Microsoft
BASIC-80
CP/M Version

Reference Guide

for the

Heath/Zenith 8-bit Digital Computer Systems

597-2335

DATA TYPE DECLARATION CHARACTERS

Character	Data Type	Examples
$	String	ZDS$,WLW$
%	Integer	1%,VALUE%
!	Single-Precision	V!,FLAG%
#	Double-Precision	DP#,PL#
D	Double-Precision	1.23456789D-12
E	Single-Precision (exponential notation)	1.23456E+23

ARITHMETIC OPERATORS

Operator	Operation Performed
+	addition
–	subtraction
*	multiplication
/	division (floating point)
\	integer division
^	exponentiation
MOD	returns remainder after division

STRING OPERATOR

+	concatenate (string together)	"A"+"B"+"C"

RELATIONAL OPERATORS

Operator	Numeric Expressions	String Expressions
<	less than	precedes
>	greater than	follows
=	equal to	equals
<= or =<	less than or equal to	precedes or equals
>= or =>	greater than or equal to	follows or equals
<> or ><	does not equal	does not equal

LOGICAL OPERATORS

Operator	Function
NOT	bitwise negation
AND	bitwise disjunction
OR	bitwise conjunction
XOR	bitwise exclusive OR
IMP	bitwise implication
EQV	bitwise equivalence

COMMANDS

Command/Function	Examples
*AUTO<line number>,<increment>	
Enable automatic line numbering.	AUTO
*CLEAR,<expression1>,<expression2>	
Set numeric values to zero, strings to null. Optional first arguments sets end of memory. Optional second argument sets amount of stack space.	CLEAR CLEAR,32768 CLEAR,32768,1000

* Denotes a command not implemented in the BASIC Compiler.

*CONT Continues program execution.	CONT
*DELETE<line number>-<line number> Deletes the specified line numbers in the current program.	DELETE 100 DELETE 10-200
*EDIT<line number> Enter Edit Mode.	EDIT 100
FILES"<filename>" List names of files residing on the current disk.	FILES "*.BAS"
*LIST <line number>-<line number> *LLIST<line number>-<line number>List List the current program (LLIST prints on line printer).	LIST LIST 10-100 LLIST LLIST 10-100
*LOAD<"filename">,R Load a program file from disk. R option means RUN.	LOAD"B.GAME"
*MERGE<"filename"> Merges a disk file with the current program.	MERGE"B:TEST.BAS"
*NEW Deletes the current program and clears all variables.	NEW
*RENUM<nn>,<mm>,<ii> Renumbers program lines starting at line <mm>, as line <nn>, with increments of <ii>.	RENUM RENUM 300,,5 RENUM 1000,900,20
RESET Change disk in default drive.	RESET
RUN<line number> Executes the current program	RUN 100 RUN
RUN<"filename">,R Loads a program from disk and executes it. R keeps all data files open.	RUN"PROG1" RUN"C:GAME",R
*SAVE "filename" SAVE"filename"A SAVE"filename",P Saves the current program on disk. A means ASCII, P means protected format.	SAVE"COM2",A SAVE"TEST1"
SYSTEM Closes all files and performs a CP/M warm start.	SYSTEM

* Denotes a command not implemented in the BASIC Compiler.

EDIT MODE SUBCOMMANDS AND FUNCTIONS

Command	Function
RETURN	End editing and return to Command Mode.
<i> Space Bar	Move curson i spaces to the right.
<i>BACK SPACE	Move cursor i spaces to the left.
L	List remainder of program line.
X	List remainder of program line and go into Insert Mode.
I	Insert text at the current cursor position.
A	Restart edit at the beginning of the line.
E	End editing, save all changes and return to Command Mode.
Q	End editing, cancel all changes and return to Command Mode.
H	Delete remainder of line and then enter Insert Mode.
<i>D	Delete specified number of characters i beginning at current cursor position.
<i>C	Change (or replace) <i> characters.
<i>S<c>	Move the cursor to the <i>th occurance of <c>.
<i>K<c>	Delete all characters to the <i>th occurance of <c>.

PRINT USING FORMAT FIELD SPECIFIERS

Numeric Specifier	Function	Example
#	Numeric field.	###
.	Decimal point position.	##.##
+	Print leading or trailing signs (plus for positive numbers, minus for negative numbers).	+##.##
–	Print trailing sign only if value printed is negative.	##.##–
**	Fill leading blanks with asterisks.	**##.##
$$	Place dollar sign immediately to left of leading digit.	$$###.##
**$	Asterisk fill and floating dollar sign.	**$###.##
,	Use comma every three digits (left of decimal point only).	##,###.##
^^^^	Exponential format. Number is aligned so leading digit is non-zero.	#.##^^^^
String Specifier		
!	Single character	!

\<spaces>\	2 + number of spaces in character field	\ \
&	Variable length string field.	&
Literal Specifier		
_	Literal character string field	_!

PROGRAM STATEMENTS

Statement/Function	Examples
Data Type Definition	
DEFINT/SNG/DBL/STR	
Declare range of variable as specified variable type.	DEFINT I-N DEFSNG A-H, O-P DEFDBL X, Y, Z
Assignment and Allocation	
DIM<list of subscripted variables>	
Allocate storage for array.	DIM A(20),B(12,2)
OPTION BASE n	
Declare minimum value for array subscript.	OPTION BASE 1
ERASE<list of array names>	
Remove an array from the program.	ERASE A,B
LET<variable> = <expression>	
Assign value of expression to variable.	LET SUM = A+B+C
REM<remark>	
Insert remark into program.	REM GRP IS GROSS PAY
SWAP<variable>,<variable>	
Exchange the values of two variables.	SWAP A,B
Sequence of Execution	
END	
Terminate program execution	100 END
FOR<V>=<X> TO <Y> STEP <Z>	
Allows repetitive execution of a series of statements.	FOR I = 1 TO 100
GOSUB<line number>	
Branch to subroutine beginning at <line number>.	GOSUB 100
GOTO<line number>	
Branch to specified line number.	GOTO 400

NEXT<variable>

Terminates a FOR loop.

```
NEXT I
```

ON<expression>GOTO line1,...linek

Evaluate expression and branch.

```
ON L+1 GOTO 10, 20, 30
```

ON<expression> GOSUB line1,...linek

Same as on ON/GOTO except branch is to a subroutine.

```
ON L GOSUB 300,400
```

RETURN

Terminates a subroutine.

```
RETURN
```

STOP

Terminates program execution

```
STOP
```

Conditional Execution

IF<expression> THEN <statement(s)>
ELSE<statement(s)>

Evaluate <expression>: If true, execute THEN clause. If false, execute ELSE clause (if present).

```
IF A=0 THEN A=1
ELSE A=0
```

WHILE <expression>

<loop statements>

WEND

Executes a series of statements in a loop as long as a given condition is true.

```
WHILE A=0
PRINT "ZERO"
WEND
```

Non-Disk I/O Statements

INPUT<"prompt string">;<list of variables>

Inputs data from the terminal.

```
INPUT "AGE";A
```

LINE INPUT[;]<"prompt string">;<string variable>

Inputs an entire line (up to 255 character) to a string.

```
LINE INPUT; J$
```

DATA<list of contents>

Stores constants.

```
DATA 34, 23, 1, 45, 0
```

PRINT<list of expression>

Outputs data on the terminal.

```
PRINT "HELLO"
```

READ<list of variables>

Reads data into specified list of variables.

```
READ I, A, B
```

RESTORE<line number>

Resets DATA pointer.

```
RESTORE
```

LPRINT<list of expressions>

Prints data on the line printer.

```
LPRINT "HELLO"
```

STRING FUNCTIONS

Function	Operation	Example
ASC(X$)	Returns ASCII code.	ASC("B")
CHR$(I)	Returns a one-character string whose character has the ASCII code of I.	CHR$(66) CHR$(N)
HEX$(X)	Converts number to hex.	HEX$(100)
INKEY$	Reads one character from the keyboard.	A$=INKEY$
INPUT$(X,Y)	Reads X characters from the keyboard or from file number Y.	INPUT$(1,1)
INSTR(I,X$,Y$)	Returns the position of the first occurrence of Y$ in X$ starting at position I.	INSTR(A$,",")
LEFT$(X$,I)	Returns left-most I characters of the string expression X$.	LEFT$(A$,1) LEFT$(C$,3)
LEN(X$)	Returns length of string X$.	LEN(A$)
MID$(X$,I,J)	Returns string of length J characters from X$ beginning with the Ith character.	MID$(X$,5,10)
MID$(X$,I,J)=Y$	Replaces the characters in X$, beginning at position I. with the characters in Y$. J characters will be replaced.	MID$(A$,1,2)="Z"
OCT$(X)	Converts the numeric expression X to an octal string.	OCT$(24)
RIGHT$(X$,I)	Returns the rightmost I characters of string X$.	RIGHT$(X$,8)
SPACE$(X)	Returns a string of X spaces.	SPACE$(20)
STR$(X)	Converts a numeric expression to a string.	STR$(100)
STRING$(I,J)	Returns a string of length I containing characters with the ASCII code J.	STRING$(20,33)
STRING$(I,X$)	Returns a string of length I containing the first character of string X$.	STRING$(20,"!")
VAL(X$)	Converts the string X$ to a numeric value.	VAL("3.14")

ARITHMETIC FUNCTIONS

Function	Operation	Example
ABS(X)	Returns absolute value.	ABS(-1)
ATN(X)	Returns arctangent of X.	ANT(3)
CDBL(X)	Converts X to double-precision.	CDBL(A)
CINT(X)	Converts X to an integer by rounding.	CINT(46.6)
COS(X)	Returns the cosine of X (X must be in radians).	COS(A+B)

CSNG(X)	Converts X to single-precision.	CSNG(V)
EXP(X)	Returns e to the power of X.	EXP(34.5)
FIX(X)	Returns truncated integer portion of X.	FIX(23.2)
INT(X)	Returns largest integer not greater than X.	INT(-12.11)
LOG(X)	Returns the natural logarithm.	LOG(45/7)
RND(X)	Returns a random number between 0 and 1.	RND(0) =SAME NUMBER RND(1) =DIFFERENT NUMBER
SGN(X)	Returns -1 for negative X, 0 for zero X, + for positive X.	SGN(C/A)
SIN(X)	Returns the sine of X.	(SIN(A*1.3)
SQR(X)	Returns the square root of X. X must be non-negative.	SQR(A*B)
TAN(X)	Returns the tangent of X.	TAN(X+Y+Z)

SPECIAL FUNCTIONS

Function	Operation	Example
FRE(X)	Returns memory space not used by BASIC-80.	FRE(0)
INP(I)	Returns the byte read from port I.	INP(255)
LPOS(X)	Returns current position of line printer print head within the line printer buffer.	LPOS(0)
NULL(X)	Sets the number of nulls.	NULL(3)
OUT I,J	Sends byte J to port I.	OUT 127,255
PEEK(I)	Reads a byte from the specified memory address.	PEEK(8192)
POKE I,J	Puts byte J into memory location I.	POKE 8192,200
POS(X)	Returns current cursor position.	POS(1)
SPC(I)	Prints I spaces on the terminal.	PRINT SPC(5)
TAB(I)	Tabs carriage to specified position.	PRINT TAB(20)
VARPTR(X)	Returns address of variable in memory.	VARPTR(V)
WAIT I,J[,K]	Status of port I is XOR'ed with K and AND'ed with J. Continued execution awaits non zero result.	WAIT 21,1
WIDTH I	Sets the printed line width.	WIDTH 80
WIDTH LPRINT I	Sets the line printer width.	WIDTH LPRINT 132

SPECIAL FEATURES

Statement/Function	Example
Error Trapping	
ON ERROR GOTO<line number>	
Enables error trapping.	ON ERROR GOTO 100
RESUME<line number>	
Continues program execution.	RESUME
ERROR<integer expression>	
Simulates the occurance of an error, also allows error codes to be defined by user.	ERROR 10
ERL	
Error line number	PRINT ERL
ERR	
Error code number	PRINT ERR
Trace Flag	
TRON	
Enables trace flag	TRON
TROFF	
Disables trace flag	TROFF
Overlay Management	
CHAIN [MERGE]"<filename>"[,[<line number>] [,ALL] [,DELETE<RANGE>]]	
Calls program and passes variables from the current program.	CHAIN "PROG"
*COMMON<list of variables>	
Pass variables to a chained program.	COMMON A,B

DISK INPUT/OUTPUT STATEMENTS

Statement/Function	Example
CLOSE#<file number>[,<file name>]	
Closes disk files.	CLOSE#6
FIELD#<file number>,<field size>AS<string variable>	
Allocates random buffer space.	FIELD#1,3 AS A$
GET#<file number>[,<record number>]	
Reads random record.	GET#1,3
INPUT#<file number>,<variable list>	
Reads from sequential file.	

* Denotes a command not implemented in the BASIC Compiler.

KILL"<filename>"

Deletes a disk file. — KILL "A:GAME.BAS"

LINE INPUT#<file number>,<string variable>

Reads an entire line from a sequential file. — LINE INPUT #1,A$

LSET<string variable> = <string expression>

Stores data in random file buffer, left justified. — LSET A$="HELLO"

OPEN<mode>,[#]<file number>,<"filename">

Opens a disk file, where <mode>:
I = sequential input
O = sequential output
R = random I/O — OPEN"O",1,"GM.DAT"

PRINT#<file number>,<list of expressions>

Writes data to a sequential disk file. — PRINT #1,A$B

PUT#<file number>[,<record number>]

Writes data from a random buffer to a data file. — PUT#1,4

RSET<string variable> = <string expression>

Stores data in a random file buffer, right justified. — RSET B$="BYE"

WRITE#<file number>,<list of expressions>

Writes data to a sequential disk file. Delimiters are inserted between items in the I/O list. — WRITE #2,A,B$

DISK INPUT/OUTPUT FUNCTIONS

Function	Operation	Example
CVD(X$)	Converts 8-character string to double precision number.	A#=CVD(A$)
CVI(X$)	Converts 2-character string to an integer.	I%=CVI(I$)
CVS(X$)	Converts 4-character string to single precision number.	B=CVS(B$)
EOF(file no)	Returns true (−1) if a file is positioned at its end.	IF EOF(1)
LOC(file no)	Returns next record number to read (random file). Returns number of sectors accessed (sequential file).	X=LOC(1)
MKD$(Z#)	Converts double precision number to an 8-character string.	A$=MKD$(A#)
MKI$(I%)	Converts an integer to a 2-character string.	I$=MKI$(I%)
MKS$(B)	Converts a single-precision number to a 4-character string.	B$=MKS$(B)