. DEXT puts extention in filespec if filespec does not have an extention.

4476 thru 4478 [PARAM] - DE => parameter list. PARAM interrogates the parameter list and sets the addresses accordingly. A parameter list consists of six character phrases, padded to the right with spaces if necessary, followed by a two byte address, and terminated with a zero.

4479 thru 447B [DOTIT] - The FORMS routine will call this address when the software indicates the form is at the top of page. DOTIT points to a return instruction when MULTIDOS is loaded. The user can place titles on printouts by using this sample program.

```
00001 TODOS   EQU     402DH
00002 TOPMM   EQU     4049H   ;MODEL I ADDRESS, USE 4411H FOR MODEL III
00003 DOTIT   EQU     4479H
00004 PRT     EQU     446AH
00005         ORG     0FC00H          ;ARBITRARY ADDRESS
00006 START   LD      HL,MYCDE
00007         LD      (DOTIT+1),HL
00008         DEC     HL
00009         LD      (TOPMM),HL      ;PROTECTS ROUTINE
00010         JP      TODOS
00011 MYCDE   EXX                     ;MULTIDOS DOESN'T USE ALT REGS.
00012         LD      HL,MYTIT
00013         CALL    PRT             ;CALL ITSELF (SAFE BECAUSE OF EXX)
00014         OR      A               ;MUST RETURN A NON ZERO
00015         EXX                     ;PUT 'EM BACK
00016         RET                     ;BACK TO CALLER
00017 MYTIT   DEFM    'THIS IS MY WORK'
00018         DEFB    0DH
00019         END     START
```

447C thru 4483 [ADRQ] - If floppy, ADRQ loads DCT+0 with the contents of the A register, then sets DCT+6, DCT+7, DCT+8, and DCT+9, in accordance with DCT+0.

4497 thru 44A0 [LWAIT] - This routine will load the floppy disk controller with the contents of the A register, then wait approx. 30 micro seconds.

44A1 thru 44AA [FPOS] - IX => FCB. FPOS returns the ERN in the HL register, and the PER in the A register.

44AB thru 44C8 [SEEK] - D = track, E = sector, C = logical drive. SEEK will position the floppy disk R/W head over the desired track.

44C9 thru 44CB [MOTON] - Floppy disk drive select, and motor turn on.

44CC thru 44CE [GRDUM] - Dumps entire VIDEO refresh RAM to the address in the PRDCB+1 and PRDCB+2, including graphic characters.

44CF thru 44D1 [ZZZZ] - A call to ZZZZ prints "Insert SYSTEM <ENTER>", and waits for the <ENTER> key pressed, then returns.

44D2 thru 44D4 [IUS] - Insert a task in 1 of 8 slots. (numbered 0 thru 7). The "A" register contains the slot number and the DE register contains the address of the routine. When a interrupt task is entered, the AF, BC, DE, and HL register pairs have been saved and the interrupts are disabled and

MUST remain disabled. The "HL" register will be at the start of the routine upon entry. The current version of MULTIDOS uses SLOT 0 for the clock, SLOT 1 for TYPE-AHEAD, SLOT 2 for utilities, and SLOT 3 for the spooler.

44D5 thru 44D7 [DUS] - Delete task in slot A. Uses AF, DE, and HL.

44D8 thru 44D9 [LNKTO] - Contains address for LINK control.

44DA [GTDCC] - Position IY register to the DCT in the C register.

44DB thru 44DD [GTDCT] - Position IY register to the DCT in the A register.

44DE thru 44E0 [READV] - D = track, E = sector, C = logical drive. READV checks the sector for readibility. A = error.

44E1 thru 44E3 [READS] - D = track, E = sector, C = logical drive, HL = 100 byte buffer. READS transfers the sector data into (HL). A = error.

44E4 thru 44E7 [WRITS] - D = track, E = sector, C = logical drive, HL = 100 byte buffer. WRITS transfers the data in (HL) to the sector. A = error.

44E8 thru 44EA [DIRRD] - DIRRD issues a READS specifically looking for a DELETED DATA MARK (DDM). DIRRD returns if a DDM sector is read, otherwise DIRRD will read physical track zero, physical sector zero, relative byte two, and update DCT+3 with this byte (directory cylinder), then reads the new TRACK, same sector. If the new track, same sector does not return a DDM, DIRRD will try "P" density1, if "P" density doesn't return a DDM, DIRRD will try "P" density2. If still a DDM isn't found, DDIRD will return an error of 11. A = error.

44EB thru 44ED [DIRWT] - Same as WRITS, except issues DDM. A = error.

44EE thru 44F0 [RDIRP] - On entry, the B register contains the DIREC position code, and the C register contains the 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; i.e. CLOSE. This is why you should NEVER change diskettes after a file is opened, and will remain 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 will write 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 will return with the HL register pointing to the register B position code DIREC, in the I/O buffer. A = error.

44F1 thru 44F3 [WDIRP] - On entry, the B register contains the DIREC position code, the C register contains the logical drive number, and the information is in BUFF. WDIRP will exit with HL => BUFF. A = error.

44F4 thru 44F6 [USRF] - D = track, E = sector, C = logical drive, HL = buffer, and the A register contains the FCB code. USRF accepts any of the read sector, write sector, or write track commands. A = error.

44F7 thru 44F9 [GETDT] - On entry, C = logical drive. On exit the D register will contain the contents of DCT+3 for the C register DCT.

44FA thru 44FC [MODE] - Called by CLOSE to see if any data has not been written to the file.

44FD thru 44FF [OPESA] - Checks for DE => FCB opened. If open saves BC, DE, HL, IX, and IY registers, IX returned => FCB.

4500 thru 457F [DCTS] Eight drive control tables.

DCT+00: The main control byte containing the configuration infomation.

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 NEWDOS/80.
BIT	4	If set - Perform a 1.8 to 1 division to read NEWDOS/80 MOD III.
BIT	5	If set - Double sided diskettes are treated as two volume.
BIT	6	If set - 8" floppy, otherwise 5 1/4" floppy.
BIT	7	If set - Double density media, otherwise single density.

Hard disk bit pattern:			MEMDISK bit pattern:
BIT	0	Step rate LSB.	N/A
BIT	1	Step rate.	N/A
BIT	2	Step rate.	N/A
BIT	3	Step rate MSB.	N/A
BIT	4	SET TO ONE.	SET TO ONE
BIT	5	SET TO ZERO.	SET TO ZERO
BIT	6	SET TO ONE.	SET TO ONE
BIT	7	SET TO ZERO.	SET TO ONE.

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	Indicate format pattern is logged. (no update to DCT+06-0F)
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. 1 to 127. Floppies will also have the high bit set to indicate a two head one volume format.

DCT+03: The directory cylinder. This byte is obtained from cylinder zero, sector zero, relative byte two.

DCT+04: The current cylinder where the head was positioned when this drive was LAST accesed 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 accesed.

DCT+05: Dynamic head number selected.

DCT+06: The number of sectors per granule.

DCT+07: The number of granules per cylinder.

DCT+08: The number of sectors per track.

DCT+09: The number of directory file sectors on the first surface.

DCT+0A: The most significant byte for the number of sectors per cylinder.

DCT+0B: The least significant byte for the number of sectors per cylinder.

DCT+0C: The most significant byte for the cylinder offset.

DCT+0D: The least significant byte for the cylinder offset.

DCT+0E: The head offset.

DCT+0F: Not assigned.

DCT+0A thru DCT+0F are not used by floppies.

For floppies, DCT+06 thru 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
5" SD/SS	5	2	10	8
5" SD/DS	5	4	10	8
5" DD/SS	6	3	18	16
5" DD/DS	6	6	18	16
8" SD/SS	8	2	16	14
8" SD/DS	8	4	16	14
8" DD/SS	10	3	30	28
8" DD/DS	10	6	30	28

A user 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 TC=80, SPT=36, GPL=8, and DDGA=6.

```
DCT+00 = 100110xxB
DCT+02 = 10000010B
DCT+06 = 5 ("....there are still 5 sectors per granule.")
DCT+07 = 8 (GPL)
DCT+08 = 18 (SPT/2, NEWDIOS/80 double sided diskette one volume)
DCT+09 = 28 (DDGA * 5 - 2, 5 = DCT+06, 2 = 1 GAT + 1 HIT)
```

4580 thru 4CFF contain the code for the balance of DOS functions.

4E00 thru 51FF is the overlay area.

III. BASIC (All numbers are decimal unless suffixed with an H)

When editing a line the <RIGHT-ARROW> moves the cursor one space as the <SPACE-BAR>. The <E>xit function has been deleted.

The REFERENCE utility is invoked with the "@" key.

Under BASIC shorthand the user can backspace to use the single keystroke commands. ([<SHIFT>]up-arrow, [<SHIFT>]down-arrow, comma, period, and slash)

Single letter commands.

A[n][,m]	Auto line numbering starting at n (default 10) incrementing by m (defaul 10)
C	Continue program execution after STOP.
D[n][-m]	Delete lines from n to m. (Changed to have similar syntax to LIST. m does not have to be an existing line; however, at least one line should be between n and m.)
E[n]	Edit line n. (Default current line)
I	Insert. Invokes (psuedo) A .+1,1
K"program	Remove program
L[n][-m]	List lines from n to m
L"program	Load program
Mn,m	Move line n to m
Nn,m	Duplicated line n at m
P[n]	List page of lines from n (defaults current line)
R[q]	Run program starting at line q (see note)
R	Run program starting at the first line
R"program	Load and run program
S"program	Save program

The period "." may be used for n or m to represent the current line.

NOTE: q = line number or a mandatory space LABEL"label" (RUN LABEL"DOGGY")

TRON has 7 additional functions when followed by the numbers 1 - 7.

TRON or TRON 0 = Trace in the upper right corner of display.
TRON 1 = trace to line printer.
TRON 2 = display the BASIC statement in the lower left corner BEFORE it is executed.
TRON 3 = single step with delay. Delay is controlled via <CTRL><D> to increase delay, and <CTRL><F> to decrease delay. (need <CTR><S> between each delay change - see below)
TRON 4 = single step line.
TRON 5 = single step instruction.
TRON 6 = single step off.
TRON 7 = display erroneous statement.

Once TRON is envoked, <CTRL><Q> thru <CTRL><W> will modify a TRON from TRON 1 to TRON 7 respectively; i.e. <CTRL><S> will modify a TRON 2 to be TRON 2 and TRON 3. A <CTRL><V> will remove a TRON 2.

TRON or TRON 0 and TRON 1 are mutually exclusive.
TRON 4 and TRON 5 are mutually exclusive.

CMD functions

B "Soft" disable of "BREAK" key.
C Invoke "SPACE COMPRESSION" utility.
D Invoke debug.
E Interrogate last disk related error after BASIC initialized.
P Invoke "PACKER" UTILITY.
Q Dual/single dimension string sort.
S Exit BASIC.
U Invoke "UNPACKER" utility.
V Display active scalar variables.
X Invoke "REM REMOVER" utility.

CMD "uuuuu" requires a minimum of 6080 free bytes to execute.

If less than 6080 bytes are free, then the message

ignored!

will be displayed and a return to the next statement, (if any), will occur.

KEYWORD changes

CLEAR - Clears all variables, closes all OPENed files, resets execution pointer, nullifies all FOR-NEXT loops, and GOSUBS; resets ON ERROR/STOP GOTOs, resets all variables to their default type, and activates the <BREAK> key.

CLEAR nnnnn - (nnnnn = 0 to 32767, -32768 to -1) Changes the amount of space allowed for string storage, and nullifies all FOR-NEXT loops and GOSUBs. Although nnnnn can be less than - 32768, a number much less than -25000 will produce an "Out of Memory" error.

ON STOP GOTO line number/"label" - This command will deactivate the <BREAK> key in a user input mode, and cause a branch to line number/"label" if the <BREAK> key is pressed during program execution.

GOTO "label"
GOSUB "label"
RESTORE line number/"label"
ON ERROR GOTO "label"
ON n GOTO/GOSUB "label"/line number,"label"/line number,etc...
RESUME "label"
IF-THEN-ELSE "label"
RUN LABEL "label".

CLS val - Homes the cursor and sets all of video refresh RAM to value of val (0 to 255).

Keywords removed from BASIC.

```
AUTO    - USE A
CLOAD   - NOT USED
CONT    - USE C
CSAVE   - NOT USED
DELETE  - USE D
EDIT    - USE E
SYSTEM  - NOT USED
'       - USE REM
```

Keywords added to BASIC.

LABEL	LABEL "label" - defines current line as being "label".
EXIT	EXIT line number/"label" - satisfies FOR-NEXT loop without FOR parameter reaching limit.
SORT	SORT var(0) - single dimension array sort. (will add later).
IND(	PRINT IND(n) - prints n spaces from current cursor position. (n = 0 to 255)
ERASE	ERASE var(0) - removes var array from RAM. (CMD"L"var(0) - v1.6)
ZERO	ZERO var(0) - sets all elements in var array to zero if numeric or null if string. (CMD"K"var(0) v1.6)
LPOS	LPOS(0) - returns the position of the printer under software control.
HEX$	HEX$(integar val) - returns a 4 character string equivalent to integer val.
BIN$	BIN$(integer val) - returns a 16 character string equivalent to integer val.
CALL	CALL integer val - executes code to located at integer val.
WPEEK	WPEEK(integer val) - returns the WORD located at integer val.

EXAMPLES of new BASIC commands.

PROBLEM: To obtain the hexadecimal equivalent in a RAM location.

```
OLD
    20 DEFINT A-Z
    30 H$= "0123456789ABCDEF"
    40 X= PEEK(N)
    50 Y= PEEK(N+1)
    60 L1= INT (X/16)
    70 L2= X-L1*16
    80 M1= INT (Y/16)
    90 M2= Y-M1*16
   100 A$=MID$(H$,M1+1,1)+MID$(H$,M2+1,1)+MID$(H$,L1+1,1)+MID$(H$, L2+1,1)

NEW
    10 A$=HEX$(WPEEK(N))
```

PROBLEM: Prematurely exit a for-next loop.

```
OLD
     200 FOR X= 1 TO N
     210 IF A$(X)= "MATCH" THEN Y=X:X=N: NEXT: GOTO 400
     220 NEXT X
     ...
     400 PRINT A$(Y)

NEW
     200 FOR X=1 TO N
     210 IF A$(X)="MATCH" THEN EXIT 400
     220 NEXT X
     ...
     400 PRINT A$(X)
```

While programs are being developed, it is easier to use LABEL's to define various routines, than to assign and remember specific line numbers.

Rules: 1. LABEL must be the first statement in a line.
2. The "label" referenced must match in character length and case.
3. Any character other than 0 and 34 is permitted.

```
60 IF A$="R"THEN GOSUB"NEW BOARD"
.. more program lines
480 LABEL"NEW BOARD"
```

The labels may be removed after a program is developed via the use of the "*" command. This command invokes RESOLVE/BOL and replaces references to labels with line numbers. The LABEL"label" is removed from each line.

PROBLEM: Limit/control an input to a numeric variable.

OLD: Varies with the limits of the acceptable input characters.

NEW:
```
10 CLEAR
20 CLS
30 AC$ = "012345"
.. menu printed here with 5 options
80 INPUT @ 704,1,95, USING A$, "SELECTION ";S
```

SYNTAX:

```
INPUT@pos,[#]len,char,[NOT][USINGexp$],["prompt";]varu

LINEINPUT @pos, [#] len, char, [NOT] [USING exp$], ["prompt";] var$

INPUT !col, row, [#] len, char, [NOT] [USING exp$], ["prompt";] varu

LINEINPUT!col,row,[#]len,char,[NOT][USINGexp$],["prompt";]var$
```

SYMBOL	MEANING
@pos	Specifies exactly - in terms of Video Display positions - where the INPUT prompt or INPUT field - if no prompt, will begin printing. Integer expression between 0 and 1023 (64X16), or 0 and 1919 (80X24).
!col,row	Specifies exactly which column (col), and which row the INPUT prompt or INPUT field - if no prompt, will begin printing. Integer expressions between 0 and 63 for "col", and 0 to 15 for "row" (64X16), or between 0 and 79 for "col", and 0 to 23 for "row" (80X24).
#	Specifies automatic <ENTER> when INPUT field is full.
len	This is the length of the INPUT field. Integer expression between 1 and 255.
char	This is the field character. Integer expression between 1 and 255. (These are Video "POKE" values not "PRINT" values).
NOT	Mask reject indicator. (see below)
USING	Mask indicator. (see below)
exp$	String expression representing the mask characters. USING exp$ = Only use characters in exp$. NOT USING exp$ = Do not use any characters in exp$.
"prompt";	The optional prompt message.
varu	Numeric or string variable, or numeric/string variable list.
var$	A single string variable.

BASIC's CMD"command" FUNCTION

The function of CMD"command" in SUPERBASIC is to execute a /CMD file from BASIC then return to BASIC with the resident program intact. The original concept (idea) of CMD"command" was provided to the TRS-80 world by the author of NEWDOS. Other than MODEL I and MODEL III TRSDOS, the other new operating systems provide a quasi-equivalent function with their BASICs. The authors of the popular operating systems - DOSPLUS, LDOS, MULTIDOS, and NEWDOS/80 - did not talk to each other and did not come close in the method the CMD"command" function is invoked. Since we are not experts on the other operating systems, we cannot fully describe how their CMD"command" function works. Although we are registered users of each of these operating systems, we do not know if they have or will make a change in the CMD"command" code. Nevertheless, we will briefly describe the known differences in the CMD"command" functions for the MODEL I and MODEL III versions of LDOS, DOSPLUS, MULTIDOS, and NEWDOS/80.

LDOS: The BASIC CMD"command" checks if about 4K of free memory is available, then modifies the @ABORT vector (4030H), the @EXIT vector (402DH), and the @ERROR vector (4409H) with code to return to LBASIC when LBASIC and the user program is pushed into higher memory. Next TOPMEM is modified to one byte less than used for LBASIC and the user program storage, then the "command" is executed.

DOSPLUS: The BASIC CMD"command" is executed directly without regard for memory usage. Since the BASIC with DOSPLUS resides at 5700H and above, this is not a problem with programs that execute in the area of 5200H to 56FFH. All of the LIBRARY commands with DOSPLUS execute below 5700H. This requires ZERO memory free!

MULTIDOS: The BASIC CMD"command" checks if about 6K of free memory is available, then pushes SUPERBASIC and the user program into higher memory, modifies the @EXIT vector (402DH) with code to return to SUPERBASIC (in high memory). Next TOPMEM is modified to one byte less than used for SUPERBASIC and the user program storage, then the "command" is executed.

NEWDOS/80: The BASIC CMD"command" tries to execute the "command" with MINI-DOS. If MINI-DOS can execute the "command", then ZERO memory is required and execution is very fast. If MINI-DOS cannot perform the "command", then BASIC checks to see if about 9K of free memory is available, computes a checksum on the memory used by BASIC and the user program, moves the STACK POINTER to 70FFH, then executes the "command". After the "command" is complete, BASIC makes a checksum on the memory used by BASIC and the user program. If the checksum is different, then BASIC performs a re-boot.

Where there is reference to memory availability, if insufficient memory is available, LDOS and NEWDOS/80 will abort with an "Out of memory" error message, but MULTIDOS will print "ignored" then continue with the next statement.

BASIC's CMD"command" FUNCTION

In applications where BASIC is requested to load a machine language program into high memory, setting HIGH$ (4049H/404AH - MODEL I, or 4411H/4412H - MODEL III) prior to entry into BASIC would solve the problem. Since BASIC has its own highest memory available pointer and DOS does not need this area protected, this request can be honored after the user has entered BASIC.

EXAMPLE: User machine code routine will reside between F014H and FFFFH.

***** THINKING *****

Required: 1) Move the stack pointer out of the way.
2) POKE BASIC's high memory pointer.
3) Reset the balance of the pointers.

Known: 1) The BASIC high memory pointer is at 40B1H.
2) The stack pointer will move with CLEARn.
3) CLEAR resets all pointers.

The values to POKE will be in two byte form - the least significant digit to 40B1H (16561), and the most significant digit to 40B2H (16562).

Printing &H40B1 gives us 16561.
Printing &HF014 gives us -4076.
Printing &HF0 gives us 240.
Printing &H14-1 gives us 19.

***** Now that we have gathered our thoughts *****

```
10 CLEAR 4200 ' Greater than 4076
20 POKE 16561, 19 ' LSD
30 POKE 16562, 240 ' MSD
40 CLEAR ' Reset ALL pointers
```

When the user exits from BASIC, the DOS HIGH$ will be where it was prior to entering BASIC.

USING MULTIDOS WITH OTHER PROGRAMS

LAZY WRITER - The "4 + 3" version, an 80 character version that runs on a Model III DOS, will not work with Model 4 MULTIDOS. A fix is available from AlphaBit Communications. This problem should not be present in Lazy Writer's purchased after 1/85. If you have a problem, send your original Lazy Writer disk back, along with a $5.00 handling fee, and you'll receive the fixed version in the mail. If you don't want to send your disk back, you can order the fix by phone; there's an extra charge of $3.00 for the disk.

OTHER 80 CHARACTER PROGRAMS - If you have an 80 character program that runs on A MODEL III DOS, it may not work with Model 4 MULTIDOS. The program may set a bit that switches in the hardware ROM. There is no status byte in the Model III DOS to tell you to reflect the condition of the I/O port that controls the screen and the ROM. Model 4 MULTIDOS does provide an image of the screen port controller at 05AH. Additional information is available to programmers who need it to integrate programs with MULTIDOS.

FROM David Welsh *04/11/85*

Subject -
Wild card DIR feature:

In version 1.71 for the Model I, III, 4, and Max-80; the wild card feature of DIR has been enhanced to provide a "not" feature or a negative wild card. If you use a - sign in front the filename to be match, it will not show any filenames that match. For example, DIR -*/CMD , will not show any files ending in "/CMD".

MEMDISK on the Model 4 and Max-80 Multidos:

Since this manual was printed, there has been an enhancement to ***MEMDISK*** in the Model 4 (80/64) version and the Max-80 version. If you type ***MEMDISK*** without a drive number, ***MEMDISK*** will automatically become drive 0 and all the system files and BASIC will be copied to it. Not everybody wants BASIC in the ***MEMDISK***, so ***MEMDISK (S)*** has been added. ***MEMDISK (S)*** will copy only the system files to the ***MEMDISK***. With ***MEMDISK (S)***, you don't have to kill BASIC and its overlays from the ***MEMDISK*** in order to have enough room to copy in programs like Lazy Writer that use memory overlays and would therefore benefit from being in ***MEMDISK***.

MEMDISK (SYS) or ***MEMDISK (SYSTEM)*** can be used instead of ***MEMDISK (S)***.

We thought it would also be nice if you could recover a ***MEMDISK*** after leaving it with ***MEMDISK (X)*** or after a system reboot, so Vernon gave us ***MEMDISK (R)***, which recovers a ***MEMDISK*** that aready exists after a reboot or ***MEMDISK (X)***. (But please remember that nothing can recover a ***MEMDISK*** if the computer has been turned off!)

MEMDISK (OUT) or ***MEMDISK (OFF)*** or ***MEMDISK (REMOVE)*** can be used instead of ***MEMDISK (X)***. ***MEMDISK (REC)*** or ***MEMDISK (RCOVER)*** can be used instead of ***MEMDISK (R)***.

These changes in ***MEMDISK*** make the operation of the DOS smoother than ever. I frequently have my disked AUTO to a DO file that sets up the ***MEMDISK*** the way I want. The extra boot time pays off in lightening fast, quiet operations that can include having data disks in both of my two physical drives.

Sincerely,

David Welsh

David Welsh

Making a ***ONE VOLUME, DOUBLE-SIDED*** system diskette

When we first issued 1.7 Multidos, we said that it could handle a dual-sided diskette as one volume. We didn't say that you could make a system diskette into a dual-sided, one volume system diskette. Many people thought that they could do this by CONFIGing drive 1 to dual-sided, one volume and making a BACKUP of the system diskette. But system files have to be in certain places on the diskette for the DOS to work correctly and BACKUP won't put them there. VFU won't either unless they already exist in the directory.

By popular demand, we have added a program that creates a correct directory for a dual-side one volume diskette. This program is called TS/CMD or TD/CMD or T3/CMD or T4/CMD or TM/CMD; depending on which version of the DOS we are dealing with. By following the instructions below, you can create a One Volume, Double-sided system diskette, if you have dual-sided drives. This program is only supposed to work with an unaltered BACKUP of the MASTER system diskette.

If you don't have dual-sided drives, you can kill this program from your working diskette using VFU. Leave it on your MASTER. Who knows, you might want to buy dual-sided drives.

Read manual before attempting to use these instructions. Everything in italics is an example of what you would type after the MULTIDOS prompt.

The dual-sided SYSTEM diskette created by this procedure becomes the MASTER system disk and can be backed up to make more dual-sided, one volume SYSTEM disks.

Procedure for creating a dual-sided, single volume MULTIDOS system diskette using the directory skeleton utility:

MODEL I - Single density, or Double density.

1. Make a backup of the system diskette (***single side, single volume***).
2. Config a drive (V=2) sic. that is NOT logical zero.

CONFIG :1 (V=2)

```
:0, Phy = 0, 5" Floppy, double density,
    one sided, one volume, step rate = 30 mS.
:1, Phy = 1, 5" Floppy, double density,
    two sided, two volume, step rate = 30 mS.
:2, Phy = 2, 5" Floppy, double density,
    one sided, one volume, step rate = 30 mS.
:3, Nil
:4, Nil
:5, Nil
:6, Nil
:7, Nil
```

(more)

Making a *ONE VOLUME, DOUBLE-SIDED* system disk

3. Format side 1 (FORMAT :d') of the backup diskette in the drive used for step three (Single density for single density, or double density for double density).

FORMAT :1'

4. Config the drive (SI=2).

CONFIG :1 (SI=2)

```
:0, Phy = 0, 5" Floppy, double density,
    one sided, one volume, step rate = 30 mS
:1, Phy = 1, 5" Floppy, double density,
    two sided, one volume, step rate = 30 mS.
:2,
:3, Nil
:4, Nil
:5, Nil
:6, Nil
:7, Nil
```

5. Type *TS* for Single density, or *TD* for Double density on this diskette. The TS or TD program will ask for the drive number of the disk to be SYSGENed to one volume dual-sided.

6. VFU % all files except DIR/SYS and SYSRES/SYS (Use the "O" option).
Here is an example (the directory of your disk will be different):

```
MULTIDOS
VFU %
Versatile File Utility - 8.1 (c) C. E. C. 1985

Press Action
     "C" Copy
     "E" Execute
     "H" Hard copy
     "M" Move
     "P" Purge
Choice C
Condition (A/C/D/E/O/S)? O
Press "S" for selective, or "T" for total. T
Include "INVISIBLE" files? Y
Include "SYSTEM" files? Y
Source drive? 0 Destination drive? 1
```

(more)

Making a ***ONE VOLUME, DOUBLE-SIDED*** system diskette

```
Drive 0
 Disk          - MULTIDOS  - 02/16/84,   29 free grans.
 Allocate/DOL    Close/DOL       Command/DOL     DIR/SYS         Debug/DOL
 Error/DOL       Help/EXT        Library1/EXT    Library2/EXT    Minidos/DOL
 Open/DOL

     Press back arrow to select or "ENTER" if done.-
     Press "A" to abort, "ENTER" to execute, or "R" to repeat.
```

Don't forget the % sign after VFU!

7. ZAP the DCT - TRACK (CYLINDER) 0, SECTOR 6. (see below)

MODEL III, MODEL 4, and MAX-80

The procedure for the Model III,4 and Max-80 is simpler.

1. Config a drive (SI=2) - NOT logical zero.
2. Format a diskette in the drive used for step one.
3. Perform T3 - MODEL III, or T4 - MODEL 4, or TM - MAX-80 on this diskette.
4. VFU % all files except DIR/SYS (Use the "O" option).
5. ZAP the DCT - TRACK (CYLINDER) 0, SECTOR 6.

```
MULTIDOS
ZAP
Easy ZAP - Version 2.4 - (c) C. E. C. 1985.

C = Copy sectors.
D = Disk sectors.
F = File sectors.
M = Memory.
T = Track format.
V = Verify.
X = Fix directory.

-Choice  D
-Drive number ( 0 to 7 )   1
-Cylinder number ( 0 to 255/0FFH )    0
-Sector number ( 0 to 255/0FFH )    6
```

(more)

Making a ***ONE VOLUME, DOUBLE-SIDED*** system disk

ZAP this byte. Change the 01 to 82.

```
HEX 00  8300 8211 0000 0606 1210 0000 0000 0000 ................
    10  8301 8211 1100 0606 1210 0000 0000 0000 ................
    20  8302 8211 1100 0606 1210 0000 0000 0000 ................
DRV 30  8303 8211 1100 0606 1210 0000 0000 0000 ................
 1  40  00FF 0000 0000 0000 0000 0000 0000 0000 ................
    50  00FF 0000 0000 0000 0000 0000 0000 0000 ................
    60  00FF 0000 0000 0000 0000 0000 0000 0000 ................
TRK 70  00FF 0000 0000 0000 0000 0000 0000 0000 ................
000 80  0000 0000 0000 0000 CDDC 45E5 D5FD 7E00 ..........E...~.
00H 90  CB5F 282E F5FD 6E08 AF67 8230 0124 2D20 ._(...n..g.0.$-
    A0  F955 6F19 3E12 5FFD CB02 7E28 0187 CD78 .Uo.>._...~(...x
SEC B0  0455 BB38 0693 213C 46CB E65F F1CB 6720 .U.8..!<F.._..g
006 C0  0114 CB57 21ED 37F5 2804 CB22 CB06 2C7B ...W!.7.(.."..,{
06H D0  2F77 2C7A 2F77 F1F6 18D1 E1C9 E5C5 2123 /w,z/w........!#
    E0  4479 E607 47A9 4F7E 70B8 F521 ED37 2811 Dy..G.O~p..!.7(.
STD F0  CD83 467E 2FFD 7704 78CD 8346 FD7E 042F ..F~/.w.x..F.~./

Press "ENTER" to update, "BREAK" to abort, or "@" to see buffer.
-Drive number ( 0 to 7 )
```

Typical TRACK 0, SECTOR 6

FROM David Welsh *05/07/85*

NOTE to MODEL I Single Density customers. The Single Density version of MULTIDOS comes on a double-sided "flippy disk". A flippy is like two disks in one and either side can be read by a sinlge sided drive. Just turn it over.

Both sides contain the operating system, but neither side has a full set of utilities. Everything that the 1.7 version of MULTIDOS provides will not fit on a Single Density diskette. We suggest you BACKUP both sides. Read the manual, then make up your working disks with only those utilities that you will be using most. The others can always be used as needed from another disk. VFU makes customizing your working disk easy.

Here is side 1 of the SINGLE DENSITY 1.71 MULTIDOS. Files that only appear on this are in bold italic:

1 MULTIDOS 02/11/85 35 log 35 phy cyls 0 grans 0.00 K

Filename	Date	Eof	Lrl	Lrec	Seg	Grans
Allocate/DOL S	02/16/85	3/7	256	4	1	1
BACKUP/CMD I	02/04/85	17/17	256	18	1	4
BASIC/CMD I	03/30/85	16/25	256	17	1	4
BBASIC/CMD I	03/30/85	22/64	256	23	1	5
CAT/CMD P	***02/17/85***	***9/203***	***256***	***10***	***1***	***2***
COPY/CMD I	02/17/85	4/255	256	5	1	1
CREF/BOL S	11/13/84	4/252	256	5	1	1
Close/DOL S	01/27/85	3/78	256	4	1	1
Command/DOL S	12/27/84	5/0	256	5	1	1
DBLFIX/CMD	***02/17/85***	***2/221***	***256***	***3***	***1***	***1***
DDT/CMD	***07/13/84***	***4/32***	***256***	***5***	***1***	***1***
DIR/SYS S	02/11/85	10/0	256	10	1	2
Debug/DOL S	08/22/84	4/246	256	5	1	1
EDIT/BOL S	05/25/84	5/0	256	5	1	1
ERROR/BOL S	07/07/84	4/210	256	5	1	1
Error/DOL S	02/17/85	4/107	256	5	1	1
FORMAT/CMD I	02/17/85	14/247	256	15	1	3
GR/CMD	***07/25/84***	***1/3***	***256***	***2***	***1***	***1***
Library1/EXT S	02/17/85	21/206	256	22	1	5
Library2/EXT S	02/19/85	22/105	256	23	1	5
MEM/CMD P	***07/13/84***	***1/195***	***256***	***2***	***1***	***1***
MEMDISK/CMD	***01/11/85***	***3/198***	***256***	***4***	***1***	***1***
Minidos/DOL S	02/17/85	4/254	256	5	1	1
Open/DOL S	01/21/85	4/144	256	5	1	1
PACK/BOL S	06/30/84	4/229	256	5	1	1
PRT/CMD	12/27/84	3/210	256	4	1	1
RENUM/BOL S	11/13/84	5/0	256	5	1	1
RS/CMD P	***06/30/84***	***4/227***	***256***	***5***	***1***	***1***
SPOOL/CMD	***12/27/84***	***4/172***	***256***	***5***	***1***	***1***
SYSRES/SYS S	02/11/85	20/0	256	20	1	4
TAPE/CMD	***07/13/84***	***7/68***	***256***	***8***	***1***	***2***
UNPACK/BOL S	11/13/84	5/0	256	5	1	1
UTIL/BOL S	02/17/85	4/176	256	5	1	1
VFU/CMD P	02/17/85	17/107	256	18	1	4
ZAP/CMD P	02/17/85	31/210	256	32	1	7

FROM David Welsh *05/07/85*

Here is side 2 of the MULTIDOS MODEL I SINGLE DENSITY 1.71. Files that only appear on this are in bold italic:

1 MULTIDOS 02/11/85 35 log 35 phy cyls 0 grans 0.00 K

Filename	Date	Eof	Lrl	Lrec	Seg	Grans
Allocate/DOL S	02/16/85	3/7	256	4	1	1
BACKUP/CMD I	02/04/85	17/17	256	18	1	4
BASIC/CMD I	03/30/85	16/25	256	17	1	4
BBASIC/CMD	03/30/85	22/64	256	23	3	5
CDIR/CMD	***02/17/85***	***2/255***	***256***	***3***	***1***	***1***
COPY/CMD I	02/17/85	4/255	256	5	1	1
CREF/BOL S	11/13/84	4/252	256	5	1	1
Close/DOL S	01/27/85	3/78	256	4	1	1
Command/DOL S	12/27/84	5/0	256	5	1	1
DIR/SYS S	02/11/85	10/0	256	10	1	2
Debug/DOL S	08/22/84	4/246	256	5	1	1
EDIT/BOL S	05/25/84	5/0	256	5	1	1
ERROR/BOL S	07/07/84	4/210	256	5	1	1
Error/DOL S	02/17/85	4/107	256	5	1	1
FMAP/CMD	***02/17/85***	***4/241***	***256***	***5***	***1***	***1***
FORMAT/CMD I	02/17/85	14/247	256	15	1	3
HELP/CMD	***09/24/84***	***23/95***	***256***	***24***	***1***	***5***
LO/CMD	***11/17/84***	***8/209***	***256***	***9***	***1***	***2***
Library1/EXT S	02/17/85	21/206	256	22	1	5
Library2/EXT S	02/19/85	22/105	256	23	1	5
Minidos/DOL S	02/17/85	4/254	256	5	1	1
Open/DOL S	01/21/85	4/144	256	5	1	1
PACK/BOL S	06/30/84	4/229	256	5	1	1
RENUM/BOL S	11/13/84	5/0	256	5	1	1
SYSRES/SYS S	02/11/85	20/0	256	20	1	4
TS/CMD	***04/10/85***	***11/44***	***256***	***12***	***1***	***3***
UNPACK/BOL S	11/13/84	5/0	256	5	1	1
UTIL/BOL S	02/17/85	4/176	256	5	1	1
VFU/CMD	02/17/85	17/107	256	18	1	4
ZAP/CMD	02/17/85	31/210	256	32	2	7

REFERENCE Cross reference variables, and integers up to 9999999.

;[p][[#]xxx]<ENTER>

p = listing directive. If p = "*" the reference listing is to the display. If p = "$" the the reference listing is to the printer and the display.

xxx = Reference target:
(1) A one or two character variable name without a type suffix.
(2) An integer number which may be a line number or a value used in the program.
(3) A 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 will be displayed sequentially, with each additional press of <;> alone.

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

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

EXAMPLES:

;*<ENTER>

All integer and variable references are displayed on the screen.

;K<ENTER>

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

;$G<ENTER>

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

;#PRINT<ENTER>

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

;$#<ENTER>

All key words will have their references directed to the video and line printer.