BASIC CONVERTER CHART

One day, all computers will understand the same language (and read each others' disks and address the screen in the same way and . . .). To tide you through until this great day arrives, however, we set out to beg, steal or even buy eleven of the most popular home micros to produce this APC Basic Converter Chart.

Whether you're trying to convert that amazing Atari game to run on your Apple, have just spent the past three hours wondering why your new Commodore 64 micro doesn't seem to give the right answer to a FRE statement or simply want to write programs which can be easily converted to other micros, the APC Basic Converter Chart is here to help.

It isn't possible, of course, to cover every micro nor every command supported by each of the machines included — much as we'd like to. Also, since different micros have an annoying tendency to use the same keyword to perform slightly — or totally — different functions, converting from one machine to another will require some rewiting beyond simply changing the syntax. What this chart aims to do, however, is provide you with an at-a-glance syntax comparison using Microsoft Basic as the standard. The chart won't convert programs for you, but it should save you the trouble of wading through masses of manuals written by authors who have apparently not yet heard about alphabetical indexing.

Due to the limited amount of information we can squeeze into each box, it hasn't always been possible to indicate the full power of every command or statement. Most LIST statements, for example, allow you to list the whole program, list a specified line, list all lines within a given range, list all lines up to a specified line or list from a specified line. Fiddling around with brackets in an attempt to represent each of these possibilities would lead to a totally incomprehensible entry. It should be assumed, therefore, that we're dealing with the most common use of each statement here and that other uses may be available.

Something to be aware of is that identical syntax may have very different effects on different machines. SYSTEM on a TRS-80 will transfer program control to a machine language routine while in Microsoft Basic closes files prior to returning to the operating system.

You will notice that we haven't included anything on sound and graphics; with most of today's micros offering both high-resolution graphics and fairly sophisticated sound control, this area would require a chart of its own. APC will be looking at sound and colour in a later issue.

The abbreviations used in the chart are as follows:

addr = address, exp = expression,
sub = subscript, stmt = statement,
var = variable,
Square bracket [ ] indicates optional code.

1 of 7.

Nov 83 4(11) 89-95

From: Australian Personal Computer November 1983 Vol. 4 No. 11
Scanned by Bob Kitch February 2021 at 400 dpi
Composited in MicroSoft Publisher 2016
Size 600 x 820 mm

BASIC CONVERTER CHART '86

Australian Personal Computer
WALLCHART

Those rotten manufacturers still insist on making machines that won't talk to each other in the same language. Some enlightened people are having a go with MSX, but in the meantime and in response to overwhelming demand, here's the 1986 APC Converter Chart.

We've added seven new Basics, covering the latest machines, and revised and updated the chart. It isn't possible, of course, to cover every micro nor every command supported by each of the machines included. What this chart aims to do is to provide an at-a-glance syntax comparison using Microsoft Basic as a reference point. The chart won't convert programs for you but it will save you the trouble of getting hold of piles of manuals — and even when you've got them it's often the beginning, not the end of your worries.

To use the chart, first check that the keyword you want isn't in the box on the right. If it is, then you're lucky: it's one of the few that IS the same on every single machine featured here. Due to the limited amount of information we can squeeze into each box, it hasn't always been possible to indicate the full power of every statement. It should be assumed, therefore, that we're dealing with the most common uses of each statement, and that other uses may be available.

Something to watch out for: identical syntax may have different effects on different machines. Watch out especially for SYSTEM and RND.

You'll notice we haven't included anything on sound and graphics: that's too complicated for a quick reference chart, but we've covered the subject in a series of articles which will appear in APC for a range of machines.

SHARED INSTRUCTIONS

ABS	(exp)	
COS	(exp)	
END		NB not available on QL
FOR	var=exp	TO exp [STEP exp]
LEN	(string)	NB Space *must* be present for Memotech
LET	var=EXP	NB LET obligatory after THEN and ELSE on MicroBee
REM	text	
SIN	(exp)	
SQR	(exp)	
STOP		
TAN	(exp)	
VAL	(exp)	NB not available on QL

ABBREVIATIONS USED IN THIS CHART:

addr	= address
exp	= expression
parm(s)	= parameter(s)
stmt	= statement
var	= variable

Square brackets [] indicate optional code.

BASIC RESERVED WORDS & FORMATS

MACHINE	ASC Returns ASCII value of first character of string.	ATN Arctangent of expression.	AUTO	CALL Calls assembler language subroutine	CHAIN Call a new program & pass variables to it.	CHR$ Gives one-char string with ASCII code of exp.	CLEAR CLEAR all [or selected] variables.	CLOSE Closes disk files — closes all files if no specification.	CONT continue program execution	DATA Lists data to be used in a READ statement	DEF Define arithmetic string function.	DELETE Delete specified program lines.	DIM Allocates space for arrays, specifies max subscript values.	EDIT Edit a program line.	EXP Raises to power of expression.	FRE Returns remaining memory space.	GET Read a record from disk or tape file.	GOSUB Branch to a Basic subroutine.	GOTO Branch to a specified line number.	IF/THEN/ELSE If exp is true stmt is executed. If not ELSE or following line is executed.
STANDARD MICROSOFT	ASC (string)	ATN (exp)	AUTO [lineno, val]	CALL var[,var, var . . .]	CHAIN "filename"	CHR$ (exp)	CLEAR [exp,exp]	CLOSE	CONT	DATA const [,const, . . .]	DEF FNvar [(var,var . . .)] =exp	DELETE lineno [,lineno]	DIM var(sub) [,var (sub), . . .]	EDIT lineno	EXP(exp)	FRE(exp)	GET [#] lineno [,record no] or INPUT#fileno, var [,var . . .] for sequential files	GOSUB lineno	GOTO lineno	If exp THEN [ELSE stmt]
AMSTRAD 464/664/6128	ASC (string)	ATN (exp)	AUTO [lineno, incl]	CALL addr [,parms]	CHAIN "filename" [,lineno,exp]	CHR$ (exp)	CLEAR [all] ERASE [list of] var NB: clears and removes arrays	CLOSEIN [NB cassette input file] CLOSEOUT [NB cassette output file]	CONT	DATA const [,const, . . .]	DEF FN(var) [(var,var, . . .)]=exp	DELETE [line no-line no]	DIM [list of] var (dimension list)	EDIT lineno	EXP(exp)	FRE(exp) Note: exp is a dummy variable	LINE INPUT#, [;string][var, . . .]	GOSUBlineno	GOTO lineno	If exp THEN stmt [ELSE stmt]
APPLE II	ASC (string)	ATN (exp)		CALL addr [var,var . . .]	CHAIN "filename"	CHR$ (exp)	CLEAR	CLOSE [filename]	CONT	DATA const [,const, . . .]	DEF FNvar (var)=exp	DEL lineno, lineno	DIM var(sub) [,var(sub) . . .]	[screen editing using ESC key]	EXP(exp)	FRE(exp) Note: exp is a dummy variable	INPUT var [,var . . .] NB: Get var(s) from current input device	GOSUB lineno	GOTO lineno	IF exp THEN stmt Note: no ELSE
ATARI	ASC (string)	ATN (exp)		USR (addr [,var,var . . .])	RUN "C:" NB: program must have been saved using SAVE "C:"	CHR$ (exp)	CLR	CLOSE [#fileno, var,var,filename]	CONT	DATA const [,const, . . .]			DIM var (sub) [,var (sub) . . .] NB: dimension ALL strings	[cursor editing]	EXP(exp)	FRE(exp) Note: exp is a dummy variable	GET #lineno, record	GOSUB lineno/ var/exp	GOTO lineno/ var/exp	If exp THEN stmt Note: no ELSE
BBC	ASC (string)	ATN (exp)	AUTO [lineno, val]	CALL addr, [var][,var, . . .]	CHAIN "filename"	CHR$ (exp)	CLEAR	CLOSE #fileno Note: CLOSE #0 to close all files	NB not available: use GOTO lineno	DATA const [,const, . . .]	DEF FNvar =exp	DELETE lineno [-lineno]	DIM var(sub) [,var(sub) . . .]	[cursor editing]	EXP(exp)	HIHEM-TOP Use PRINT	INPUT #lineno, record [,record . . .]	GOSUB lineno/ [var][exp]	GOTO lineno/ [var][exp]	If exp THEN stmt [ELSE stmt]
COMMODORE 64 & VIC 20	ASC (string)	ATN (exp)		SYS addr		CHR$ (exp)	CLR	CLOSE #fileno	CONT	DATA const [,const, . . .]	DEF FNvar =exp		DIM var(sub) [(sub) . . .]	[cursor editing]	EXP(exp)	FRE(exp) Note: exp is a dummy variable	GET #fileno, var	GOSUB lineno	GOTO lineno	IF exp THEN stmt Note: no ELSE
IBM PC-BASIC A	ASC (string)	ATN (exp)	AUTO [lineno] [,inc]	CALL addr [var, . . .,var]	CHAIN filename	CHR$ (exp)	CLEAR	CLOSE [#] [filename]	CONT	DATA const [,const, . . .]	DEF FNvar [(parms)]=exp	DELETE [lineno] [-lineno]	DIM var(sub) [,var(sub) . . .]	EDIT lineno	EXP(exp)	FRE(exp) Note: exp is a dummy variable	GET [#] filename [,rec no]	GOSUB lineno	GOTO lineno	If exp THEN stmt [ELSE stmt]
MEMOTECH MTX 512	ASC (string)	ATN (exp)	AUTO [lineno] [,inc]	USR (addr)		CHR$ (exp)	CLEAR	DISC CLOSE # channel no	CONT	DATA const [,const, . . .]	DEF FNvar [(parms)]=exp		DIM var(sub) [,var(sub)] . . .	EDIT lineno	EXP(exp)		Disc INPUT # channelno	GOSUB lineno	GOTO lineno	If exp THEN stmt [ELSE stmt]
MICROBEE	ASC (string)	ATAN (real-exp)	AUTO [lineno, val]			CHR (integer-exp)	STRS (int-exp) Note: set limits for string memory		CONT	DATA expr (,exp ("exp"))	FNn=exp	DELETE lineno. lineno.)	DIM var(sub) [,var(sub) . . .]	EDIT (lineno.)	EXP(exp)	FRE(0) mem. space FRE($) str. space		GOSUB NB: sq. br. significant	GOTO lineno	If exp THEN stmt [ELSE stmt]
MSX BASIC	ASC (string)	ATN (exp)	AUTO [lineno ,inc]	USR (addr)		CHR$ (exp)	CLEAR [var]	DISK basic only	CONT	DATA const [,const, . . .]	DEF FN(var) [(parms)]=exp	DELETE [lineno] -lineno]	DIM var(sub) [,var(sub) . . .]	[cursor editing]	EXP(exp)	FRE(exp) Note: exp is a dummy variable		GOSUB lineno	GOTO lineno	IF exp THEN stmt [ELSE stmt]
TANDY 100	ASC (string)	ATN (exp)		CALLadr [,param,param]		CHR$ (exp)	CLEAR [(exp)] — Clears string space	CLOSE [fileno] if exp is given	CONT	DATA const [,const . . .]			DIM var(sub) [,var(sub) . . .]	EDIT lineno [-lineno]	EXP (exp)	FRE (exp)	INPUT #fileno, var [,var . . .]	GOSUB lineno	GOTO lineno	If exp THEN stmt [ELSE stmt]
TANDY COLOR	ASC (string)	ATN (exp)		EXEC addr		CHR$ (exp)	CLEAR [(exp)] clears string space if exp is given	CLOSE #-fileno	CONT	DATA const [,const, . . .]	DEF FNvar (var)=exp	DELETE lineno-neno	DIM var(sub) [,var(sub) . . .]	EDIT lineno	EXP(exp)	MEM	INPUT #-fileno, record	GOSUB lineno	GOTO lineno	If exp THEN stmt [ELSE stmt]
SINCLAIR QL	CODE (str)	ATAN (exp)	AUTO [lineno] [,inc]	CALL addr [,parms]	M RUN "filename"	CHR$ (exp)	CLEAR	CLOSE # channel	CONTINUE	DATA const [,const, . . .]	DEF FNvar [$/%]=exp: END DEF	DLINE lineno [TO]lineno]	DIM var(sub) [,var(sub) . . .]	EDIT lineno [,step]	EXP(exp)		INKEY$ (#channel)	GOSUB lineno/ var/exp	GOTO lineno/ var/exp	IF exp THEN stmt [ELSE stmt][END IF]
TRS-80 II/SYSTEM 80	ASC (string)	ATN (exp)	AUTO [lineno, val]			CHR$ (exp)	CLEAR [(exp)] Note: Clears string space if exp given	[depends on OS; consult OS manual]	CONT	DATA const [,const, . . .]	Various DEF statements available but none equivalent	DELETE lineno -lineno]	DIM var(sub)	EDIT lineno	EXP(exp)	FRE(exp) [TRS-80] or MEM [System 80]	INPUT #-fileno, record [,record . . .]	GOSUB lineno	GOTO lineno	If exp THEN stmt [ELSE stmt]
VA-200	ASC (string)	ATN (exp)				CHR$ (exp)	CLEAR [exp] N clears string space		CONT	DATA const [,const, . . .]			DIM var(sub) [,var(sub) . . .]		EXP(exp)		INPUT # file-name var[,var . . .] NB: Gets record from tape	GOSUB lineno	GOTO lineno	IF exp THEN stmt [ELSE stmt]
ZX SPECTRUM	CODE (string)	ATN (exp)		LET var=USR addr		CHR$ (exp)	CLEAR [var]	CLOSE # channel no	CONT	DATA const [,const, . . .]	DEF FN(var) [(var,var . . .)]		DIM var(sub)	EDIT [lineno] Note: cursor line by default	EXP(exp)		Consult Microdrive manual	GOSUB lineno [exp]	GOTO lineno [exp]	IF exp THEN stmt Note: no ELSE

MACHINE	INKEY$ Returns character typed at keyboard or null if no character used	INPUT Read data from terminal	INT Evaluates expression for largest integer contained.	LEFT$ Returns specified no. of characters starting at beginning of string.	LIST List specified program lines at terminal.	LLIST List specified program lines at printer.	LOAD Load a program file into memory.	LOG Natural logarithm of expression.	MID$ Gives specified no. of characters to the right of start position in string.	NAME Rename a file.	NEW Delete current program & data from memory.	NEXT End of FOR/NEXT loop.	ON ERROR Error trap subroutine.	ON/GOSUB GOTO lineno specified by evaluation of expression.	ON/GOTO GOTO lineno specified by evaluation of expression.	OPEN Open disk file.	OUT Put specified byte to specified output port.	PEEK Read byte from specified memory location.	POKE Put specified byte to specified memory address.	PRINT Write data to disk tape or terminal.
STANDARD MICROSOFT	INKEY$	INPUT [string:] var[,var . . .]	INT (exp)	LEFT$ (string, length)	LIST [lineno, lineno]	LLIST [lineno, lineno]	LOAD ["filename"]	LOG(exp)	MID$(string,start [,length])	NAME "filename" AS "filename"	NEW	NEXT var [,var . . .]	ON ERROR GOTO lineno	On exp GOSUB lineno [,lineno . . .]	On exp GOTO lineno [,lineno . . .]	OPEN mode [#] fileno "filename"	OUT port, byte	PEEK(addr)	POKE addr,byte	PRINT [[#] fileno] [exp] [,exp . . .]
AMSTRAD 464/664/6128	INKEY$	INPUT [#no] [prompt] [var [list]]	INT (exp)	LEFT$ (string, length)	LIST [lineno, lineno]	LIST [lineno, lineno] #8	LOAD ["filename"] [,addr]	LOG(exp) Note: LOG10(exp) gives Log base 10	MID$(string, start, length)		NEW	NEXT [var][,var, . . .]	ON ERROR GOTO lineno	On exp GOSUB lineno [,lineno, . . .]	ON exp GOTO lineno [,lineno, . . .]	OPEN mode [#] fileno "filename"	OUT port, byte	PEEK(addr)	POKE addr,byte	PRINT [#]fileno] [exp][,exp . . .]
APPLE II	Get var	INPUT [string,] var [,var . . .]	INT (exp)	LEFT$ (string, length)	LIST [lineno, lineno] Note: '—' may be used in place of ','	PR # slotno: LIST [lineno,lineno]: PR # 0	LOAD filename	LOG(exp)	MID$(string, start[,length])	RENAME oldname, newname	NEW	NEXT [var,var . . .]	ONERR GOTO lineno	On exp GOSUB lineno [,lineno . . .]	ON exp GOTO lineno [,lineno . . .]	OPEN filename, parm		PEEK(addr)	POKE addr,byte	PRINT exp [,exp . . .] NB: prints to current output device
ATARI		INPUT [string,] var [,var . . .]	INT (exp)	string (start, length)	LIST [lineno, lineno]	LIST [lineno, lineno]	CLOAD ["filename"] [cases] or LOAD "fileno-filename" [disk]	LOG(exp)	string(start [,length])		NEW	NEXT var	TRAP lineno/ var/exp	ON exp GOSUB lineno/var/exp [,lineno/var/exp . . .]	ON exp GOTO lineno/var/exp [,lineno/var/exp . . .]	OPEN #fileno, mode control code, filename	[not equivalent]	PEEK(addr)	POKE addr,byte	PRINT #device, exp[,exp]
BBC	Get var [unlimited] time] or INKEY$ (time) Note: 100ths sec.	INPUT [string;] var [,var . . .]	INT (exp)	LEFT$ (string, length)	LIST [lineno, lineno]	CTRL-B then LIST [lineno-lineno]	LOAD "filename" Note '*DISK' or '*TAPE' to select device	LN(exp) NB: LOG(exp) gives common rather than natural log	MID$(string, start[,length])		NEW Note: under cert. circum. may be recovered using OLD	NEXT [var][,var . . .]	ON ERROR stmt [OFF]	ON exp[var] GOSUB lineno [,lineno . . .]	ON exp[var] GOTO lineno [,lineno . . .]	fileno=OPENIN [to read] or fileno= OPENOUT [to write]		?addr NB: '?' does NOT mean 'print' in BBC Basic	?addr,byte	PRINT #fileno record [,record . . .]
COMMODORE 64 & VIC 20	GET var	INPUT [string:] var [,var . . .]	INT (exp)	LEFT$ (string, length)	LIST [lineno-lineno]	OPEN 4,4:CMD4: LIST [lineno-lineno] OPEN 3,4:CMD3: LIST [lineno-lineno]	LOAD ["filename"] [cass] or LOAD "filename",8 [disk]	LOG(exp)	MID$(string, start[,length])	OPEN 1,8,15, "RO: filename= filename" [disk only] N/A	NEW	NEXT [var][,var . . .]		ON exp GOSUB lineno [,lineno . . .]	ON exp GOTO lineno [,lineno . . .]	OPEN #exp, fileno, mode, "filename"		PEEK(addr)	POKE addr,byte	PRINT #fileno record [,record . . .]
IBM PC-BASIC A	var$=INKEY$	INPUT [prompt] var [,var]	INT (exp)	LEFT$ (string, length)	LIST [1st line] [-last line] [,filespec]	LIST [lineno, lineno]	LOAD filename [,R]	LOG(exp)	MID$(string, start[,length])	NAME filename AS filename	NEW	NEXT [var,var, . . .]	ON ERROR GOTO lineno	ON [exp:COM: KEY:PEN:STRING] GOSUB lineno	ON exp GOTO lineno	OPEN filename [FOR Mode] AS [#] filename [LEN=rec]	OUT port,data	PEEK(addr)	POKE addr,byte	PRINT [exp][;]
MEMOTECH MTX 512	var$=INKEY$	INPUT [prompt] var [,var]	INT (exp)	LEFT$ (string, length)	LIST [1st line] [,last line]	LIST [lineno, lineno]	LOAD "filename"	LN(exp)	MID$(string, start[,length])	DISC REN string=string	NEW	NEXT var		ON exp GOSUB lineno	ON exp GOTO lineno	DISC OPEN # channel no, "filename", filetype, record length	OUT port,data	PEEK(addr)	POKE addr,byte	[DISC] PRINT [#channelno,] print list
MICROBEE	KEY	INPUT [string] var [,var]	INT (real-exp)	var(;1, length)	LIST [lineno [,lineno]] forceloads	LLIST (lineno. (,lineno))	LOAD (U) (?) ("filename") LOAD U	LOG(real-exp)	var(;n,n+m-1) -n-start character, m-length		NEW	NEXT var NEXT *var lineno. -exits loop before completion	ON ERROR GOTO lineno.	ON exp GOSUB ((exp(,exp))lineno (,((exp . . .	ON exp GOTO lineno [,lineno . . .]		OUT port,byte	PEEK(addr)	POKE addr, byte	PRINT list of arguments
MSX BASIC	var$=INKEY$	INPUT [prompt] var [,var]	INT (exp)	LEFT$ (string, length)	LIST [1st line] [-last line]	LLIST [lineno, lineno]	CLOAD filename	LOG(exp)	MID$(string, start[,length])		NEW	NEXT [var][,var, . . .]	ON ERROR GOTO lineno	ON exp GOSUB lineno [,lineno, . . .]	ON exp GOTO lineno [,lineno, . . .]	OPEN "device: filename" for OUTPUT [for INPUT] AS#var	OUT port,data	PEEK(addr)	POKE addr,byte	PRINT [#fileno,] var[,var, . . .]
TANDY 100	INKEY$	INPUT [string;] var [,var . . .]	INT (exp)	LEFT$ (string, length)	LIST [lineno-lineno]	CLOAD lineno]	LOG (exp) ["filename"]	MID$(string,	start[,length])	NEW		NEXT [var,var . . .]	ON ERROR GOTO lineno	On exp GOTO lineno [,lineno . . .]	OPEN "filename" lineno	OUT port,byte FOR (mode)	PEEK(addr)	POKE addr,byte	PRINT [#fileno]	[exp,exp . . .]
TANDY COLOR	INKEY$	INPUT [string;] var [,var]	INT (exp)	LEFT$ (string, length)	LIST [lineno-lineno]	LLIST [lineno-lineno]	CLOAD ["filename"]	LOG(exp)	MID$(string, start[,length])		NEW	NEXT [var][,var . . .]		ON exp GOSUB lineno [,lineno . . .]	ON exp GOTO lineno [,lineno . . .]	OPEN mode, #-fileno "filename"		PEEK(addr)	POKE addr,byte	PRINT #-fileno. exp[,exp . . .]
SINCLAIR QL	INKEY$ (#channel ,time)	INPUT [# channel,] [prompt] [var [,var] . . .]	INT (exp)	string (TO finish)	LIST [#channel] 1st line [To last line]	[LIST #[channel] [lineno] [To lineno]	LOAD device [inc. filename]	LN(exp) Note: LOG10(exp) gives common rather than natural log	string(start TO finish)		NEW	NEXT var[,var]/ END FOR var[,var]	WHEN ERROR var: END WHEN Note: OS JS	ON var GOSUB lineno [,lineno]	ON var GOTO lineno [,lineno]	OPEN #channel, "filename"		PEEK [or W or L](addr)	POKE [or W or L](addr),byte	PRINT [#channel,] exp[,exp] . . .
TRS-80 II/SYSTEM 80	INKEY$	INPUT [string;] var [,var . . .]	INT (exp)	LEFT$ (string, length)	LIST [lineno-lineno]	LLIST [lineno-lineno]	CLOAD "[filename]" [cass] or LOAD "filename" [disk/ floppy tape]	LOG(exp)	MID$(string, start,length)	[depends on OS; consult OS manual]	NEW	NEXT [var][,var . . .]	ON ERROR GOTO lineno	ON exp GOSUB lineno [,lineno . . .]	ON exp GOTO lineno [,lineno . . .]	[depends on OS; consult OS manual.]	OUT port,byte	PEEK(addr)	POKE addr,byte	PRINT #fileno, record [,record . . .] [cass]
VZ-200	INKEY$	INPUT [string;] var [,var . . .]	INT (exp)	LEFT$ (string, length)	LIST [lineno-lineno]	LLIST [lineno-lineno]	CLOAD ["file-name"]	LOG(exp)	MID$(string, start,[,length])		NEW	NEXT [var]					OUT port,byte	PEEK(addr)	POKE addr,byte	PRINT #"filename", exp[,exp . . .] NB prints to tape
ZX SPECTRUM	INKEY$	INPUT [string;] var	INT (exp)	string (TO finish)	LIST [lineno] Note: will fill screen then ask SCROLL	LLIST [lineno]	LOAD "filename" [code start, length]	LN(exp)	string(start TO finish)		NEW	NEXT var				Consult Microdrive manual	OUT port,byte	PEEK(addr)	POKE addr,byte	Consult manual

MACHINE	RANDOMIZE Reset random number generator.	READ Read from data statements into specified variables	RENUM Change program line numbers.	RESTORE Resets pointer to facilitate re-reading of DATA statements.	RESUME Return from ON ERROR subroutine to stmt that caused error.	RETURN Return from subroutine to statement following last GOSUB executed.	RIGHT$ Returns specified no. of characters starting at end of string.	RND Generates a random number.	RUN Execute a program.	SAVE Save a program either onto disk or tape.	SGN Returns 1 if exp>0 0 if exp=0 −1 if exp<0.	STRING$ Returns a string of specified length containing specified character.	STR$ Converts a numeric expression to a string.	SYSTEM Close files for return to operating system.	TROFF Trace off.	TRON Trace on.	USR Calls an assembler language subroutine which returns one value.	WAIT Suspend program execution for specified time.	WHILE/END Execute statements in WHILE/WEND loop as long as exp is true	WIDTH Sets printer carriage/screen width.
STANDARD MICROSOFT	RANDOMIZE [exp]	READ var [,var . . .]	RENUM [lineno, val]	RESTORE	RESUME	RETURN	RIGHT$ (string, length)	RND[exp]	RUN [lineno]	SAVE filename	SGN(exp)	STRING$(length string)	STR$(exp)	SYSTEM	TROFF	TRON	USR(parameter)	WAIT port,mark [,select]	WHILE exp WEND	WIDTH(val)
AMSTRAD 464/664/6128	RANDOMIZE (exp)	READ var [,var . . .]	RENUM [new start no ,old start no ,inc]	RESTORE [lineno]	RESUME [line no] or RESUME NEXT	RETURN	RIGHT$ (string, length)	RND[exp]	RUN ["filename"] [line no]	SAVE "filename" [, file type] [,binary parms]	SNG(exp)	STRING$ (length,string)	STR$(exp)		TROFF	TRON	CALL addr [,parms]	WAIT addr, mask [,inversion]	WHILE exp WEND	WIDTH exp
APPLE II		READ var [,var . . .]		RESTORE	RESUME	RETURN	RIGHT$(string, length)	RND(-exp)	RUN [lineno]	SAVE [filename [,binary parms]]	SGN(exp)		STR$(exp)		NOTRACE	TRACE	USR(parameter)	WAIT addr, exp [,exp]		POKE 32, left margin: POKE 33, screen width
ATARI	RND (-exp)	READ var [,var . . .]		RESTORE [lineno]		RETURN	string(start NB: not strictly equivalent	RND(exp) Note: exp is a dummy variable	RUN [lineno]	CSAVE "filename" [cass] or SAVE ["fileno:filename" [disk]	SGN(exp)		STR$(exp)	BYE NB: note equivalent			USR(addr,parameter [,parameter . . .])			POKE 82, val [left margin]: POKE 83, val [right margin]
BBC	RND (-time)	READ var [,var . . .]	RENUMBER [start lineno][,interval]	RESTORE [lineno]		RETURN	RIGHT$(string, length)	RND(exp)	RUN	SAVE "filename" Note: see note under LOAD	SGN(exp)	STRING$(length, string)	STR$(exp)	*DISK NB: disk-handling done through Basic so not true eq.	TRACE OFF	TRACE ON	USR(parameter)	(no WAIT stmt but see INKEY$]	REPEAT stmt UNTIL exp Note: reverse logic	WIDTH val Note: 0='unlimited'
COMMODORE 64 & VIC 20	RND (-TI)	READ var [,var . . .]		RESTORE		RETURN	RIGHT$(string, length)	RND(exp) Note: exp is a dummy for VIC	RUN [lineno]	SAVE ["filename"] [cass] or SAVE "filename",8 [disk]	SNG(exp)		STR$(exp)				USR(parameter)	WAIT addr, exp[,exp]		
IBM PC-BASIC A	RANDOMIZE (exp)	READ var [,var, . . .]	RENUM [new start no] [,old start no] [,inc]	RESTORE [lineno]	RESUME	RETURN [lineno]	RIGHT$ (exp, length)	RND(exp)	RUN [lineno]	SAVE "filename" [,A,P]	SGN(exp)	STRING$ (length,string)	STR$(exp)	SYSTEM	TROFF	TRON	USR(exp)	WAIT port, exp[,exp]	WHILE exp WEND	WIDTH exp
MEMOTECH MTX 512	RAND (exp)	READ var [,var, . . .]		RESTORE [lineno]		RETURN	RIGHT$ (exp, length)	RND(exp)	RUN [lineno]	SAVE "filename"	SGN(exp)		STR$(exp)	BYE			USR(parameter)	PAUSE (delay)		
MICROBEE		READ [(lineno.)] var[,var]	RENUM (new-start (,increment (,start-line (,finish-line))))	RESTORE (lineno.)		RETURN	var(;LEN(var)-n-1) -n - number of characters required	RND	RUN	SAVE "filename" - 200 bpi SAVEF "filename" - 1200 bpi	SGN(real-exp)	PRINT [An m] -n=length of string; m=ASCII code of character	STR(exp)		TRACE OFF	TRACE ON	USR(address (, integer-exp))	PLAY 0, int (1(int(255; 1-1/8 second)		ZONE (integer-exp) 1 < integer-exp < 16
MSX BASIC	RND (-time)	READ var [,var, . . .]	RENUM [new start no] [,old start no] [,inc]	RESTORE [lineno]	RESUME [lineno] or RESUME TEXT	RETURN [lineno]	RIGHT$ (STRING, length)	RND Note: X=dummy val	RUN [lineno]	CSAVE "filename"	SGN(exp)	STRING$ (length,string)	STR$(exp)		TROFF	TRON	USR(parameter)			WIDTH(exp)
TANDY 100	RND (-exp)	READ var [,var . . .]		RESTORE [lineno]	RESUME [lineno] or RESUME NEXT	RETURN	RIGHT$(string, length)	RND (exp)	RUN [lineno]	CSAVE ["filename"]	SNG (exp)	STRING$ (length,string)	STR$(exp)	CALL 0 — similar effect						
TANDY COLOR		READ var [,var . . .]	RENUM [lineno, start, interval]	RESTORE		RETURN	RIGHT$(string, length)	RND(exp)	RUN [lineno]	CSAVE "filename"	SGN(exp)	STRING$(length, string)	STR$(string)		TROFF	TRON	USR(parameter)			
SINCLAIR QL	RANDOMISE [exp]	READ var [,var . . .]	RENUM [[old start no] [TO old end no;] [new start no], [inc]]	RESTORE [lineno]	RETRY	RETURN exp	stringname (first car to last char	RND [exp TO exp]	RUN [lineno]	SAVE "filename" [lineno [to lineno]] . . .	SGN(exp)	FILL$(string, length)	Note: conversion automatic on assignment	Note: disk handling done through Basic			See CALL (or use EXEC)	PAUSE [delay]	REPEAT name IF cond EXIT name END REPEAT name	WIDTH [#chanel,] exp
TRS-80 II/SYSTEM 80	RANDOM	READ var [,var . . .]	RENUM start, interval Note: System 80 only	RESTORE [lineno/ exp]	RESUME [lineno/exp]	RETURN	RIGHT$(string, length)	RND(exp)	RUN [lineno]	CSAVE "filename" [cass] or SAVE "filename" [disk/ floppy tape]	SGN(exp)	STRING$(length, string)	STR$(exp)		TROFF	TRON	USR(parameter)			
VZ-200		READ var [,var . . .]		RESTORE		RETURN	RIGHT$(string, exp)	RND(exp) NB: Nonstandard — see VZ200 manual P58	RUN [lineno]	CSAVE ["filename"]	SGN(exp)		STR$(exp)				USR(parameter)			
ZX SPECTRUM	RAND (exp)	READ var [,var . . .]		RESTORE [lineno/ exp]		RETURN	string(TO start)	RND	Run [lineno/ var/exp]	SAVE "filename" [CODE start, length] [SCREEN$]	SGN(exp)		STR$(exp)				USR addr	PAUSE no. of frames (50/second)		