M.BASIC and MDOS

Page number in Chapter 5

M.BASIC COMMANDS FOR WRITING AND RUNNING PROGRAMS

EDIT <linenumber>	Enter edit command mode	3
(SPACE)	Advance the edit pointer	3
C<new character>	Change the next character in the edit buffer	3
D	Delete the next character	4
I<new characters>	Insert new characters into the line	4
L	List the line in the special editing buffer	4
S<character>	Search to a specified character	4
K<character>	Delete to a specified character	4
(RETURN)	Replace line in file and exit edit mode	4.1
Q	Quit the edit mode; leave original line unchanged	4.1
RENUM [<starting no.>] [<increment>] [<start line>]	Renumber file lines	4.1
MERGE "[<unit>:]<filename>"	Merge program on disk line by line into current file	4.3
DELETE [[<linenumber>]-[<linenumber>]]	Delete lines from current program	4.6
LIST [[<linenumber>]-[<linenumber>]]	Display some or all of current program	4.6
RUN	Execute program currently in program buffer	6
(control-)C	Interrupt a running program	7
CONT	Continue executing an interrupted program	7

M.BASIC CONSTANTS AND VARIABLES

[-]<n...n>	Integer format	9	[-]<xxRn...n>	Integer in Radix xx	9
[-]<n...n.n...n>	Real format	9	[-]<n...n.n...nE[-]xx	Scientific format	9
"<characters>"	String format	10			
<one letter>%	Integer variable	10	<one letter>[<one digit>]	Real variable	10
<one letter>$	String variable	10			

M.BASIC OPERATORS

+ Addition	- Subtraction	* Multiplication	/ Division	14
\ Integer division	^ Exponentiation	+ String concatenation		
< Less than	> Greater than	= Equal to		15
<= Less than or equal to	>= Greater than or equal to	<> Not equal to .		
AND logical AND	OR logical OR	NOT logical NOT		16

M.BASIC NUMERIC FUNCTIONS 18

x and y stand for numeric expressions:

ABS(x)	Absolute value	ATN(x)	Arctangent in radians
COS(x)	Cosine of angle in radians	EXP(x)	Exponentiation
FIX(x)	Truncate fractional part	FRAC(x)	Fractional part
INT(x)	Greatest integer not greater than	LN(x)	Logarithm to base e
LOG(x)	Logarithm to Base 10	MAX(x,y)	Greatest of the two values
MIN(x,y)	Lesser of the two values	MOD(x,y)	x modulo y
RND(x)	Random number using x as seed	SGN(x)	+1 if pos., -1 if neg., 0 if 0
SIN(x)	Sine of angle in radians	SQR(x)	Square Root
TAN(x)	Tangent in radians		

M.BASIC STRING FUNCTIONS 20

x, y and n stand for numeric expressions; x$ and y$ stand for string expressions.

ASC(x$)	ASCII code of first char. in x$	LEFT$(x$,n)	n leftmost characters of x$
CHAR$(x)	Character whose ASCII code is x	LEN(x$)	Length of x$
FMT(x,y$)	Give x as a string modeling y$	MID$(x$,n,[y])	y char's of x$ beg at n'th
NON PRINTING CHARACTERS IN y$:		MAX(x$,y$)	The greater (by ASCII code)
9	digit; leading 0's become "0"'s	MIN(x$,y$)	The lesser (by ASCII code)
Z	digit; leading 0's become blanks	REPEAT$(x$,n)	x$ repeated n times
V	decimal point location	RIGHT$(x$,n)	n rightmost characters of x$
$	digit; print $ where appropriate	STR$(n)	n converted to a string
*	digit; leading 0's become "*"'s	VAL (x$)	x$ converted to a number
,	gives a blank, *, or $ as needed	VERIFY(x$,y$)	Pos of first char not in y$
INDEX(x$,y$)	Position in x$ of first y$		

MISC. M.BASIC FUNCTIONS 22

IN(x)	Input from I/O port x	PGMSIZE	Size of current program in bytes
PEEK(x)	Contents of memory location x	SPACELEFT	Bytes left in program buffer

M.BASIC STATEMENTS

DATA <numeric or string constant>,...,	Data to be assigned to variables by a READ	36
DEF FN<letter>[(<perameter name>)] = <expression>	User defined function	37
DEF FA<letter> = <start address>	Assembly lang. function	37
DIM <letter>[%] (<size>,...<size>)	Sizes of 1 to 4 dimensions in array <letter>	38
DIM <letter>$ ([<size>,...]<length>)	Sizes of 0 to 4 dim's & length of string array	38
END	Physical end of program file	38
EXEC <string expression>	Execute string expression as a BASIC statement	39
FLOW	Enable trace mode (display each program line when executed)	39
FOR <num. var.> = <num. expr.> TO <num. expr.> [STEP <num. expr.>]	Initiate loop	40
GOSUB <linenumber>	Execute subroutine	42
GOTO <linenumber>	Transfer control	43
IF <logical expression> THEN <linenumber>	Conditional transfer of control	43
IF <log. expr.> [THEN] STATEMENT [:STATEMENT]...	Conditional execution of statements	43
INPUT ["<prompt>"<; or ,>] <variable>[;variable]...	Wait for input from console	44
[LET] <variable> = <expression>	Assign value of <expression> to <variable>	44
MEMEND <numeric expression>	Define upper limit of memory used by M.BASIC	45
NEXT <numeric variable>	Terminate loop begun by FOR and increment counter	45
NOFLOW	Disable trace mode	45
ON <num. expr.> <GOTO or GOSUB> <lineno.>[,<lineno.>]...	Variable transfer of control	45
OUT (<port number>) = <num. expr>	Output to port	46
POKE (<address>) = <num. expr.>	Store in given memory address	46
PRINT <expr.>[<, or ;>]...[TAB(<num. expr.>)[<, or ;>]]...	Display values	47
READ <variable> [,<variable>]...	Give variable(s) value(s) found in DATA statement	49
REM <remark text>	Non-executed remark for documentation purposes	49
RESTORE [<linenumber>]	Position DATA list pointer	49
RETURN	Return from subroutine to calling routine	49
SIZES (<rsize>,<isize>,<ssize>,[<prog.size>])	Allocate number of bytes of storage	50
STOP	Stops program execution; continue with CONT	50
STRING "<string delimiter>"	Define string delimiter for INPUT and GET statements	50

M.BASIC DISK FILE I/O STATEMENTS AND COMMANDS

DISPLAY "[<unit>:]DIR"	Display directory of disk in drive	53
LOAD "[<unit>:]<filename>"	Load program or object file into memory	53
PLOADG "[<unit>:]<filename>"	Load and execute program file	53
SAVE "[N:][<unit>:]<filename>"[<startaddr>,<endaddr>]	Save file on disk	54
SCRATCH "[<unit>:]<filename>"	Delete any file from disk	54.1
CHAIN "[<unit>:]<filename>"	Load and execute next program segment	54.1
LINK "[<unit>:]<filename>"	Load and execute overlay file	54.1
OPEN <filenumber>"[N:][<unit>:]<filename>" [illegible] [END<line no.>] [ERROR<line no.>]	Open disk file for program access	55
PUT <filenumber> [RECORD <rec.num.>] <expr>[<, or ;>]...	Store data on disk	57
GET <filenumber> [RECORD <rec.num.>] <expr>[,<expr>]...	Get data from disk	60
CLOSE <filenumber>	Close file	60
ATTRS (<fileno.>)=<sum 16 (prog), 8 (obj), 2 (perm), 1 (write prot)>	File attributes	61
EOF (<filenumber>) = <file length>	Set file length parameter	61
FREESPACE <filenumber>	De-allocate unused tracks allocated to a file	62
GETSEEK (<filenumber>) = <new GET pointer>	Set sequential GET pointer	62
PUTSEEK (<filenumber>) = <new PUT pointer>	Set sequential PUT pointer	62
RENAME (<filenumber>) = "<new filename>"	Change name of a file	63

M.BASIC DISK I/O FUNCTIONS 64

ATTR(<fileno>)	Attribute parameter	RECPUT (<fileno>)	Val of seq PUT pointer
ERR	Error code of last disk error	SIZE (<fileno>)	File size in records
ERR$	Error message of last disk error	TRACKS (<fileno>)	Number of tracks
NAME (<fileno>)	Name of the file	FREETR (<fileno)>	Number of free tracks
RECGET (<fileno>)	Val of seq GET pointer		

M.BASIC PRINT FILE OUTPUT

OPEN <filenumber>"<*P,*T,or *N>" [PAGESIZE <number of lines>] [ENDPAGE<line number>]	Open output file on printer, terminal, or null device	65
PUT <filenumber> <expr>[<, or ;>]...	Output to printer or terminal	66
CLOSE <filenumber>	Close file	66
ENDPAGE <filenumber>	Position output device to top of next page	67
ASSIGN <device #> <logical stream mask> [<width> [<null count>]]	I/O control	67
LISTP [[<linenumber>]-[<linenumber>]]	Print some or all of current program	69
PAGESIZE <number of lines per page>	Set size of program listing pages	69

MDOS EXECUTIVE COMMANDS

COMP <start blk 1> <end blk 1> <start blk 2>	Compare two blocks of data
DUMP <start> [<end>]	Hex dump of memory
ENTR <start>	Enter data in memory
FILL <start> <end> <byte>	Fill block of memory with a constant
MOVE <source start> <source end> <destination>	Move a block of memory
SEAR <start> <end> <byte>	Search a block for a particular byte
SEARN <start> <end> <byte>	Search a block for non occurrence of a byte
CREATE "[<unit>:]<filename>" [<filetype>]	New directory entry is created
DISP "[<unit>:]<filename>" [<record number>]	Hex dump of file on disk
FILES [<unit>]	Output formatted display of disk directory
FREE [<unit>]	Output the number of free tracks
SCRATCH "[<unit>:]<filename>"	Remove a named file from the disk directory
LOAD "[<unit>:]<filename>" [<start>]	Load a named file from disk
SAVE "[<unit>:]<filename>" <start> <end> [<file type>] [<exec.addr.>]	Save new file
RENAME "[<unit>:]<filename>" "<new name>"	Change the name of a disk file
TYPE "[<unit>:]<filename>" <type>	Change the file type on the directory
APP ["<ASCII>" "<ASCII>"] [<hex> <hex>]	Transfer program control to 2800
ASSIGN <device #> <logical stream mask> [<width> [<null count>]]	I/O control
EXEC <address>	Execute object code
MATH <hex number> <hex number>	Do hex arithmetic
PROMPT "<ASCII>"	Change the prompt string to an arbitrary string
INIT <unit>	Initialize a diskette in the indicated unit
ZSM "[<unit>:]<sourcefile>" "[<unit>:]<objectfile>" "<options>" [<offset>]	Assemble
OPTIONS: E (only errors) P (paginated listing) S (print listing only) M (memory image) L (delete line numbers) T (print symbol table)	
DEBUG-XX (XX is version number created by DEBUG-GEN)	DEBUG utility
DEBUG-GEN	DEBUG Generation utility
LINEEDIT	MDOS Line Editor
[<unit>:]SYMSAVE "<filename>" ["<mask string>"]	Create EQUates from Symbol Table
[<unit>:]FILECOPY "[<unit>:]<filename>" "[<unit>][:][<newfilename>]"	Copy File
[<unit>:]COPYFILE "[<unit>:]<filename>"	Copy file to same drive but different disk
DISKCOPY	Copy disk from one drive to another

LINEEDIT COMMANDS

CLEAR	Clear file text from memory
NAME "<filename>"	Name the current text file
FILE	Display all file parameters
AUTO <number>	Set the auto linenumber increment
PROMPT "<message>"	Change the prompt string
LOAD "[<unit>:]<filename>"	Load a text file into memory
APPEND "[<unit>:]<filename>"	Concatenate a file to the existing file
SAVE [<unit>]	Save the current file on disk
RESAVE [<unit>]	Save an old file on disk
LIST [<linenumber 1>] [<linenumber 2>]	Output a formatted display
LISTP [<linenumber 1>] [<linenumber 2>]	Output formatted display to printer
PRINT [<linenumber 1>] [<linenumber 2>]	Output unformatted display
PRINTP [<linenumber1>] [<linenumber 2>]	Output unformatted display to printer
TAB [<op code col>] [<operand col>] [<comment col>]	Set tabs for formatted output
DELT <linenumber 1> [<linenumber 2>]	Delete lines from file
RENUM [<starting no.>] [<increment>] [<start line>]	Renumber file lines
SEARCH [<linenumber 1>] [<linenumber 2>]	Invoke search mode using mask
SEARCHALL [<linenumber 1>] [<linenumber 2>]	Search comment lines also
CHANGE [<linenumber 1>] [<linenumber 2>]	Global search and replace
CHANGEALL [<linenumber 1>] [<linenumber 2>]	As above including comments
EDIT <linenumber>	Enter edit command mode
(SPACE)	Advance the edit pointer
C<new character>	Change the next character in the edit buffer
D	Delete the next character
I<new characters>	Insert new characters into the line
L	List the line in the special editing buffer
S<character>	Search to a specified character
K<character>	Delete to a specified character
(RETURN)	Replace line in file and exit edit mode
Q	Quit the edit mode; leave original line unchanged
DOS	Exit from the line editor and return to MDOS

ASSEMBLER DIRECTIVES

ORG	Set the value of the assembler program counter to the value of the operand
LINK '<source file>'	Permits additional source files to be linked from the disk
END [<execution address>]	Identifies the physical end of the source file
EQU <value>	Equates a literal value to the line's label
REQ ['<prompt>']	Inputs a numeric argument from the console keyboard
PRT ['<text>'],[<expression>],...	Displays given information on console
TAB [<op code col>] [<operand col>] [<comment col>]	Set tabs for formatted output
NLIST	Suppresses the listing of the assembly from here on
LIST	Enable listing to the printer as it is encountered
FORM	Produce a form feed in the listing when encountered
DB <byte>,[<byte>],...	Define storage with operands evaluating to one byte
Z	Same as DB 0
DW <word>,[<word>],...	Define storage byte pairs in low/high sequence
DD <word>,...	As above except in high/low sequence
DT '<text>'	Define a line of text containing any ASCII literal characters
DTZ '<text>'	Define a line of text as above except terminated in zero
DTH '<text>'	As DT except the last byte is ORed with 80H
DS <expression evaluating to 16 bits>	Reserve storage for arbitrary number of bytes
FILL <8 bit expression>,<8 bit exp.>	Fill locations with the second argument
IFF <operand>	Conditional assembly of a block of code if the argument is zero
IFT <operand>	Same as above except if the argument is nonzero
ENDIF	Define the end of a conditional assembly block (can be nested)

ASSEMBLER ERROR CODES

A Argument error	D Duplicate label	J Jump relative error
L Label error	M Missing label error	O Opcode error
R Register error	S Syntax error	U Undefined symbol error
V Value error		

ASSEMBLER OPERATORS

+ Arithmetic sum	- Arithmetic difference	* Arithmetic product
/ Integer quotient	% Integer remainder	& Bitwise logical AND
! Bitwise logical OR	# Bitwise logical EXCLUSIVE-OR	
> <operand> Right rotational operator	< <operand> Left rotational operator	

MDOS FILE TYPES

00-03 MDOS & BASIC data files	
04-07 Editor/Assembler source files	
08-0B Assembler object & BASIC "save memory" files	
0C-0F Executable overlay files	
10-13 BASIC program files	Protect Status (LS 2 bits):
14-17 Executable system files	0=Read/Write File
18-1B Executable user files	1=Read Only File
1C-7F Reserved for future expansion	2=Permanent Read/Write File
80-FF Available for user definition	3=Permanent Read Only File

DEBUG COMMANDS

COMP; DUMP; ENTR; FILL; MOVE; SEAR; SEARN; MATH; EXEC	Same as in MDOS Executive
LIST <start addr.> <end addr.>	List in instruction mnemonics
DISR	Display processor state
<register name> <hex value>	Set value of register
REGISTER NAMES: A, B, C, D, E, H, L, BC, DE, HL, SP, PC, @SP (top of stack)	
FZ; FNZ; FC; FNC; FP; FM; FPE; FPO; FH; FNH	Set or reset processor flag
RST <vector number>	Change restart vector
SET <breakpoint number> <address>	Define a permanent breakpoint
DISB	Display all current breakpoints
CLR [<breakpoint number>]	Clear one or all breakpoints
EXEC <start addr.>	Execute program but return to DEBUG when breakpoint is reached
REPT <breakpt. number> <repeat count>	Execute until breakpt. is hit <count> times
CONT [<break1>[<break2>[<break3>[<break4>]]]]	Execute & display state at up to 4 pts
RET	Execute & display state at breakpt. on top of stack
(SPACE)	Execute next instruction only, and display proc. state
TRACE	Execute program and display proc. state after each intructio