8 Concise reference guide to AMSTRAD BASIC

A listing of BASIC keywords illustrated by examples, giving the precise form of use and associated keywords.

Subjects covered in this chapter

* The form of the notation
* Special characters and their significance
* All AMSTRAD BASIC keywords in alphabetical order

This chapter contains a concise summary of the functions in the BASIC supplied in ROM with the CPC464. The range of features available represents an industry-standard implementation of the language with specific extensions to complement the hardware features of the CPC464.

8.1 **Notation.**

Special Characters

& or **&H**	Prefix for hexadecimal constant
&X	Prefix for binary constant
:	Separates statements typed on the same line
#	Prefix for stream director

Data types

Strings may be from 0 to 255 characters long, and are expressed as (string expression,. Strings may be appended to one another using the + operator, as long as the resulting ‹**string** expression> is less than 255 characters long.

Numeric data can be either Integer or real. Integer data is held in the range **-32768....32767** and real data is held to a little over nine digits of precision in the range **±1.7E+38** with the smallest value above zero approximately **2.9E-39**.

Type markers are % Integer, **!** real, **$** string.

A ‹**numeric** expression> is any expression that results in a numeric value: it may simply be numbers, or it may be a numeric variable, or it may be numbers operated on by variables. Just about anything that is not a ‹**string** expression,.

A &ream expression, refers to a ‹**numeric** expression') which identifies the screen, printer or cassette where the text is required to 'stream'.

An 'improper argument' means that a ‹**numeric** expression, has returned a value that is outside the range defined as permissable in that situation or that a command parameter is invalid in some **way**.**and** that BASIC has not. accepted this as valid input.

KEYWORDS

Please note that AMSTRAD BASIC keywords are listed here using the form:

KEYWORD
Syntax/Function
Example
Description
Associated KEYWORDS

IMPORTANT:

Keywords are either

COMMANDS : operations that are executed directly.
FUNCTIONS : operations that are invoked as arguments in an expression

Brackets

() are required as part of the command or function. Other types of brackets used in the KEYWORD description are for the purposes of the description, and should not be typed as part of the line:
[] enclose optional items.
‹› enclose various expressions which are described in the subsequent description.

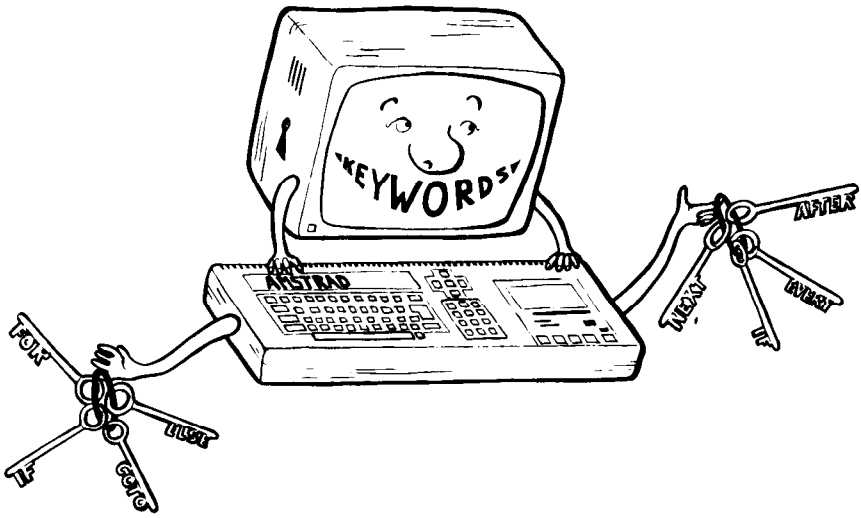
Quotes

Only **""** form part of the actual BASIC program structure. " are used to emphasise or highlight certain aspects of the description. " should not appear in any part of the syntax/Function or Example entries other than within ‹**string expression**›**s**.

Entering

BASIC converts all keywords entered in **l** owe ***r case*** letters into *UPPER C A S E* when a program is L I S Ted. The examples shown here use U P P E R C A S E, since this is how the program will appear when L I S Ted - if you enter using **l** owe r case, you will be able to spot typing errors more readily since the mistyped keyword will still be displayed in 1 ***owe*** r c a s ***e*** when L I S Ted.

Keywords are delimited by separators, since AMSTRAD BASIC allows you to 'bury' keywords into variable names: eg end2 and LI *ST CO DE* are acceptable as variables in AMSTRAD BASIC.



ABS

A B S (numeric expression)

```
PRINT ABS(-67.98)
 67.98
```

FUNCTION: Returns the absolute value of the given expression -which primarily means that negative numbers are returned as positive.

Associated keywords: S G N

AFTER

A F T E R integer expression›[, integer expression)] G 0 S *U B* line number.

```
AFTER 200,2 GOSUB 320
```

COMMAND: Invoke a subroutine after a given time period has elapsed. The first (integer expression, indicates the period of the delay, in units of 1/50 second, and the second (integer expression, (in range 0....3), indicates which of the four available delay timers should be used. See also Chapter 10.

Associated keywords: E V E R Y , R E M A I N

ASC

A S C (string expression))

```
PRINT ASC("X")
 88
```

FUNCTION: Gets the numeric value of the first character of a string as long as ASCII characters are used.

Associated keywords: C H R $

ATN

ATN(numeric expression) **)**

```
PRINT ATN(1)
 0.785398163
```

FUNCTION: Calculates the arc-tangent (forcing the numeric expression) to a real number ranging from -PI/2 to **+PI/2)** of the value specified.

Associatedkeywords: SIN, COS, **TAN, DEG, RAD**

AUTO

AUTO [line **number**›][**,** increment,]

```
AUTO 100,50
```

COMMAND: Generate line numbers automatically. The line number, sets the first line to be generated, in case you want to add to the end of an existing program. The value of the ‹**increment**› between line numbers, and the first line number to be generated, both default to 10 if not specified.

Where an existing program line is in danger of being overwritten, BASIC inserts a star * after the line number generated as a warning.

BIN$

BIN$ (unsigned integer expression:›[, <integer expression] **)**

```
PRINT BIN$(64,8)
01000000
```

FUNCTION: Produces a string of **binary** digits that represents the value of the unsigned integer expression>, filling with leading zeros to the number of digits instructed by the second ‹**integer** expression,.

Associated keywords: HEX**$,** STR**$**

BORDER

BORDER ‹**colour**›[**,**‹**colour**›]

```
BORDER 3,2
```

COMMAND: To change the **colour** of the border on the screen. If two colours are specified, the border alternates between the two at the rate determined in the SPEED INK command, if given. The range of border colours is **0....26.**

Associated keywords: SPEED INK

CALL

CA L L (address expression›[, list of ‹parameter›]
CALL **&BD19**

COMMAND: Allows an externally developed sub-routine to be invoked from BASIC. Use with caution, not a function for the inexperienced to experiment with. The above CA L L is relatively harmless, since it waits for the next frame **flyback,** which is particularly useful for tidying up the movement of characters around the screen when producing animation effects.

Associated keywords: U N T

CAT

CAT

CAT

COMMAND: Causes BASIC to start reading the cassette and to display the names of all files found. This does not affect the program currently in memory, and **so** may be used to verify a program that has just been saved before altering the program memory. The Command asks you to play the cassette, and on finding a program responds:

FILENAME **Block Number** Flag Ok

Flags indicate the type of recording made:

$ a BASIC program file
% a Protected BASIC file
* an ASCII text file
& a Binary file

Other characters may occur in this column if the file was not produced by BASIC.

Associated keywords: L 0 A D , R U N , S A V E

CHAIN
CHAIN MERGE

C HAIN ‹file name›[, line number expression)]
C HAIN MERGE ‹file name›[, line number expression]
[,DELETE‹line number range)]

CHAIN "TEST", 350

COMMAND: C H A I N loads a program from cassette into the memory; replacing the existing program. C HA I N ME R G E merges a program from cassette into the current program memory. It adds the contents of a file to the current program in memory. The line number expression, indicates the line number from which execution is to begin once the new program is chain merged. In the absence of line number expression>, BASIC will default to the lowest line number available.

If no file name is stated, then BASIC will attempt to merge the first valid file encountered on tape. If the first character of the filename is a ! , then it is removed from the filename, and suppresses the usual messages generated by the cassette reading process.

C H A I N M E R G E retains all current variables although User Functions and open files are discarded. ON E R R 0 R G 0 T 0 is turned off, a R E S T 0 R E is implemented and the DEFINT, DEFREAL & DEFSTR settings are reset, and all active FOR W H I L E and G 0 S U B commands are forgotten. Protected files will not merge.

Associated keywords: L 0 A D , M E R G E

CHR$

C HR$((integer expression))

```
PRINT CHR$(100)
 d
```

FUNCTION: Converts a numeric value to its character equivalent, (using the AM S T R A D C P C 4 6 4 character set in Appendix III)

Associatedkeywords: ASC, LEFTS, RIGHT$,MID$,STR$

CINT

C I NT(numeric expression,)

```
10 n=578.76543
20 PRINT CINT(n)
RUN
 579
```

FUNCTION: Converts the given value to a rounded integer in the range -32768...32767.

Associatedkeywords: CREAL, INT, FIX, ROUND, UNT

CLEAR

CLEAR

CLEAR

COMMAND: Clears all variables and files.

CLG

C LG[‹masked ink›]

CLG

COMMAND: To clear the graphics screen.

Associated keywords: CLS, ORIGIN

CLOSEIN

CLOSEIN

CLOSEIN

COMMAND: Close the cassette input file. Commands such as NEW and CHAIN MERGE will abandon any open files.

Associated keywords: OPENIN, CLOSEOUT

CLOSEOUT

CLOSEOUT

CLOSEOUT

COMMAND: Close the output cassette file.

Associated keywords: OPENOUT, CLOSEIN

CLS

C L S [# stream expression›]

```
CLS
```

COMMAND: To clear the given screen window to its Paper Ink.

CONT

CONT

CONT

COMMAND: Continue program execution after a *B r e a k *, S *T 0 P* or *E N D*, as long as the program has not been altered. Direct commands may be entered.

COS

C 0 S (numeric expression)

```
?COS(34)
-0.848570274
```

and

```
deg:?cos(34)
 0.829037573
```

FUNCTION: Calculates the C 0 S I N E of a given value. The function defaults to radian measure unless specifically instructed otherwise by the D E G command. Note in the above example the use of ? shortform for P R I NT, and the use of lowercase entry for keywords - a feature fully compatible with AMSTRAD BASIC.

Associatedkeywords: SIN, *TAN*, *ATN*, *DEG*, RAD

CREAL

C *R E A L* (numeric expression,)

```
5 DEFINT n
10  n=75.765
20 d=n/34.6
30 PRINT d
40 PRINT CREAL(n)
50 PRINT n/55.4
run
2.19653179
76
1.37184116
Ready
```

FUNCTION: Converts a value to a real number. (As opposed to integer).

Associated keywords: C I N T

DATA

DATA ‹listof: constant)

```
10 REM Proofreader list
20 DIM Proofer$(5)
30 DIM ProoferSurname$(5)
40 FOR n=1 to 5
50 READ Proofer$(n)
60 READ ProoferSurname$(n)
65 PRINT Proofer$(n);" "ProoferSurname$(n)
70 DATA Bob,Smith,Dicky,Jones,
   Malcolm,Green,Alan,Brown,Ivor,Curry
90 NEXT
```

COMMAND: Declares constant data for use within a program. One of the most widely used features of BASIC that lumps constant data in D AT A statements for retrieval as required. The data type must be consistent with the variable invoking it. A DA T A statement may appear anywhere in a program.

Associated keywords: R E A D , R E S T 0 R E

DEF FN

D E F FN‹name›[(‹formal parameters>)]=‹general expression

```
10 DEF FNinterest(principle)=1.14*principle
20  INPUT "What is the principle sum";principLe
30  PRINT "The amount owing plus interest
    after one year is";FNinterest(principle)
```

COMMAND: BASIC allows the program to define and use simple value returning functions. D E FFuNction is the definition part of this mechanism and creates program-specific function which works within the program in the same way as a function such as C 0 S operates as a built-in function of BASIC.

It may be invoked throughout the program. Variable types must be consistent and the D E FFuNction command should be written in part of the program outside the execution loop.

DEFINT
DEFSTR
DEFREAL

D E F type range(s) of letters,

```
DEFINT  I-N
DEFSTR  A,W-Z
DEFREAL
```

COMMAND: Define default variable types where 'type' is integer, real or string. The variable will be set according to the first letter of the variable's name -which may be either upper or lower case.

Associatedkeywords: *LOAD, RUN, CHAIN,NEW, CLEAR*

DEG

DEG

DEG

COMMAND: Set degrees mode. The default condition is for functions such as S I N and CO S is to assume radian measure for numeric data. The command sets to degree mode until instructed otherwise by a C *LEAR* or *RA D* - or if any new program is loaded.

Associated keywords: R A D

DELETE

DELETE ‹line number range›

DELETE 100-200

COMMAND: A command that removes part of the current program as defined in the line number range, expression. Not recoverable if issued in error, so use with care, and check for mistyping before entering.

Associated keywords: N E W

DI

DI

```
10 CLS
20 TAG
30 EVERY 10 GOSUB 100
40 X1=RND*320:X2=RND*320
50 Y=200+RND*200
60 FOR x=320-X1 TO 320+X2 STEP 2
70 DI:PLOT 320,0,1:MOVE X-2,
   Y:PRINT " ";:MOVE X,Y:PRINT "#";:EI
80 NEXT
90 GOTO 40
100 MOVE 320,0
110 DRAW X+8,Y-16,0
120 RETURN
```

COMMAND: Disable interrupts (other than the *Break* interrupt) until re-enabled explicitly by EI or implicitly by the RETURN at the end of an interrupt GOSUB routine.

Used when the program wishes to get on literally without interruption - for example when two routines within a program are competing for use of resources. In the example above, the main program and the interrupt subroutine are competing for the use of the graphics display.

Associated keywords: E I

DIM

DIM ‹list of: subscripted variable,

```
10 CLS:PRINT "Enter 5 names....":PRINT
20 DIM B$(5)
30 FOR N=1 TO 5
40 PRINT "Name"N"please";
50 INPUT B$(N)
60 NEXT
70 FOR N=1 TO 5
80 PRINT "How wise of you ";B$(N);
  " to buy a CPC464"
90 NEXT
```

COMMAND: Allocate space for arrays and specify maximum subscript values. Basic must be advised of the space to be reserved for an array, or it will default to 10. Once set either implicitly or explicitly, the size of the array may not be changed, or an error will result.

A subscripted variable, is one where the same variable name can take a series of values as set out in the list of integer numbers that comprise the 'dimension list'. The first number in the dimension list may be thought of as levels in a multistorey car park, and the subsequent numbers, the number of parking bays etc. A full understanding of arrays is a major element in advanced BASIC programming. The size of an array is limited only by available memory, and the programmers ability to keep track of the entries in the dimension list.

Associated keywords: ERASE

DRAW

DRAW ‹x co-ordinate›,‹y co-ordinate›[,<masked ink›]

```
DRAW 200,200,13
```

COMMAND: Draws a line on the screen from the current graphics cursor position to an absolute position. The co-ordinate positions remain unchanged between the three different screen modes. Further examples in Chapter 5.

Associated keywords: DRAWR, *PLOT, PLOTR, MOVE, MOVER, TEST, TESTR, XPOS, YPOS,* **ORIGIN**

DRAWR

DRAWR ‹x offset›,‹y offset›[,‹masked ink›]

```
DRAWR 200,200,13
```

COMMAND: To draw a line on the screen from the current graphics cursor position to a position relative to it.

Associated keywords: *DRAW, PLOT, PLOTR, MOVE, MOVER, TEST, TESTR, XPOS, YPOS, ORIGIN*

EDIT

EDIT line number

```
EDIT 110
```

COMMAND: Edit a program line by calling for a specific line number. See Chapter 1.4.

Associated keywords: L I S T

EI

```
EI
```

EI

COMMAND: To Enable Interrupts disabled by a D I command.

Associated keywords: D I

END

```
END
```

```
END
```

COMMAND: End of program. An **END** is implicit in **AMSTRAD** BASIC as the program passes the last line of instruction. EN D closes all cassette files and returns to the direct mode. Sound queues will continue until empty.

Associated keywords: S T 0 P

ENT

EN T <envelope number›[, <envelope sections,]

```
10 ENT 1,100,2,20
20 SOUND 1,100,1000,4,0,1
```

COMMAND: While a sound is being generated, it is possible to vary its tone. A tone envelope defines how the tone is to be varied. See Chapter 6 and Appendix VII for a fuller description of sound and its operation.

The <envelope number is an <integer expression) whose absolute value must be in the range 1.. 15, which specifies the tone envelope to set. If the <envelope number is negative, then the envelope is repeated.

Up to five <envelope sections may be supplied, and each may take one of the forms:
step ‹count›,‹step size›,‹pause time›
or: =‹tone period›,‹pause time›

The first form specifies an incremental change relative to the current tone period setting. The second form specifies an absolute setting for the tone period. Where :
step ‹count› gives the number of steps in the section - an (integer expression in the range 0..239.

‹step size› gives the amount by which to vary the tone period at each step in the envelope - an ‹integer expression yielding a value in the range -128.. + 127.

‹pause time› gives the time to wait between steps - an ‹integer expression, specifying the time in 0.01 second units. The expression must yield a value in the range 0..255 (where 0 is treated as 256).

tone period, gives the new setting for the period - an (integer expression yielding a value in the range 0..4095.

The SOUND command sets the initial tone period, and may specify one of the fifteen tone envelopes. If no envelope, or an envelope which has not been set up is specified, then the tone remains constant throughout the sound.

A tone envelope has no effect on the duration of the sound. If there are steps remaining in the tone envelope when the sound finishes, they are simply abandoned.

A repeating tone envelope will be restarted each time it finishes until the sound terminates.

The expressions in the tone envelope are evaluated when the command is executed and the results stored away for future use. Using the tone envelope does not cause the command to be re-executed.

Each time a given tone envelope is set, its previous value is lost. Changing an envelope while a sound using it is active or pending will produce indeterminate (but possibly interesting) effects.

Specifying an envelope with no sections cancels any previous setting. Any further use of the envelope will be ignored and the default used instead.

Associated keywords: E NV , S 0 U N D

ENV

ENV <envelope number›[, <envelope sections)]

```
10 ENV 1,100,2,20
20 SOUND 1,100,1000,4,1
```

COMMAND: While a sound is being generated, it is possible to vary its volume using this command. A volume envelope consists of:

step ‹count› , ‹step size), ‹pause time›

and defines how the volume is to be varied, by specifying a number of steps in the range 0..127, the size of the step in the range -128....+127, and the pause time in 0.01 second intervals in the range 1....256.

The <envelope number is an ‹integer expression, yielding a value in the range 1..15 specifying the volume envelope to set.

Up to five (envelope sections, may be given. Each may take one of the forms:
step count›,‹step size›,‹pause time›

or: =‹hardware envelope›,‹envelope period,

The first form specifies an envelope section under software control, where the parameters are :

step count gives the number of steps in the section - an integer expression in the range 0..127.

step size› gives the amount by which to vary the amplitude at each step in the envelope - an integer expression) in the range -128....+127.

or : if the step count is zero, then the value to set the amplitude to - that is an absolute setting.

‹pause time) gives the time to wait between steps - an ‹integer expression specifying the time in 1/100th's of a second. The expression must yield a value in the range 0....255 (where 0 is treated as 256).

The second form specifies an envelope section to be executed directly by the sound hardware, where :

<hardware envelope, is the value to be set into the envelope shape register (register 15, octal).

<envelope period, is the value to be set into the envelope period registers (registers 13 & 14, octal).

Hardware envelope settings do not have an associated pause time, so the next section of the envelope is immediately executed. It is advisable therefore that the next step should have a pause of a suitable length. If there is no next step, then a pause of 2 seconds is taken.

The **SOUND** command sets the initial volume, and may specify one of the fifteen volume envelopes. If no envelope, or an envelope which has not been set up, is specified, then the volume remains constant throughout the sound.

Setting a step size› of zero with a non-zero step count causes the current volume setting to be maintained.

The expressions in the amplitude envelope are evaluated when the command is executed and the results stored away for future use. Using the amplitude envelope does not cause the command to be re-executed.

Each time a given amplitude envelope is set, its previous value is lost. Changing an envelope while a sound using it is active or pending will produce indeterminate (but possibly interesting) effects.

Specifying an envelope with no sections cancels any previous setting. Any further use of the envelope will be ignored and the default used instead.

Associated keywords: E N T, S 0 U N D

EOF

EOF

```
PRINT EOF
-1
```

FUNCTION: Tests to see if the cassette input is at the end of the file. Returns -1 (true) at the end, otherwise 0 (false).

Associated keywords: 0 P E N I N

ERASE

ERASE list of; variable **name**››

ERASE **A,B$**

COMMAND: When an array is no longer required, it may be *ERAS* Ed and the memory used be reclaimed ready for other use.

Associated keywords: *D I* **M**

ERR
ERL

ERR
ERL

```
10 CLS
20 ON ERROR GOT0 1000
30 DATA SALLY,EMMA,JOANNE,HELEN,GEMMA
40 READ A$
50 PRINT A!$
60 GOT0 40
70 REM error detection starts here
1000 IF ERR=4 THEN PRINT "I have spotted
 a DATA EXHAUSTED error!"
1010 IF ERL<70 AND ERL>20 THEN PRINT
 "..... in lines 30-60"
1020 END
```

VARIABLES: These variables are used in error handling subroutines to discover the error number and line where the error **occured.** See the error message listing in Appendix VIII.

Associated keywords: 0 N ERR0R,ERR0R

ERROR

ERROR integer expression)

ERROR 17

COMMAND: Take Error action with a given error number. The error may be one already used and recognised by BASIC (Appendix VIII), in which case the action taken is the same as would be taken if such an error had been detected by BASIC. Error numbers beyond those recognised by BASIC may be used by the program to signal its own errors.

Associatedkeywords: ON *'ERROR, ERR, ERL*

EVERY

EVERY integer **expression**›[, <integer expression] GOSUB (line number

EVERY **500,2 GOSUB** 50

COMMAND: The CPC464 maintains a real time clock. The *EVERY* command allows a BASIC program to arrange for subroutines to be called at regular intervals. Four delay timers are available, specified by the 2nd ‹**integer** expression in the range **0....3** each of which may have a subroutine associated with it. See also Chapter 10.

Associated keywords: AFTER, REMAIN

EXP

EXP (numeric expression>)

```
PRINT EXP(6.876)
 968.743625
```

FUNCTION: Calculates 'E' to the power given in numeric expression - where 'E' is approximately 2.7182818 - the number whose natural logarithm is 1.

Associated keywords: LOG

FIX

FIX(‹**numeric** expression>)

```
PRINT FIX(9.99999)
 9
```

FUNCTION: Unlike CINT, FIX merely removes the part of the numeric expression, to the right of the decimal point, and leaves an integer result, rounding towards zero.

Associated keywords: CINT, INT, *ROUND*

FOR

FOR simple variable, **=** ‹**start**› **TO** ‹**end**› [ST **EP** ‹**size**›]

FOR DAY=1 to 5 **STEP** 2

COMMAND: Execute a body of program a given number of times, stepping a control variable between a start and an end value. If not specified, S T E P defaults to 1.

Associated keywords: N E X **T**, W H I L E

FRE

FRE(numeric expression **)**

F R **E(**‹**string** expression> **)**

PRINT FRE(0)

PRINT FRE("")

FUNCTION: Establishes how much **memory** remains unused by BASIC. The form **FRE("")** forces a 'garbage collection' before returning a value for available space.

GOSUB

GOSUB ‹**line** number ›

GOSUB 210

COMMAND: Call a BASIC subroutine by branching to the specified line number.See **RETURN.**

Associated keywords: R E T U R N

GOTO

GOTO ‹**line** number)

GOTO 9 0

Branch to specified line number.

HEX$

H E X **$ (** unsigned integer **expression›[,‹integer** expression,] **)**

```
PRINT HEX$(65534)
FFFE
```

FUNCTION: Converts the number given into Hexadecimal form. (See Appendix II). The second (integer expression can be used to specify the minimum length of the result.

Associated keywords: B I N **$,** S T R **$**

HIMEM

```
HIMEM

?HIMEM
 43903
```

VARIABLE: Gives the address of the highest byte of memory used by BASIC, and can be used in numeric expressions in the usual way.

Before resetting the highest byte of available memory using the ME MO RY command, it is advisable to issue the command m **m=** H I M E **M**. *You* will thereafter be able to return to the previous memory capacity by issuing the command: *MEMORY mm*.

Associated keywords: **F** R E , **M** EM 0 R Y

IF

IF

I F logical expression) T H E N ‹option part›[*E L* **S E** ‹option part›]

I F logical expression, G 0 T 0 line number [*E LS E* (option **part›**]

IF **A>B** *THEN A=C ELSE A=D*

IF **A>B** *GOTO 1000 ELSE 300*

COMMAND: A very widely used command that is used to conditionally determine branch points in a program. The logical part is evaluated, and if true the T H E N or G 0 T 0 part is executed, if false, the program skips to the E LS E part, or merely passes onto the next line. The statements may be nested to any depth, limited by line length. The I **F** is terminated by the line end. It is not permissable to have further statements that are independent of the I **F** on the same line.

Associatedkeywords: *THEN , ELSE , GOTO , OR , AND , WHILE*

INK

INK ‹ink›, colour&colour)]

INK 0,1

COMMAND: Depending on the current Screen Mode (Chapter 5), a number of INKs are available. The colour, or colours, used for an INK may be changed by an INK command, according to the table of colour values in Appendix IV.

Associated keywords: PEN, PAPER

INKEY

INKEY(integer expression>)

```
10 CLS:IFINKEY(55)=32 THEN 30 ELSE 20
20 CLS:GOTO 10
30 PRINT "You're pressing [SHIFT] and V"
40 FOR t=1 TO 1000:NEXT:GOTO 10
```

FUNCTION: This function interrogates the keyboard to report which keys are being pressed. The keyboard is scanned at 1/50 sec. The function is particularly useful for spotting Y/N responses, since the state of shift is not required according to one of the interpretation options. The above example detects when **[SHIFT]** and **V** are pressed together. Shift and control are identified according to

Value returned	[SHIFT]	[CTRL]	KEY
-1	?	?	UP
0	UP	UP	DOWN
32	DOWN	UP	DOWN
128	UP	DOWN	DOWN
160	DOWN	DOWN	DOWN

Associated keywords: INPUT, INKEY$

INKEY$

INKEY$

```
10 CLS
20 PRINT "Are you clever (y or n) ?"
30 a$=INKEY$: IF a$="" GOTO 30
40 IF a$="N" OR a$="n" THEN
     PRINT "You must have been to buy me!":END
50  IF a$="Y" OR a$="y" THEN
     PRINT "You're too modest!!!":END
60 GOTO 20
```

FUNCTION: Reads a key from the keyboard to provide operator interaction without hitting **[ENTER]** after every answer. If there is a key pressed, then the function responds - if no key is pressed, it continues to return an empty string which is used to loop until a valid input is detected for processing.

Associated keywords: INPUT, INKEY

INP

INP (port number,)

```
PRINT INP(&FF77)
```

FUNCTION: A function that returns the input value from the I/O port specified in the address.

Associated keywords: OUT, WAIT

INPUT

INPUT [# <stream expression, I];][‹quoted string) ;]‹list of: [variable],
or INPUT *[#(stream* expression,][;][<quoted string, ,] list of: [variable],

```
10 CLS
20 INPUT "Give me two numbers, separated by
  a comma ";A,B
30 IF A=B THEN PRINT "The two numbers
  are the same"
40 IF A>B THEN PRINT A "is greater than" B
50 IF A<B THEN PRINT A "is less than" B
60 CLEAR:GOTO 20
```

COMMAND: Reads data from the stated stream. A semicolon after INPUT suppresses the carriage return typed at the end of the line being entered. A semicolon after the {quoted string) causes a question mark to be displayed. A comma suppresses the question mark. If an entry is made that is of the wrong type (eg a letter 0 was typed instead of a 0 in a numeric expression, then BASIC will prompt with:

```
?Redo from start
```

..and the original prompt text that you entered.

All responses must be terminated with an **[ENTER].** The semicolon immediately after the stream expression) has the effect of suppressing the carriage return typed at the end of the line being entered, leaving the cursor at the end of the text just entered. Where a cassette stream is indicated, no prompt is generated, If one is specified, it will be ignored by the cassette software, so the same program may read from either stream.

One item will be read from the stream for each variable in the list given. It must be compatible with the type specified in the INPUT command, which is: a numeric variable, terminated either by comma, carriage return, white space or end of file. Commas or **[ENTER]**s sent after trailing space will be ignored. Quoted strings will he read verbatim until terminated by double quotes, subsequent entries are ignored as for numeric values. Unquoted string items are terminated as in the case of numeric values.

Associatedkeywords: *LINE INPUT, READ,* INKEY$

INSTR

I N S T R ([integer expression> **,**]‹**string** expression,, string expression) **)**

```
PRINT INSTR(2,"BANANA","AN")
```

FUNCTION: Searches the first string expression, for the first occurance of the second string expression), where the optional number at the start indicates where to start the search - otherwise the search begins at the first character of the first string expression).

Associatedkeywords: **MID$,LEFT$,RIGHT$**

INT

I N T **(** numeric expression> **)**

```
PRINT INT(-1.995)
-2
```

FUNCTION: Rounds the number to the nearest lower integer, removing any **frac**tional part. The same as **F** I X for positive numbers, but returns one less than FIX for negative numbers not already integers.

Associated keywords: C I NT , **F** I X, R 0 UN D

JOY

J 0 Y **(**‹**integer** expression **)**

```
10 IF JOY(0) AND 8 THEN GOTO 100
```

FUNCTION: The J 0 Y function reads a bit-significant result from the joystick specified in the <integer expression> (either 0 or 1).

Bit	Decimal
0: up	1
1: Down	2
2: **Left**	4
3: Right	8
4: Fire 2	16
5: Fire 1	32

Associated keywords: I N K E Y

KEY

KEY ‹integer expression,, string expression)

```
KEY 140,"RUN"+CHR$(13)
```

COMMAND: Fixes a new function key definition. There are thirty two keyboard 'expansion' characters in the range 128-159 listed in Appendix III. When one of these characters is read it is expanded into the string associated with it - although the total number of expansion characters cannot exceed 100. The KEY command associates a string with a given expansion character.

Associated keywords: KEY DEF

KEY DEF

KEY DEF (key number, , ‹repeat›[,‹normal›[,‹shifted›[,‹control›]]]

```
KEY DEF 46,1,63
```

COMMAND: Associates the value as defined in Appendix III to a key on the keyboard. The above example converts the N key to print the question mark character ? . (Character number 63).

To return the key to its normal function:

```
KEY DEF 46,1,110
```

where the character number 110 is the lower case n.

Associated keywords: KEY

LEFT$

LEFT$ (string expression>, ‹integer expression>)

```
10 CLS
20 A$ = "AMSTRAD"
30 B$ = LEFT$(A$,3)
40 PRINT B$
RUN
     [SCREEN CLEARS]
AMS
Ready
```

FUNCTION: Extracts characters to the left of, and including the position specified in the ‹integer expression> from the the given ‹string expression). If the (string expression is shorter than the required length, the whole ‹string expression, is returned.

Associated keywords: MID$, RIGHT$

LEN

LEN(string expression **)**

```
A$="AMSTRAD":PRINT LEN(A$)
7
```

FUNCTION: Returns a number corresponding to the number of all types of characters, including spaces, in the string expression.

LET

LET ‹variable› **=** ‹expression›

LET *x=100*

COMMAND: A remnant from early BASICs where variable assignments had to be 'seen coming'. No use apart from providing compatibility with the programs supplied in early BASIC training manuals. The above example need only be typed:

x=100

using AMSTRAD BASIC.

LINE INPUT

LINE INPUT [<#stream expression,,][**;**] [quoted string;]‹**string** variable,
LINE INPUT [**<#stream** expression,,][**;**][**quoted** string,]‹**string** variable)

```
LINE INPUT A$
LINE INPUT "NAME"; N$
```

COMMAND: Reads an entire line from the stream indicated. The first optional semicolon suppresses the echo of carriage return / line feed. The default stream expression is, as always, # 0 :**screen**.

Associatedkeywords: **READ, INPUT, INKEY$,INPUT$**

LIST

LIST [line number **range**›][**,** #stream expression)]

LIST **100-1000,#1**

COMMAND: List program lines to the given stream. 0 is the default screen, 8 is the printer. L I S Ting may be suspended by pressing **[ESC]** once, and restarted by pressing the space bar. A double **[ESC]** will return BASIC to the direct mode. Programs may be listed to previously defined windows to assist in debugging programs without overwriting the entire screen area. Listing may be performed from the start of a program to a given point, or from a specified line number to the program end, by omitting the first or last numbers in the line number range,. eg.

LIST -200 or LIST **30-**

LOAD

LOAD ‹file name›[,‹address expression]

```
LOAD "INVENT"
```

COMMAND: To read a BASIC program from cassette into memory, replacing any existing program, or if using the optional address expression, to load a binary tile into memory. See Chapter 2.

LOCATE

LOCATE [#*stream* expression,,]‹x coord›,‹y coord›

```
10 MODE 1
20 LOCATE 20,12
30 PRINT CHR$(249)
```

COMMAND: Moves the text cursor at the stream indicated, to the position specified by the x and y co-ordinates, which are relative to the origin of the stream (WINDOW). Stream 0 is the default stream.

Associated keywords: WINDOW

LOG

LOG (numeric expression)

```
?LOG(9999)
 9.21024037
```

FUNCTION: Calculates the natural logarithm of numeric expression.

Associated keywords: EXP, LOG 10

LOG10

L O G 10 (numeric expression>)

```
?LOG10(9999)
 3.99995657
```

FUNCTION: Calculates the base 10 logarithm of numeric expression).

Associated keywords: E X P , L 0 G

LOWER$

L O W E R **$(** string expression))

```
A$="AMSTRAD":PRINT LOWER$(A$)
amstrad
```

FUNCTION: Returns a new string expression the same as the input string expression but in which all upper case characters are converted to lower case. Useful for processing input where the answers may come in mixed upper/lower case.

Associated keywords: *U P P E R* **$**

MAX

MA *X* (list of: numeric expression,)

```
10 n=66
20 PRINT  MAX(1,n,3,6,4,3)
```

FUNCTION: Extracts the largest value from the list of numeric expressions

Associated keywords: **M** I N

MEMORY

M *E* **M** *O R Y* [address expression

```
MEMORY &20AA
```

COMMAND: Beset BASIC memory parameters to change the amount of memory available by setting the address of the highest byte. See the description of the keyword H I M E M. To examine the amount of memory, use the **F** R E command.

Associated keywords: H I **M** EM , F R E

MERGE

MERGE [‹file name›]

MERGE "PLAN"

COMMAND: Merge a program from cassette into the current program memory. It adds the contents of a file to the current program in memory. If no file name is stated, then BASIC will attempt to merge the first valid file encountered on tape. If the first character of the filename is a '!', then it is removed from the filename, and has the effect of suppressesing the usual messages generated by the cassette reading process.

If you wish to merge a program into memory without overwriting the current program, then the current program lines should be R E N U Mbered to a range different from those in the incoming program.

All variables, User Functions and open files are discarded. ON ERROR GOT 0 is turned off, a *RESTORE* is implemented and the *DEFINT*, *DEFREAL* & *D E F S T R* settings are reset. Protected files will not merge.

Associatedkeywords: *LOAD*, *CHAIN*, *CHAIN MERGE*

MID$

M I D**$(**‹string›**,** integer expression4 , ‹integer expression] **)**

```
A$="AMSTRAD":PRINT MID$(A$,2,4)
MSTR
A$="AMSTRAD":b$=MID$(A$,2,4):PRINT b$
M S T R
```

COMMAND and FUNCTION: M I *D* **$** specifies part of a string (a sub-string) which can be used either as the destination of an assignment (**M** I D **$** as a command) or as an argument in a string expression (M I D **$** as a Function). The first ‹integer expression specifies the position of the first character of the sub-string.

The second integer expression specifies the length of the sub-string to be returned. If omitted, this extends to the end of the original string.

Associated keywords: L E **F** T **$**, R I G H T **$**

MIN

M I N **(** list **of:** numeric expression) **)**

```
PRINT MIN(3,6,2.999,8,9)
 2.999
```

FUNCTION: Extracts the smallest value from the list of numeric expressions

Associated keywords: MAX

MODE

MODE ‹integer expression,

MODE 1

COMMAND: To change the screen mode (0,1 or 2), and clear the screen to INK 0, which may not be the current PAPER INK. All text and graphics WINDOWS are reset to the whole screen, and the text and graphics cursors homed to their respective origins.

Associatedkeywords: WINDOW, ORIGIN

MOVE

MOVE ‹x coord›,‹y coord›

HOVE 34,34

COMMAND: To move the graphics cursor to a position specified by the absolute co-ordinates. YPOS and XPOS are the corresponding functions to establish the current graphics cursor position.

Associated keywords: *MOVER, PLOT, PLOTR, DRAW, DRAWR, ORIGIN, TEST, TESTR, XPOS, YPOS*

MOVER

MOVER ‹x offset›,‹y offset,

MOVER 34,34

COMMAND: To move the graphics cursor to a position relative to the current co-ordinates. YPOS and XPOS are the corresponding functions to establish the current graphics cursor position.

Associated keywords: MOVE, *PLOT, PLOTR, DRAW, DRAWR, TEST, TESTR, XPOS, YPOS*

NEW

NEW

NEW

COMMAND: Delete current program and variables. KEY definitions are not lost, and the display is not cleared.

NEXT

NEXT [list of:‹variable›]

FOR *n=l TO 1000:NEXT*

COMMAND: Delimits the end of a **F** 0 R loop. The NEXT command may be anonymous, or may refer to its matching F 0 R, which in the above example would be:

NEXT n

Associated keywords: **F** 0 R

ON GOSUB
ON GOT0

0 N integer expression) G 0 S *U B* ‹**list of:‹line number›**

0 N integer expression) G 0 T 0 ‹**list of:‹line number›**

10 ON *DAY* **GOSUB 100,200,300,400,500**

10 ON RATE GOT0 **1000,2000,3000,4000**

COMMAND: **GOSUB to** the subroutine, or G 0 T 0 the statement as directed *by the* result of the ‹**integer** expression,. If the result is 1, then the first line number in the list is chosen, if 2 then the second etc. In the above line 10, when DAY **=** 1, the subroutine at line 100 would be visited. *DAY* **=2** would cause **a** branch to line 200, and so on.

Associated keywords: G 0 T 0, G 0 S U B

ON BREAK GOSUB

ON BREAK **GOSUB** ‹**line number›**

```
10 ON BREAK GOSUB 40
20 PRINT "program running"
30 GOTO 20
40 CLS
50 PRINT "pressing [ESC] twice calls GOSUB
 routine"
60 FOR t=l TO 2000:NEXT
70 RETURN
```

COMMAND: Calls a subroutine on breaking from program execution by pressing **[ESC]** twice.

Associatedkeywords: ON *BREAK STOP, RETURN*

ON BREAK STOP

ON BREAK STOP

```
10 ON BREAK GOSUB 40
20 PRINT "program running"
30 GOTO 20
40 CLS
50 PRINT "pressing [ESC] twice calls GOSUB
 routine"
60 FOR t=1 TO 2000:NEXT
65 ON BREAK STOP
70 RETURN
```

COMMAND: When issued in an ON *BREAK* interrupt subroutine, ON *BREAK STOP* disables the trap, but has no other immediate effect. In the above program, which includes ON *BREAK STOP*, the ON *BREAK* GOSUB trap will only operate once.

Associatedkeywords: ON *BREAK* GOSUB

ON ERROR GOTO

ON ERROR GOTO ‹line number›

```
10 ON ERROR GOTO 80
20 CLS
30 PRINT"if there is an error, I would"
40 PRINT"like the program listed, so that"
50 PRINT"I can see where I went wrong"
60 FOR t=1 TO 4000:NEXT
70 GOTO 200
80 CLS:PRINT"THERE IS AN ERROR IN LINE";ERL:PRINT
90 LIST
```

COMMAND: Go to a specified line number in the program on detecting an error. In this example, an error will be found in line 70.

Associated keywords: ERR, ERL, RESUME

ON SQ GOSUB

ON SQ (‹channel›) GOSUB line number,

ON SQ (2) GOSUB *2000*

COMMAND: Enable an interrupt for when there is a free slot in the given sound queue. The ‹channel› is an integer expression yielding one of the values:

1: for channel A
2: for channel B
4: for channel C

Associated keywords: SOUND, SQ

OPENIN

0 **PENIN** ‹filename›

100 **OPENIN "!** INFORMATION"

COMMAND: Opens an input file from cassette which contains information for use in the current program in the computer's memory.

If the first character in the ‹**file name**› is **!** then the displayed cassette processing messages are suppressed. The program reads in the first block from the cassette, ready for processing.

Associated keywords: C **LOSEIN, OPENOUT**

OPENOUT

OPENOUT ‹filename›

OPENOUT"! FACTS"

COMMAND: Opens an output file onto cassette for use with the current program in the computer's memory. If the first character in the ‹**file name**› is **!** then the displayed cassette processing messages are suppressed. The program creates the first block of data, in the file with the given name. Each block consists of up to 2048 bytes of data.

NB A NEW command will abandon any open file buffered, and data will be lost.

Associatedkeywords: **CLOSEOUT, OPENIN**

ORIGIN

ORIGIN ‹x›,‹y›[,‹left›,‹right›,‹top›,‹bottom›]

```
10 CLS:BORDER 13
20 ORIGIN 0,0,50,590,350,50
30 DRAW 540,350
40 GOT0 20
```

COMMAND: Determines the start point for the graphics cursor. The [optional part] of the command contains the instructions to set a new graphics window, which will be operational in all screen modes due the pixel addressing technique employed.

The 0 R I G I N is the point with co-ordinates 0,0 (co-ordinates grow up and right).

If any of the window edges are specified to a position that is off the screen, they are assumed to represent the furthest 'visible' position in the given direction.

Associated keywords: W I N D 0 W

OUT

OUT ‹port number›, ‹integer expression,

OUT &F8F4,10

COMMAND: Sends the value in the ‹integer expression, (which must lie in the range 0....255) to the port specified in the port number> by its address.

Associated keywords: I N P, W A I T

PAPER

PAPER [#stream expression ,]‹masked ink›

```
10 MODE 0
20 FOR p=0 TO 15
30 PAPER p:CLS
40 PEN 15-p
50 LOCATE 6,12:PRINT "PAPER"p
60 FOR t=1 TO 500: NEXT t
70 NEXT p
```

COMMAND: Sets the background ink for characters. When characters are written to the text screen, the character cell is filled with the ***PAPER*** ink before the character is written - unless the transparent mode has been selected.

For PAPER/MODE/colour correlation, see Table 2, page F3.4.

Associated keywords: INK, WINDOW, PEN

PEEK

PEEK ([address expression])

```
10 MODE 2
20 INK 1,0: INK 0,12: BORDER 12
30 INPUT "Start address for examination";first
40 INPUT "End address for examination";last
50 FOR n= first TO last
60 VALUE$=HEX$(PEEK(n),2)
70 PRINT VALUES;
80 PRINT" at ";HEX$(n,4),
90 NEXT
```

FUNCTION: Examine the contents of a memory location. The above 'utility' program allows you to browse through the RAM of the CPC464. It reads the ***RAM*** under the lower (&0000-&3FFF) and upper (&C000-&FFFF) ROM - not the ROM.

Associated keywords: POKE

PEN

PEN [#&ream expression) ,]‹masked ink›

```
PEN 1,2
```

COMMAND: **PEN** sets the ink to be used when drawing at the given screen stream, defaulting to screen #0.

Associated keywords: *INK, PAPER*

PI

PI

```
PRINT PI
3.14159265

10 REM Perspective drawing
20 MODE 2
30 RAD
40 INK 1,0
50 INK 0,12
60 BORDER 9
70 FOR N= 1 TO 200
80 ORIGIN 420,0
90 DRAW 0,200
100 REM draw angles from vanishing point
110 DRAW 30*N*SIN(N*PI/4),(SIN(PI/2))*N*SIN(N)
120 NEXT
130 MOVE 0,200
140 DRAWR 0,50
150 DRAWR 40,0
160 WINDOW 1,40,1,10
170 PRINT"Now you can finish the
 Hangman program!"
```

FUNCTION: The value of the ratio between the circumference and the diameter of a circle. It is used extensively in graphics routines such as the one listed above.

Associated keywords: DEG, RAD

PLOT

P LOT ‹x co-ordinate›, ‹y co-ordinate›[,‹masked ink›]

```
10 MODE 2:PRINT "Enter 4 numbers,
 separated by commas":PRINT
20 PRINT "Enter X origin (0-639),
 Y origin (0-399), radius and
  angle to step":INPUT x,y,r,s
30 ORIGIN x,y
40 FOR angle = 1 to 360 STEP s
50 XPOINT = r*COS(angle)
60  YPOINT = r*SIN(angle)
70  PLOT XPOINT,YPOINT
74 REM MOVE 0,0
75 REM DRAW XPOINT,YPOINT
80 NEXT
```

COMMAND: Try 320,200,20,1 as your first response. P L 0 T is the same as M 0 V E, except that the pixel at the destination is written. If you un-R *EM* line 75 above and *R E M* line 70 to make it inoperative, you will see the difference. (Un-R EM line 74 to fill the circle).

Note that the process tills in the outline of the circle by repeated running around the perimeter. Remember that this program has not reset the RADian mode of angular calculation, so the 'angle' in each step is considerably more than one degree. Enter the command 2 5 DEG and run again.

Associated keywords: *DRAW, DRAWR, PLOT, PLOTR, MOVE, MOVER,* ORIGIN, *TEST, TESTR, XPOS, YPOS*

PLOTR

P LOT R ‹x co-ordinate>, ‹y co-ordinate›[,‹masked ink›]

```
5 DEG
10 MODE 2:PRINT "Enter 4 numbers,
 separated by commas":PRINT
20 PRINT "Enter X origin (0-639),
 Y origin (0-399), radius and
  angle to step":INPUT x,y,r,s
30 ORIGIN x,y
40 FOR angle = 1 to 360 STEP s
50 XPOINT = r*COS(angle)
60 YPOINT = r*SIN(angle)
70 PLOTR XPOINT, YPOINT
80 NEXT:GOTO 40
```

COMMAND: Try 320,0,2,1 in reponse. *P LOT R* is the same as DR A W R, *except* that only the pixel at the destination is written.

Associated keywords: *DRAW, DRAWR, PLOT, PLOTR, MOVE, MOVER,* ORIGIN, *TEST, TESTR, XPOS, YPOS*

POKE

POKE (address expression, <integer expression>

POKE &00FF,10

COMMAND: Provides direct access to the machine memory, and loads the ‹integer expression> in the range 0....255 directly at the address specified. Not to be used by the unwary.

Associated keywords: PEEK

POS

POS (# stream expression)

PRINT POS(#0)
1

FUNCTION: Establishes the current position for a given stream. In this instance there is no default for stream expression,, and omitting it will result in a Syntax error message.

Screen: Returns the current 'X' co-ordinate of the text cursor, relative to the current window origin. The top left corner is represented as 1,1.

Printer: Returns the carriage position, where 1 is the left margin. All characters with ASCII reference numbers greater than &1F (31) are included.

Cassette output stream: As for the printer.

Associated keywords: VPOS

PRINT

PRINT [#<stream expression ,][‹print list›][‹USING clause›][‹separator›]

PRINT #0,"abc"

COMMAND: For a full explanation of PRINT, see page 54 of this chapter.

Associated keywords: USING, TAB, SPC

RAD

RAD

RAD

COMMAND: Set Radians Mode

Associatedkeywords: **DEG, SIN, COS, TAN, ATN**

RANDOMIZE

RAND OM I Z E [(numeric expression)]

```
10 RANDOMIZE 23
20 PRINT RND(6)
```

COMMAND: BASIC's random number generator produces a pseudo random sequence in which each number depends on the previous number • starting from a given number, the sequence is always the same. RAN D 0 M I Z E sets a new initial value for the random number generator, either to a given value, or to a value entered by the operator. R A N D 0 M I Z E T I M E will produce a sequence that will be difficult to repeat.

Associated keywords: R N D

READ

R E A D list of:‹variable›

```
10 FOR X=1 TO 4
20 READ N$
30 PRINT N$
40 DATA ADAM,DANNY,JAMIE,RICHARD
50 NEXT
```

COMMAND : *R E A D* fetches data from the list of constants supplied in the corresponding *DA TA* statements and assigns it to variables, automatically stepping to the next item in the data statement. R E S T 0 R E will return the pointer to the beginning of the D A TA statement. See the DA TA keyword.

Associated keywords: DA T A, R E S T 0 R E

RELEASE

RELEASE sound channels

RELEASE 4

COMMAND: When a sound is placed on a sound queue it may include a 'hold' state. If any of the channels specified in this channel are in 'hold' state, then they are released, The expression to identify the sound channel is 'bit significant': A= bit 0, B= bitl, C= bit2. Thus 4 (binary 0100) releases Queue C.

Associated keywords: S 0 U N D

REM

REM rest of line›

```
10 REM Intergalatic Hyperspace Monster
   Invaders   Deathchase by AMSOFT
20 REM Copyright AMSOFT 1984
```

COMMAND: Used to place R EMarks *or* R EMinders in programs without affecting the program operation in any way. The : line separator is also ignored, everything from the *REM* to the line end is ignored. A single quote character ' in a line (not part of a string expression,) is equivalent to : *REM*, other than in a *DATA* command, where it is treated as part of an unquoted string.

REMAIN

REMAIN ((integer expression,)

```
REMAIN (3)
PRINT #6,REMAIN(0);
```

FUNCTION: Disables the specified delay timer (in the range 0....3).
Reads the remaining count from the delay timer. Zero igreturned if the delay timer was not enabled.

Associated keywords: AFTER, EVERY

RENUM

R E NUM [‹new line number›][*,*[‹old line number›][*,*‹increment›]]

RENUM
RENUM 100,,100

COMMAND: Renumber program lines from the line specified, using the increment specified. The ‹new line number gives the first number for. the new sequence, defaulting to 10. The ‹old line numberidentifies where *R ENUM* is to commence, and assumes the first program line if omitted. The (increment sets the increment to use between the line numbers, again defaulting to 10.

R E N U M takes care of all G 0 S *U B, G 0 T 0* and other line calls. If all the specifiers are omitted from the command, the program is renumbered as if R E NUM 10 , ,10 were issued.

Line numbers are valid in the range 1....65535.

RESTORE

RESTORE [‹line number›]

RESTORE 300

```
10 FOR N=1 TO 6
20 READ A$
30 PRINT A$;" ";
40 DATA restored,data,can,be,read,again
50 NEXT
60 PRINT
70 RESTORE
80 GOTO 10
```

COMMAND: Restores the position of the reading pointer back to the beginning of the *DA TA* statement specified in the optional ‹line number,. Omitting line number) restores the position of the pointer back to the beginning of the first D AT A statement.

Associated keywords: R E A D, DA TA

RESUME

RESUME [‹line number›]
or *RESUME NEXT*

RESUME 300

COMMAND: When an error has been trapped by an 0 N E R R 0 R G 0 T 0 command, and has been processed, R E S U M E allows normal program execution to continue, the resuming line number being optionally specifiable. If not specified, the line in which the error has occurred is returned to. R E S UM E N E X T returns to the line immediately following the statement in which the error was detected.

Associatedkeywords: ON *ERROR GOT0*

RETURN

RETURN

RETURN

COMMAND: Signals the end of a subroutine. BASIC returns to continue processing at the point after the G 0 S U B which invoked it.

Associatedkeywords: **GOSUB,** *ON x GOSUB,ON SQ* **GOSUB,** *AFTER* n *GOSUB,EVERY n GOSUB,ON BREAKGOSUB*

RIGHT$

R I G H T **$ (** string expression,, (integer expression, **)**

```
10 CLS
20 A$ = "AMSTRAD"
30 B$ = RIGHT$(A$,3)
40 PRINT B$
RUN
[SCREEN CLEARS]
RAD
Ready
```

FUNCTION: Extracts the number of characters specified by the ‹**integer** expressions from the right of the string expression. If the string expression is shorter than the required length, the whole ‹**string** expression, is returned.

Associated keywords: **M** I *D* **$**, *L E F T* **$**

RND

R N D [**(** numeric expression) **)**]

```
10 RANDOMIZE 23
20 PRINT RND
```

FUNCTION: Fetches a random number, which may be the next in sequence, a repeat of the last one, or the first in a new sequence. The **RAN DOM** I Z E command in the above program ensures that *RN D* returns the same number each time

RN D **(** 0 **)** returns a copy of the last random number generated. Where numeric expression1 is negative, the number sequence generated is predictable.

Associated keywords: R A N D 0 M I Z E

ROUND

R 0 UN D (numeric **expression**›[, ‹**integer** expression])

```
10  x=0.123456789
20  FOR r=9 TO 0 STEP -1:PRINT r,ROUND(x,r):NEXT
25  x=123456789
30  FOR r=0 TO -9 STEP -1
40  PRINT r,ROUND (x,r)
50  NEXT
```

FUNCTION: Rounds ‹**numeric** expression) to a number of decimal places or power of ten specified in (integer expression). If the ‹**integer** expression> is less than zero, then value is rounded to give an absolute integer followed by a number of zeros determined by the <integer expression) before the decimal point.

Associatedkeywords: INT, FIX, CINT, *ABS*

RUN

R U N (string expression,

RUN "WELCOME"

COMMAND: Load a program from cassette and start executing it. If the string expression is empty "" BASIC attempts to load and execute the first file it encounters on the tape, If the first character of the string expression is ! then the displayed cassette processing messages are suppressed.

NB: BASIC effectively executes an implied NEW immediately a ‹**filename**› is read on the tape.

Associated keywords: L 0 A D

RUN

R U N [<line number,]

RUN 100

COMMAND: Starts executing the current program at the line specified, or from the beginning if no line is specified. All current program, user functions and variables are deleted from memory. D E F I N T, *D E F R E A L* and D E F S T R settings are reset. All cassette files are abandoned, and any buffered output is lost.

Associated keywords: L 0 A D

SAVE

SAVE ‹filename›[,file type›][,‹binary parameters1

```
SAVE "PROG",P
SAVE "BINARY",B,10000,16000,10003
```

COMMAND: Save the program in memory with name ‹filename›. Binary parameters comprise start address, ‹file length›[, (entry point›]

, A saves program in ASCII.
, P protects file.
,B saves an area of memory as a binary file - **e.g.** the screen.

Associatedkeywords: *LOAD, RUN* ‹filename›,MERGE,CHAIN, *CHAIN MERGE*

SGN

SGN (numeric expression)

```
10 INPUT "What's your current Bank BaLance";CASH
20 IF SGN(CASH) <1 GOTO 30 ELSE 40
30 PRINT "Oh dear, oh dear":END
40 PRINT"You've got more money than me"
```

FUNCTION: Determines the sign of the ‹numeric expression). Returns -1 if <numeric expression) is less than 0, returns 0 if ‹numeric expression) = 0, and returns 1 if (numeric expression) is greater than zero.

Associated keywords: A B S

SIN

SIN((numeric expression>)

```
PRINT SIN(PI/2)
 1
```

FUNCTION: Calculates the Real value for the Sine of ‹numeric expression), defaulting to the Radian measure mode unless otherwise declared by a *DEG* command.

Associatedkeywords: COS, *TAN, ATN, DEG, RAD*

SOUND

S O U N D channel status , tone period›[, <duration ›[, ‹volume›[, volume envelope)
[, tone envelope›[, noise period›]]]]]

```
SOUND 1,200,1000,7,0,0,1
```

COMMAND: The *SOUND* features of the CPC464 is one of the most complex extensions to BASIC and is introduced in chapter 6.

Associated keywords: E N V , E N T

SPACE$

S PA C E $ (integer expression)

```
SPACE$(190)
```

FUNCTION: Creates a string of spaces of the given length.

Associated keywords: P R I NT , S P C , T A B .

SPEED INK

SP E E D INK integer expression), integer expression

```
5 INK 0,9,12:INK 1,0,26
10 BORDER 12,9
20 SPEED INK 50,20
```

COMMAND: The I N K and BO R D E R commands allow two colours to be associated with each Ink, in which case the I N K alternates between the two colours. The first integer expression) specifies the time for the first I N K specified to be used, and the second integer expression sets the time for the second I N K. Times between colour changes are measured in units of 1/50 second. (50 Hz mains versions)
You must exercise careful judgement to avoid mesmeric effects when selecting colours and repeat rates!

Associated keywords: I N K , B 0 R D E R

SPEED KEY

SPEED KEY &art delay,, repeat period)

SPEED KEY 20,3

COMMAND: If held down continuously, the keys on the CPC464 auto repeat at the repeat period) after the given ‹start delay› period. The setting is made in 1/50 sec units, in the range 1....255. The default rate is set to 30,2

Very small start delays will interact with keyboard de-bounce routines. The actual speed at which the keyboard is 'read' by the software is not affected by this command.

Not all keys repeat, the KEY DEF commnd will allow the user to redefine the particular attributes of a given key.

Associated keywords: KEY DEF

SPEED WRITE

SPEED WRITE <integer expression,

SPEED WRITE 1

COMMAND: The cassette can be witten at either 2000 baud (where ‹integer expression is 1), or the default of 1000 baud (where the ‹integer expression, is 0). When loading a file from tape, the CPC464 automatically establishes the correct reading speed from information in the file software, thus it is not necessary for the user to specify.

When using cassette tape of uncertain data recording ability, the 1000 baud rate is recommended for maximum reliability, see the notes in Chapter 2 for further information.

Associated keywords: SAVE

SQ

S Q (‹channel›)

```
10 MODE 1
20 FOR n=20 TO 0 STEP-1
30 PRINT n;
40 SOUND 1,10+n,100,7
50 WHILE SQ(1)>127:WEND
60 NEXT
```

FUNCTION: The S Q function is used to check the number of free entries in the queue for a given channel, where channel A is 1, B is 2, and C is 3. The function determines whether the channel is active - and if not - why the entry at the head of the queue (if any) is waiting. The result is bit significant:

0,1,2 indicate the number of free entries in the queue
3,4,5 indicate the Rendezvous state at the head of the queue (if any)
6 is set if the head of the queue is held
7 is set if the channel is currently active

Associatedkeywords: *SOUND*, *ON SQ* GOSUB

SQR

S Q R (numeric expression)

```
PRINT SQR(9)
 3
```

FUNCTION: Returns the square root of <numeric expression,.
Associated keywords: P R I NT

STOP

STOP

```
300 IF n<0 THEN STOP
```

COMMAND: To stop execution of a program, but leave BASIC in a state where the **program can be restarted after** the S T 0 P command by using the C 0 NT command. This may be used to interrupt the program at a particular point when debugging.

Associated keywords: *C 0 N T , E N D*

STR$

S T R **$** (numeric expression))

```
PRINT STR$(&766)
 1894

PRINT STR$(&X1010100)
 84
```

FUNCTION: Converts the numeric expression) to a decimal string representation in the same form as used in the P R I N T command.

Associatedkeywords: *VAL*, *PRINT*, **HEX$,** BIN3

STRING$

S T R I N G **$** (integer expression, character specificier›)

```
PRINT STRING$(&16,"*")
**********
```

FUNCTION: Delivers a string expression) consisting of the specified character repeated a number of times.

Associated keywords: S PA C E **$**

SYMBOL

S Y MBO *L* character number, ‹list of:‹row›

```
5  MODE 2
10  SYMBOL AFTER 90
20  SYMBOL 93,&80,&40,&20,&10,&8,&4,&2,&1
30  FOR n=1 TO 2000
40  PRINT CHR$(93);
50  NEXT
60  GOTO 60
```

COMMAND: The S Y M BO L command redefines the representation of a given character that has first been specified in the S Y M BO L A F T E R command. The character number, is chosen from the available ASCII or other characters from the CPC464's standard character set, and the following entries define the new character on an 8x8 pixel matrix. A 0 in the row indicates the paper colour to be used, and a 1 indicates that the pixel is to be set to the current ink colour. Also see Appendices II and III. The example above produces a backslash that goes diagonally across the character cell, accessible by pressing the] key.

Associated keywords: S Y MB 0 L A F T E R

SYMBOL AFTER

SYMBOL AFTER ‹integer expression,

```
SYMBOL AFTER 90
```

COMMAND: The number of user definable characters is set by the SYMBOL AFTER command. The default setting is 240, giving 16 user defined characters. If the (integer expression, is 32, then all characters from 32 to 255 are redefinable.

Whenever a SYMBOL AFTER command is used, all user defined characters are reset to the default condition.

Associated keywords: SYMBOL

TAG

TAG [# stream expression,]

```
10 MODE 2
11 BORDER 9
14 INK 0,12
15 INK 1,0
20 FOR n=1 TO 100
30 MOVE 200+n,320+n
40 TAG
50 IF n<70 GOTO 60 ELSE 70
60 PRINT"Hello";:GOTO 80
70 PRINT" Farewell";
80 NEXT
90 GOTO 20
```

COMMAND: Text sent to a given stream may be redirected to be written at the graphics cursor position. This allows text and symbols to be mixed with graphics. The stream expression) defaults to 0 if omitted. The top left of the character cell is tagged to the graphics cursor, and non-printing control characters will display. In particular, 'new line' characters will display if the PRINT statement is not followed by a semi-colon ;

Associated keywords: TAG OFF

TAGOFF

TAG OFF[#‹stream expression]

```
TAGOFF #0
```

COMMAND: Cancels the TAG for a given stream, and sends the text to the previous text cursor position at the point at which TAG was invoked.

Associated keywords: TAG

TAN

TAN (numeric expression>)

```
PRINT TAN(45)
```

FUNCTION: Calculates the tangent for the angles given in numeric expression), which must be in the range -200,000....+200,000, defaulting to radian measure unless declared otherwise by a *DEG* command.

Associatedkeywords: COS, SIN, *ATN*, *DEG*, *RAD*

TEST

TEST (‹x co-ordinate), ‹y co-ordinate))

```
PRINT TEST(300,300)
```

FUNCTION: Reports the value of the ink currently at the specified graphics screen location.

Associated keywords: TESTR, *HOVE*, *MOVER*, *PLOT*, *PLOTR*, *DRAW*, *DRAWR*

TESTR

TESTR (‹x offset,, ‹y offset›)

```
TESTR(5,5)
```

FUNCTION:Moves the graphics cursor to the specified location and reports the value of the ink there.

Associated keywords:TEST, *MOVE*, *MOVER*, *PLOT*, *PLOTR*, *DRAW*, *DRAWR*

TIME

TIME

```
10 DATUM = INT(TIME/300)
20 TICKER=((TIME/300)-DATUM)
30 PRINT TICKER;
40 GOTO 20
```

FUNCTION: Reports the elapsed time since switch-on, excluding periods when reading or writing the cassette. The units of time are 11300th of a second.

TRON
TROFF

TRON

TROFF

TRON

COMMAND: BASIC includes the facility to trace the execution of a program, by reporting the number of each line in square brackets [1, just before it is executed. T R 0 N enables the feature, T R 0 FF turns it off.

Associated keywords: R U N

UNT

U N T (address expression,)

```
PRINT UNT(65535)
-1
```

FUNCTION: Converts an unsigned 16-bit integer in the range 0....65535. Returns an integer value in the range -32768....+32767.

Associatedkeywords: INT, FIX, CINT, *ROUND*

UPPER$

U P P E R $ (string expression>)

```
PRINT UPPER$("amstrad")
AMSTRAD
```

FUNCTION: Returns a new string expression the same as the input string expression) but in which all lower case characters are converted to upper case.

Associated keywords: L 0 W E R $

VAL

VA L (string expression,)

```
10 A$="7 is my lucky number"
20 PRINT VAL(A$)
```

FUNCTION: Extracts a numeric expression) from the beginning of the string expression,. The opposite of S T R $.

Associated keywords: S T R $

VPOS

V P O S (*#stream* expression,)

PRINT VPOS(#0)

FUNCTION: Returns the vertical position of the text cursor for the stream expression

Associated keywords: *P O S*

WAIT

WA I T ‹port number›,‹mask›[,‹inversion›]

WAIT &FF34,20,25

COMMAND: Suspends operation until a given I/O port returns a particular value in the range 0...255. BASIC loops whilst reading the I/O port. The value-read is Exclusive ORed with the (inversion and then A NDed with the ‹mask› until a non-zero result occurs. BASIC will get stuck in a WAI T loop if the required condition does not occur. If you type in the above example, you will have to fully reset the computer to escape.

Associated keywords: I N P, 0 U T

WEND

WEND

```
10 MODE 1:REM BASIC CLOCK TIME ROUTINE
20 INPUT "Enter the current hour, minute, and
   second (h,m,s)";hour,minute,second
30 CLS:datum = INT(TIME/300)
40 WHILE hour<13
50 WHILE minute<60
60 WHILE tick<60
70 tick=(INT(TIME/300)-datum)+second
80 LOCATE 70,4
90 PRINT #0,USING"##  ";hour,minute,tick
100 WEND
110 tick=0
115 second=0
120 minute=minute+1
130 GOTO 30
140 WEND
150 minute=0
160 hour=hour+1
170 WEND
180 hour=1
190 GOTO 40
```

COMMAND: The W H I L E / W EN D loop repeatedly executes a body of program until a given condition is true. The illustration here uses the W H I L E/W EN D loop to demonstrate the elegance of programs constructed using this approach. The body of the features of a variety of different clocks can now be added. The ***WEND*** command terminates the W H I L E loop..

Associated keywords: W H I L E

WHILE

W H I L E logical expression)

WHILE DAY <0

see example above

COMMAND: A W H I L E loop repeatedly executes a body of program until a given condition is true. The W H I L E command defines the head of the loop, and gives the condition. The W E N D command terminates the W H I L E loop.

Associated keywords: W E N D

WIDTH

WIDTH ‹integer expression

WIDTH 86

COMMAND: Tells BASIC how wide the printer is in characters, this information allows BASIC to insert carriage returns as required when printing.

Associated keywords: PRINT, POS

WINDOW

WINDOW [#‹stream expression ,]‹left›, ‹right›,‹top›,‹bottom›

```
10 MODE 1
20 BORDER 6
30 WINDOW 10,30,7,18
40 PAPER 2:PEN 3
50 CLS
60 PRINT CHR$(143);CHR$(242);"THIS IS LOCATION"
70 PRINT "1,1 IN TEXT WINDOW"
80 GOTO 80
```

COMMAND: Sets a text window for a given screen stream.

Associated keywords: ORIGIN

WINDOW SWAP

WINDOW SWAP stream expression, stream expression>

```
WINDOW SWAP 0,2
```

COMMAND: Exchanges the text windows. For example, BASIC messages sent to stream #0 may be swapped with another window to highlight aspects of program development and operation.

Associatedkeywords: *WINDOW, PEN, PAPER, TAG*

WRITE

WRITE [#stream expression),][‹write list›]

```
WRITE #2,"HELLO",4,5
"HELLO",4,5
```

COMMAND: Prints the values of a number of expressions to the given stream, separating them by commas and enclosing strings in double quotes. Used mainly for outputting data to cassette files.

XPOS

XPOS

PRINT XPOS

FUNCTION: Establishes the horizontal position of the graphics cursor.

Associated keywords: **YPOS,** MOVE, MOVER, ORIGIN

YPOS

YPOS

PRINT YPOS

FUNCTION: Establishes the vertical position of the graphics cursor.

Associatedkeywords: **XPOS, MOVE, MOVER,** ORIGIN

ZONE

ZONE {integer expression)

```
10 PRINT 1,2,3
20 ZONE 19
30 PRINT 4,5,6
```

COMMAND: Changes the width of the Print Zone used in **PRINT,** from the default value of 13 to a new value in the range **1....255.** Reset by **NEW,** LOAD, CHAIN and RUN "‹file name›" commands.

Associatedkeywords: **PRINT, WIDTH**

PRINT

PRINT [#‹stream expression) ,][‹print list›][‹ USING clause›][‹separator›]

‹print list, is ‹print item›[‹separator›‹print list›]
‹print item› is (expression
or SPC(<integer expression))
TAB (integer expression))

Write data to a cassette file.

```
10 OPENOUT "DATA"
20  PRINT  #9, "Hello"
30 CLOSEOUT
```

Write data to a line printer

```
10 BOBSSALARY ❑  23000*PI
20 PRINT #8, USING "######.##";BOBSSALARY
30 PRINT #0, BOBSSALARY
RUN
 72256.6311
Ready

[Meanwhile, on the PRINTER.....]
 72256.63
```

COMMAND: Print data at stream expression using the format specified. See table for format characters.

Where no formats are specified, BASIC prints in 'free format', where a comma following the ‹print item, will send the printed item to start at the next print ZONE (default to **13).** A semi-colon simply separates the expressions.

SPC(‹integer expression)) prints the given number of spaces, defaulting to zero if the numeric expression) is less than 1. S PC does not require to be terminated by a comma or semi-colon - a semi-colon is assumed at all times.

TAB(‹integer expression,) prints spaces to move to the given print position, if less than 1, then 1 is assumed.

If the required position is equal to or exceeds the current position, spaces are printed to reach the required position - if less, then a carriage return is sent, followed by spaces to reach the required position. *TAB* does not require termination by a comma or semi-colon.

(Continued. . . . . ..)

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 number will have +	###+
	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	#,##↑↑↑↑

STRING

!			First character only	
\‹spaces›\			Number of spaces specified, plus a leading and trailing space character field	i \
&			Variable length field	&