KAYPRO

Microsoft™ BASIC

This book ia a quick reference guide to the Microsoft™ BASIC language, including the BASIC-80 Interpreter and the BASIC-86 Interpreter.

SPECIAL CHARACTERS (↑ means control)

↑A	Enters Edit Mode on line being typed or last line typed
↑C	Interrupts program execution, returns to BASIC command level and types OK
↑G	Rings the bell at the terminal
↑H	Deletes last character typed
↑I	Tab. Tab stops are every 8 columns
↑O	Halts/resumes program output
↑R	Retypes the line currently being typed
↑S	Suspends program execution
↑Q	Resumes execution after control-S
↑U	Deletes line being typed
↑X	Deletes line being typed
< return >	Ends every line typed in
< linefeed >	Used to break a logical line into physical lines
< rubout >	Deletes last character typed
< escape >	Escapes Edit Mode subcommands
.	Current line for EDIT, RENUM, DELETE, LIST, LLIST commands
&O or &	Prefix for octal constant
&H	Prefix for hexadecimal constant
:	Separates statements typed on the same line
?	Equivalent to PRINT statement (L? is not equivalent to LPRINT)

VARIABLE TYPE DECLARATION CHARACTERS

			Storage Bytes Used
$	String	(0 to 255 characters)	3+# of characters
%	Integer	(−32768 to 32767)	2
!	Single precision	(7.1 digit floating point)	4
#	Double precision	(16.8 digit floating point)	8

Syntax Conventions used in this book

In the syntax of statements, functions and commands, lower case items are to be supplied by the user. "filename" means a string expression that follows the naming convention of the operating system. "string" means a string expression and "exp" means a numeric expression. "line" and "line number" both mean line number. "f" is a file number or expression that evaluates to a file number. "n" means an integer. An item in square brackets is optional. Ellipsis (. . .) indicates an item may be repeated.

COMMANDS

NOTE: The FILES, RESET, and SYSTEM commands are in CP/M BASIC-80 only. The CP/M operating system appends the default extension .BAS to filenames used with LOAD, MERGE, RUN and SAVE.

Command	Syntax/Function	Example
AUTO	AUTO [line] [,inc] *Generate line numbers automatically.*	AUTO 100,50
CLEAR	CLEAR [,[exp1] [,exp2]] *Clear program variables. Exp1 sets end of memory and exp2 sets amount of stack space.*	CLEAR ,32768 CLEAR ,,2000
CONT	CONT *Continue program execution.*	CONT
DELETE	DELETE start line [-end line] *Delete program lines.*	DELETE 200 DELETE 20-25
EDIT	EDIT line number *Edit a program line. See Edit Mode Subcommands.*	EDIT 110

Command	Syntax/Function	Example
FILES	FILES [filename] *List files in disk directory that match filename. ? matches any character. * matches any name or extension.*	FILES FILES "*.BAS" FILES "TEST.BAS" FILES "B:*.*"
LIST	LIST [line[-[line]]] *List program lines at terminal.*	LIST 100-1000
LLIST	LLIST [line[-[line]]] *List program lines at printer.*	LLIST 50-
LOAD	LOAD filename [,R] *Load a program file.* *,R option means RUN.*	LOAD "INVEN"
MERGE	MERGE filename *Merge program on disk with program in memory. Program on disk must have been SAVEd in ASCII mode.*	MERGE "SUB1"
NAME	NAME old filename AS new filename *Change the name of a disk file.*	NAME "SUB1" AS "SUB2"
NEW	NEW *Delete current program and variables.*	NEW
NULL	NULL exp *Set the number of nulls printed after each line.*	NULL 2
RENUM	RENUM [[new line] [,[old line] [,inc]]] *Renumber program lines.*	RENUM 100,,100
RESET	RESET *Reinitialize CP/M disk information. Use after changing diskettes.*	RESET
RUN	RUN [line number] *Run a program (from line number).*	RUN RUN 50
	RUN filename [,R] *Load a program from disk and run it.* *,R used to keep files open.*	RUN "TEST"
SAVE	SAVE filename [,A or ,P] *Save the program in memory with name "filename." ,A saves program in ASCII. ,P protects file.*	SAVE "PROG",P

Command	Syntax/Function	Example
SYSTEM	SYSTEM *Close all files and return to CP/M. May also be used as a program statement.*	SYSTEM
TROFF	TROFF *Turn trace off.*	TROFF
TRON	TRON *Turn trace on.*	TRON
WIDTH	WIDTH [LPRINT] exp *Set terminal or printer carriage width. Default is 80 for terminal, 132 for printer.*	WIDTH 86 WIDTH LPRINT 100

EDIT MODE SUBCOMMANDS

Subcommand	Function
A	Restore original line and restart EDIT at the start of the line.
nCc	Change n character(s).
nD	Delete n character(s) at the current position.
E	End editing and save changes but don't type the rest of the line.
Hstring <escape>	Delete the rest of the line and insert string.
Istring <escape>	Insert string at current position.
nKc	Kill all characters up to the nth occurrence of c.
L	Print the rest of the line and go to the start of the line.
Q	Quit editing and restore original line.
nSc	Search for nth occurrence of c.
Xstring <escape>	Go to the end of the line and insert string.
<rubout>	Backspace over characters. In insert mode, delete characters.
<return>	End editing and save changes.
<space>	Move to next character

PROGRAM STATEMENTS (except I/O)

Statement	Syntax/Function	Example
CALL	CALL variable [(arg list)] *Call an assembly language or FORTRAN subroutine.*	CALL ROUT (I,J,K)

Statement	**Syntax/Function**	**Example**
CHAIN	CHAIN [MERGE] filename [,[line exp] [,ALL] [,DELETE range]] *Call a program and pass variables to it.* *MERGE with ASCII files allows overlays.* *If line exp is omitted, CHAINed program starts with the first line.* *,ALL means all variables will be passed, otherwise variables designated with COMMON.* *DELETE allows deletion of an overlay before CHAIN is executed.*	CHAIN "PROG1",1000 CHAIN MERGE"OVRLY2",1200
COMMON	COMMON list of variables *Pass variables to a CHAINed program.*	COMMON A,B( ),C$
DEF	DEF FNx[(arg list)]=exp *Define an arithmetic or string function.*	DEF FNA (X,Y)= SQR(X*X+Y*Y)
	DEF USRn=address *Define the entry address for the nth assembly language subroutine.*	DEF USR3=&2000
	DEFtype range(s) of letters *Define default variable types where "type" is INT, SNG, DBL, or STR.*	DEFINT I-N DEFSTR A,W-Z DEFDBL D
DIM	DIM list of subscripted variables *Allocate space for arrays and specify maximum subscript values.*	DIM A(3),B$(10,2,3)
END	END *Stop program, close all files and return to BASIC command level.*	END
ERASE	ERASE variable [,variable...] *Release space and variable names previously reserved for arrays.*	ERASE A,B$

Statement	Syntax/Function	Example
ERROR	ERROR code *Generate error of code (see table). May call user ON ERROR routine or force BASIC to handle error.*	ERROR 17
FOR	FOR variable=exp TO exp [STEP exp] *Used with NEXT statement to repeat a sequence of program lines. The variable is incremented by the value of STEP.*	FOR DAY=1 TO 5 STEP 2
GOSUB	GOSUB line number *Call a BASIC subroutine by branching to the specified line number. See RETURN.*	GOSUB 210
GOTO	GOTO line number *Branch to specified line number.*	GOTO 90
IF/THEN	IF exp THEN statement [:statement...] [ELSE statement...] *If exp is not zero, the THEN clause is executed. Otherwise, the ELSE clause or next statement is executed.*	IF X <Y THEN Y=X ELSE Y=A
IF/GOTO	IF exp GOTO line [ELSE statement...] *If exp is not zero, the GOTO clause is executed. Otherwise the ELSE clause or next statement is executed.*	IF ENDVAL > 0 GOTO 200
LET	[LET] variable=exp *Assign a value to a variable.*	LET X=I+5
MID$	MID$(string1,n[,m]) =string2 *Replace a portion of string1 with string2. Start at position n and replace m characters.*	MID$ (A$,14)="KS"
NEXT	NEXT variable [,variable...] *Delimits the end of a FOR loop.*	NEXT I

Statement	Syntax/Function	Example
ON ERROR GOTO	ON ERROR GOTO line *Enables error trap subroutine beginning at specified line. If line=0 , disables error trapping. If line=0 inside error trap routine, forces BASIC to handle error.*	ON ERROR GOTO 1000
ON/GOSUB	ON exp GOSUB line [,line] *GOSUB to statement specified by expression. (If exp=1, to 20; If exp=2, to 20; If exp=3, to 40; otherwise, error.)*	ON DATE%+1 GOSUB 20,20,40
ON/GOTO	ON exp GOTO line [,line...] *Branch to statement specified by exp. (If exp=1, to 20; If exp=2, to 30; otherwise, error.)*	ON INDEX GOTO 20,30
OPTION BASE	OPTION BASE n *Declare the minimum value for array subscripts. n is 0 or 1.*	OPTION BASE 1
OUT	OUT port,byte *Puts byte specified to output port specified.*	OUT 41,16+DATAO%
POKE	POKE address,byte *Puts byte specified into memory location specified.*	POKE &23100,255
RANDOMIZE	RANDOMIZE [exp] *Reseed the random number generator.*	RANDOMIZE 5
REM	REM any text *Allows user to insert comments in program (not executed). NOTE: ":" does not terminate a REM statement.*	REM COMPUTE AVERAGE
RESTORE	RESTORE [line number] *Resets DATA pointer so that DATA statements may be re-read.*	RESTORE
RESUME	RESUME or RESUME 0 *Returns from ON ERROR routine to statement that caused error.*	RESUME

Statement	Syntax/Function	Example
	RESUME NEXT *Returns to statement after the one that caused the error.*	RESUME NEXT
	RESUME line *Returns to the specified line.*	RESUME 100
RETURN	RETURN *Return from subroutine to statement following last GOSUB executed.*	RETURN
STOP	STOP *Stop program execution, print BREAK message, and return to command level.*	STOP
SWAP	SWAP variable,variable *Exchanges values of two variables.*	SWAP A$,B$
WAIT	WAIT port,mask [,select] *Suspends program execution, reads input at port until (input bit [XOR select] AND mask) returns non-zero, then continues execution with the next statement.*	WAIT 21,1
WHILE/ WEND	WHILE exp...WEND *Execute the statements in the WHILE/WEND loop as long as exp is true.*	WHILE AMT >0 . . WEND

PRINT USING Format Field Specifiers

NUMERIC

Specifier	Possible Digits	Field Characters	Definition	Example
#	1	1	Numeric field	####
.	0	1	Decimal point	#.#
+	0	1	Print leading or trailing sign. Positive numbers will have "+", negative numbers will have "–"	+## ###+

Specifier	Possible Digits	Field Characters	Definition	Example
–	0	1	Trailing sign. Prints "–" if negative, otherwise blank.	##.##–
**	2	2	Leading asterisk	**###.##
$$	1	2	Floating dollar sign. $ is placed in front of the leading digit.	$$##.##
**$	2	3	Asterisk fill and floating dollar sign	**$#.##
	1	1	Use comma every three digits (left of decimal point only.)	##,###.##
↑↑↑↑	0	4	Exponential format. Number is aligned so leading digit is non-zero.	#.##↑↑↑↑
underscore	0	1	Next character literal	__!#.#

STRING

!	Single character	!
\<spaces>\	2+ number of spaces character field	\ \
&	Variable length field	&

INPUT/OUTPUT STATEMENTS

Statement	Syntax/Function	Example
CLOSE	CLOSE [[#] f [,[#] f...]] *Closes disk files. If no argument, all open files are closed.*	CLOSE 6
DATA	DATA list of constants *Lists data to be used in a READ statement.*	DATA 2.3,"PLUS",4

Statement	Syntax/Function	Example
FIELD	FIELD [#] f,n AS string variable [,n AS string variable ...] *Define fields in a random file buffer.* *Note: PRINT#[USING] and [LINE] INPUT# statements to random files write and read data into the FIELD buffer.*	FIELD #1,3 AS A$,7 AS B$
GET	GET [#] f [,record number] *Read a record from a random disk file.*	GET #1,17*I+1
INPUT	INPUT [;] [prompt string;] variable [,variable...] INPUT [;] [prompt string,] variable [,variable...] *Read data from the terminal. Semicolon after INPUT suppresses echo of carriage return/line feed. Semicolon after prompt string causes question mark after prompt. Comma after prompt string suppresses question mark.*	INPUT "VALUES";A,B
	INPUT #f, variable [,variable...] *Read data from a disk file.*	INPUT #1,A,B
KILL	KILL filename *Delete a disk file.*	KILL "INVEN.BAS"
LINE INPUT	LINE INPUT [;] [prompt string;] string variable *Read an entire line from the terminal. Semicolon after LINE INPUT suppresses echo of carriage return/line feed.*	LINE INPUT A$ LINE INPUT "NAME";N$
	LINE INPUT #f,string variable *Read an entire line from a disk file.*	LINE INPUT #2,B$
LSET	LSET field variable=string exp *Store data in random file buffer left justified. Or left justify a non-disk string in a given field.*	LSET A$="JOHN JONES" LSET B$=MKS$(MAX)
OPEN	OPEN mode,[#] f,filename[,reclen] *Open a disk file. Mode must be one of:* *I (sequential input file)* *O (sequential output file)* *R (random input/output file)*	OPEN "O",#1,"OUTPUT"
PRINT	PRINT [USING format string;] exp [,exp...] *Print data at the terminal using the format specified. See table for format characters.*	PRINT USING "!";A$,B$

Statement	Syntax/Function	Example
	PRINT #f, [USING format string;] exp [,exp...] *Write data to a disk file.*	PRINT #4,A,B
	LPRINT [USING format string;] variable [,variable] *Write data to a line printer.*	LPRINT A,B
PUT	PUT [#] f [,record number] *Write data from a random buffer to a data file.*	PUT #3,4
READ	READ variable [,variable...] *Read data from a DATA statement into the specified variables.*	READ I,X,A$
RSET	RSET field variable=string exp *Store data in a random file buffer right justified. Or right justify a non-disk string in a given field.*	RSET B$="CORRECT" RSET C$=MKS$(COUNT)
WRITE	WRITE [list of exps] *Output data at the terminal.*	WRITE A,B,C$
	WRITE #f,list of exps *Write data to a sequential file or a random field buffer.*	WRITE #1,A$,B$

OPERATORS

Symbol	Function
=	Assignment or equality test
–	Negation or subtraction
+	Addition or string concatenation
*	Multiplication
/	Division (floating point result)
↑	Exponentiation
\	Integer division (integer result)
MOD	Integer modulus (integer result)
NOT	One's complement (integer)
AND	Bitwise AND (integer)
OR	Bitwise OR (integer)

Symbol	Function
XOR	Bitwise exclusive OR (integer)
EQV	Bitwise equivalence (integer)
IMP	Bitwise implication (integer)
=,< ,>, < =,=< , > =,=> < >	Relational tests (result is TRUE = −1 or FALSE = 0)

The precedence of operators is:

(1) Expressions in parentheses
(2) Exponentiation (A ↑ B)
(3) Negation (−X)
(4) *,/
(5) \
(6) MOD
(7) +,−
(8) Relational operators (=,< >,<,>,< =, >=)
(9) NOT
(10) AND
(11) OR
(12) XOR
(13) IMP
(14) EQV

ARITHMETIC FUNCTIONS

Function	Action	Example
ABS(exp)	Absolute value of expression	Y=ABS(A+B)
ATN(exp)	Arctangent of the expression (in radians)	PRINT ATN(A)
CDBL(exp)	Convert the expression to a double precision number	A=CDBL(Y)
CINT(exp)	Convert the expression to an integer	B=CINT(B)
COS(exp)	Cosine of the expression (in radians)	A=COS(2.3)
CSNG(exp)	Convert the expression to a single precision number	C=CSNG(X)
EXP(exp)	Raises the constant e to the power of expression	B=EXP(C)

Function	Action	Example
FIX(exp)	Returns truncated integer of expression	J=FIX(A/B)
FRE(exp)	Gives memory free space not used by BASIC	PRINT FRE(0)
INT(exp)	Evaluates the expression for the largest integer contained	C=INT(X+3)
LOG(exp)	Gives the natural logarithm of the expression	D=LOG(Y−2)
RND[(exp)]	Generates a random number. Expression: <0 seed new sequence; =0 return previous random number; >0 or omitted, return new random number	E=RND(1)
SGN(exp)	1 if expression > 0; 0 if expression = 0; −1 if expression < 0	B=SGN(X+Y)
SIN(exp)	Sine of the expression (in radians)	B=SIN(A)
SQR(exp)	Square root of expression	C=SQR(D)
TAN(exp)	Tangent of the expression (in radians)	D=TAN(3.14)

STRING FUNCTIONS

Function	Action	Example
ASC(string)	Returns the ASCII value of the first character of a string	PRINT ASC(A$)
CHR$(exp)	Returns a one-character string whose character has the ASCII code of exp	PRINT CHR$(48)
FRE(string)	Returns remaining memory free space	PRINT FRE(A$)
HEX$(exp)	Converts a number to a hexadecimal string	H$=HEX$(100)
INKEY$	Returns either a one-character string read from terminal or null string if no character pending at terminal.	A$=INKEY$
INPUT$(length [,[#] f])	Returns a string of length characters read from console or from a disk file. Characters are not echoed.	X$=INPUT$(4) X$=INPUT X$=INPUT$(5,#2)

Function	Action	Example

INSTR([exp,]string 1,string2)	Returns the first position of the first occurrence of string2 in string1 starting at position exp	INSTR(A$,":")
		INSTR(3,X$,Y$)

LEFT$(string,length)	Returns leftmost length characters of the string expression	B$=LEFT$(X$,8)
LEN(string)	Returns the length of a string	PRINT LEN(B$)
MID$(string,start [,length])	Returns characters from the middle of the string starting at the position specified to the end of the string or for length characters	A$=MID$(X$,5,10)
OCT$(exp)	Converts a number to an octal string	O$=OCT$(100)
RIGHT$(string,length)	Returns rightmost length characters of the string expression	C$=RIGHT$(X$,8)
SPACE$(exp)	Returns a string of exp spaces	S$=SPACE$(20)
STR$(exp)	Converts a numeric expression to a string	PRINT STR$(35)
STRING$(length,string)	Returns a string length long containing first character of string	X$=STRING$(100,"A")
STRING$(length,exp)	Returns a string length long containing characters with numeric value exp	Y$=STRING$(100,42)
VAL(string)	Converts the string representation of a number to its numeric value	PRINT VAL("3.1")

I/O AND SPECIAL FUNCTIONS

Function	Action	Example

CVI(string)
CVS(string)
CVD(string)
	Converts a 2-character string to an integer (CVI). Converts a 4-character string to a single precision number (CVS). Converts an 8-character string to a double precision number (CVD).	Y!=CVS(N$)
		A%=CVI(B$)
		C#=CVD(X$)

Function	Action	Example
EOF(f)	Returns true (−1) if file is positioned at its end	IF EOF(1) GOTO 300
ERL	Error line number	PRINT ERL
ERR	Error code number	IF ERR=62 THEN...
INP(port)	Inputs a byte from an input port	PRINT INP(21)
LOC(f)	Returns next record number to read or write (random file), or number of sectors read or written (sequential file)	PRINT LOC(1)
LPOS(n)	Returns carriage position of line printer (n is dummy argument)	IF LPOS(3) > 60...
MKI$(value) MKS$(value) MKD$(value)	Converts an integer to a 2-character string (MKI$). Converts a single precision value to a 4-character string (MKS$). Converts a double precision value to an 8-character string (MKD$).	LSET D$=MKS$(A) LSET A$=MKI$(B%)
PEEK(exp)	Reads a byte from memory location specified by expression	PRINT PEEK(&2000)
POS(n)	Returns carriage position of terminal (n is dummy argument)	IF POS(3) > 60...
SPC(exp)	Used in PRINT statements to print spaces	PRINT SPC(5),A$
TAB(exp)	Used in PRINT statements to tab carriage to specified position	PRINT TAB(20),A$
USR[n](arg)	Calls the user's machine language subroutine with the specified argument. See DEF USR.	X=USR2(Y)
VARPTR(var)	Returns address of variable in memory or zero if variable has not been assigned a value	I=VARPTR(X)
VARPTR(#f)	For sequential files, returns the address of the disk I/O buffer assigned to file number. For random files, returns the address of the FIELD buffer.	J=VARPTR(#2)

TABLE OF ERROR CODES

Code	Error	Code	Error
1	NEXT without FOR	14	Out of string space
2	Syntax error	15	String too long
3	RETURN without GOSUB	16	String formula too complex
4	Out of data	17	Can't continue
5	Illegal function call	18	Undefined user function
6	Overflow	19	No RESUME
7	Out of memory	20	RESUME without error
8	Undefined line	21	Unprintable error
9	Subscript out of range	22	Missing operand
10	Redimensioned array	23	Line buffer overflow
11	Division by zero	26	FOR without NEXT
12	Illegal direct	29	WHILE without WEND
13	Type mismatch	30	WEND without WHILE

Disk Errors

Code	Error	Code	Error
50	Field overflow	58	File already exists
51	Internal error	61	Disk full
52	Bad file number	62	Input past end
53	File not found	63	Bad record number
54	Bad file mode	64	Bad file name
55	File already open	66	Direct statement in file
57	Disk I/O error	67	Too many files