CENTURY COMMUNICATIONS

AMSTRAD

MICRO GUIDE

A Quick Reference Guide to the BASIC AND SYSTEM OPERATIONS of the

AMSTRAD

This edition published in 1986 by Book Club Associates
by arrangement with Century Communications Limited.

Grateful acknowledgement is hereby given to
Amstrad Limited.

Printed in Great Britain

MICROGUIDE
FOR THE
AMSTRAD

Professor Peter Morse
Brian Hancock

GUILD PUBLISHING London

CONTENTS

1 CONVENTIONS AND TERMS USED

& (ampersand) is used in this book and on the computer to prefix a *hexadecimal* (hex) number.

Baud Communication speed in bits per second.

Default Value assumed when none is specified by user.

Expression Any legal combination of constants, variables, functions and arithmetic operators.

Line number Any number between 1 and 63999 at the beginning of a line which serves to identify the line as a BASIC statement.

List A one-dimensional array.

Number A positive or negative decimal quantity which is significant to nine digits and whose magnitude lies between 2.93873588 E-39 and +1.70141183 E+38. Over-large numbers produce an overflow error, numbers smaller than the lower limit are treated as zero.

Numeric Variables Two types of name are used for these:

> ***Real variables*** may be represented by any set of alphanumeric characters provided that they start with a letter, do not include spaces or quotes and do not start with a BASIC keyword. They can take the full range of legal numeric values.
>
> ***Integer variables*** follow the same rules and are distinguished by having a % sign as the last character of the name (eg. AREA1%). They can only hold whole numbers between -32768 and +32767.

String A sequence of ASCII characters.

String Variable Used to store strings. Names as for numeric variables but must end with $ (eg. N$ or NAME$).

Substring Any set of consecutive characters taken from a parent string. CON is a substring of CONVINCE.

Table A two- or multi-dimensional array.

Abbreviations and Symbols

<>	statement
[ ]	optional items
add	address (0–65535 — &0000–&FFFF)
n	numeric expression
i	integer expression
st	stream number
s	string
x,y,xr,yr	x and y co-ordinates x = 0–640 and y = 0–400
ln	line number (0–63999)
t	timer number
v	variable
le	logical expression
di	data item (string or number)

Ink colours There are 27 ink colours as shown in this table:

Ink Number	*Colour*	*Ink Number*	*Colour*
0	black	14	pastel blue
1	blue	15	orange
2	bright blue	16	pink
3	red	17	pastel magenta
4	magenta	18	bright green
5	mauve	19	sea green
6	bright red	20	bright cyan
7	purple	21	lime green
8	bright magenta	22	pastel green
9	green	23	pastel cyan
10	cyan	24	bright yellow
11	sky blue	25	pastel yellow
12	yellow	26	bright white
13	white		

2 CONTROL KEYS

CURSOR KEYS △ ▽ < > move cursor up, down, left and right.

[CAPS LOCK] causes keys pressed to produce capital letters.

[CLR] deletes the character under the cursor during EDIT.

[COPY] is used with cursor keys to edit program lines.

[CTRL] In the CP/M environment a variety of operations are carried out using **[CTRL]** and another key simultaneously. Eg. [CTRL] S halts the screen output from CP/M and [CTRL] P toggles printer output on and off.

[CTRL] [SHIFT] [ESC] resets the computer. Any program currently in memory is lost.

[DEL] is used to delete characters to the left of the cursor.

[ESC] Pressing it once will cause a temporary pause in the current process, if pressed twice the computer will abandon the process.

[ENTER] causes a screen command line to be interpreted.

[CTRL] [ENTER] (using the little [ENTER] key on the numeric pad) types the command RUN" " and 'enters' it on tape-based systems.

3 OPERATING SYSTEM COMMANDS

AUTO [ln,i]
Automatically generates line numbers starting at line **ln** with increment **i** between line numbers. Use [ESC] to leave AUTO mode. Default value for ln and i is 10.
Example:
AUTO 100,5 generates line numbers 100, 105, 110 . . .

CONT
CONTinues program execution interrupted either by [ESC] [ESC] or as a result of **STOP** within the program.

DELETE[ln] [-ln]
Deletes the specified line(s) from program.
Examples:
DELETE deletes the whole program
DELETE -100 deletes up to and including line 100

EDIT ln
Displays the specified program line to be edited. Use cursor keys to move to the required position on the line.

LIST [ln] [-ln] [,#st]
Produces a listing of the program lines specified on stream **st**.
Example:
LIST -100,#8 lists up to line 100 on the printer

NEW
Clears computer memory, deleting program and variables.

RENUM [new ln] [[,old ln] [,i]]
Renumbers program lines. Default values number 10, 20, 30 etc.
Examples:
RENUM,,5 renumbers the program as 10, 15, 20, . . .
RENUM 50,10,5 renumbers the program as 50, 55, 60, . . .

RUN se
Loads a program from disk and executes it. Protected programs can only be executed using RUN without a LOAD.
Example:
RUN"EX1"

RUN [ln]
Executes the program in memory from the specified line number. **ln** defaults to lowest line number.
Example:
RUN 100

TROFF
Turns off the program flow trace *(see TRON below)*.

TRON
Turns on the program flow trace for debugging. Causes the line number of each statement executed to be displayed.

WIDTH i
Specifies maximum **i** characters to be printed on a line when outputing to a printer. Default value is 132.

4 ARITHMETIC & LOGIC OPERATIONS

Arithmetic operators

+	addition	/	division
-	subtraction	↑	exponentiation
*	multiplication	\	integer division
MOD	gives remainder after division		

Order of precedence

Group 1	( ), functions, NOT
Group 2	↑
Group 3	*, /, MOD
Group 4	+, -
Group 5	=, <>, <, >, <=, >=
Group 6	AND
Group 7	OR, XOR

Relational operators

=	equal to
<>	not equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

Logical operators

AND

(condition 1) AND (condition 2) is only true when both conditions are true. Also used as a bitwise operator for binary numbers.

Examples:

```
IF A>1 AND A<11 THEN PRINT "A IS BETWEEN 1 AND 10"
PRINT 85 AND 28 prints 20
```

85 in binary is	0	1	0	1	0	1	0	1
28 in binary is	0	0	0	1	1	1	0	0
85 AND 28 = 20	0	0	0	1	0	1	0	0

NOT n

Gives logical reverse of expression. Also used as bitwise operator.

Examples:

```
IF NOT (A=B) THEN PRINT "A AND B ARE NOT EQUAL"
PRINT NOT 5 prints -6
```

OR

(condition 1) OR (condition 2) is true when either or both conditions are true. Also used as bitwise operator.

Examples:

```
IF X>0 OR Y>0 THEN PRINT "ONE OR BOTH IS POSITIVE"
PRINT 85 OR 28 prints 93
```

XOR

(condition 1) XOR (condition 2) is true when either (condition 1) or (condition 2) is true but not both. Also used as bitwise operator.

Examples:

```
IF X=0 XOR Y=0 THEN PRINT "ONLY ONE OF THEM IS ZERO"
PRINT 85 XOR 28 prints 73
```

5 GENERAL BASIC COMMANDS

AFTER i[,t] GOSUB ln

Waits for i/50 seconds and then jumps to the subroutine at line **ln**.

CALL add [,list of :parameters]
Allows an externally developed subroutine to be called by BASIC.
Example:
CALL 0 resets the computer completely

CLEAR
Clears all variables from memory, leaving the program in memory unchanged. All open files are abandoned.

CLEAR INPUT
Clears the keyboard buffer of any characters.

CLS [#st]
Clears the given screen stream to its paper colour.

CURSOR [system switch] [,user switch]
Turns cursor on or off. Both **switches** must be 0 (off) or 1 (on).

DATA di1,di2,di3, . . .
Provides data for a READ statement.

DEF FN function-name [dummy variable] = e
Defines a user-specified function *(see FN)*.

DEFINT letter range
Sets the default for variable(s) with the specified first letter(s) to integer. The letter range could be an inclusive range A–Z.
Example:
```
10 DEFINT F,S . . . . (or 10 DEFINT A-Z)
20 FIRST = 111.11 : SECOND = 22.2
30 PRINT FIRST, SECOND
prints 111              22
```

DEFSTR letter range
Forces all variable(s) starting with the specified letter(s) to be string variables. The $ does not need to be added to DEFSTR variable names.
Example:
10 DEFSTR N sets all variables starting with letter N as strings

DEG
Sets the mode of calculation to degrees (default is radians).

DI
Disables interrupts (but not [ESC]) until re-enabled by EI command or by RETURN at end of an interrupt service routine.

DIM v[$] (i1[,i2])
Specifies storage space to be allocated for list or table **v[$]**. **i1** is number of rows, **i2** is number of columns. If a list or a table is not specified by DIM, **i1** and **12** default to 10.

EI
Enable interrupts which have been disabled by DI.

END
Indicates end of program.

ERASE v[$(i1[,i2])
Clears the contents of an array that is no longer required.

ERROR i
Returns the error message whose error code number is **i**.

EVERY i[,t] GOSUB ln
BASIC branches to the subroutine at line **ln** every **i/50** seconds.

FN function-name [(variable)]
Calls up the specified user-defined function.
Example:
```
10 DEF FNMULT(X)=X*3
20 Y=25 : PRINT FNMULT(Y)
```
prints 75

FOR v = i1 TO i2 [STEP i3]
Used with NEXT to specify loop limits. The loop is executed over range specified by **i1** and **i2**. **STEP** defaults to 1.
Example:
```
10 FOR I=10 TO 40 STEP 5
20 PRINT I; : NEXT I
```
prints 10 15 20 25 30 35 40

GOSUB ln
Transfers control to subroutine at line **ln**. A RETURN statement ends subroutine and returns control to the line following GOSUB.
Example:
```
100  GOSUB 1000
110  END
1000 REM SUBROUTINE
     . . . .
1100 RETURN
```

GOTO ln
Transfers control to line **ln**. Can also be used to start execution without destroying any pre-set variable values.

IF e THEN <> [ELSE <>]
The expression is evaluated and, if true, control is transferred to the statement following THEN, otherwise (if false) to the statement following ELSE. If ELSE portion is omitted, control is passed to the next line instead.
Example:
```
IF X=0 THEN PRINT "ZERO" ELSE PRINT "NON-ZERO"
```

INK ink,c1[,c2]
Describes ink colour(s) (0–15) to be assigned for use by PEN or PAPER commands. **c1** (0–26) specifies the colour, if **c2** is included, then **ink** alternates between colours **c1** and **c2** at rate specified by the SPEED INK command. *(see table in section 1).*
Example:
```
INK 0,13,1 : SPEED INK 50,50
```

INPUT [#st,] ["prompt";] list of v[$]
Allows data to be entered from the specified stream (default 0).
Examples:
INPUT A, B, C inputs data from keyboard to variables A, B and C
INPUT #9, A$, B, C inputs data from disk file to A$, B and C

KEY expansion token number, se
Used to assign **se** to the specified function key. The **expansion token number** must be in the range 0–31 or 128–159.
Example:
```
KEY 138, "RUN"+CHR$(13)
```
redefines the full-stop key on the numeric pad

KEY DEF number, repeat[, normal[, shifted[control]]]
Causes the string expression assigned to an expansion key to be returned by another, key (specified by **number**).
Example:
KEY DEF 46, 1, 63 redefines the N key as ASCII 63, a ?.

LET v[$] = e
Assigns value of **e** to the variable **v[$]**. The word LET may be omitted.

LINE INPUT [#st,] ["prompt";]v$
Accepts input of up to 255 characters, ending with [ENTER], from the specified stream.

LOCATE [#st,]x,y
Positions the text cursor at the position specified by **x** and **y**.
Example:
LOCATE 10, 20: PRINT "THIS IS COLUMN 10 ROW 20"

MEMORY add
Allocates the amount of memory to be used by BASIC by setting the address of the highest byte it may use.

MID$ (v$,i1[,i2]) = se
Inserts the string expression **se** into the string specified by **v$**, starting at position **i1** in **v$**. **i2** gives the length of **se**.
Example:
A$="ABCDEIJH": MID$(A$, 6, 2)="FG": PRINT A$
prints ABCDEFGH

MODE i
Selects the screen mode (0, 1, or 2), clears screen to INK value 0 and resets all text and graphics windows to the whole screen.

NEXT [v[,v]]
Terminates a FOR . . . NEXT loop if limit of loop is reached. If more than one **v** used, loops are completed in left-to-right order.

ON BREAK CONT
Prevents the interruption of program execution by the [ESC] key.

ON BREAK GOSUB ln
Passes control to subroutine at line **ln** when [ESC] [ESC] pressed.

ON BREAK STOP
Restores normal function of [ESC] key during program execution.

ON ERROR GOTO ln
Passes the control to line **ln** if an error is detected in the program.

ON ERROR GOTO 0
Turns off the error trap, and restores normal error processing.

ON e GOSUB ln and **ON e GOTO ln**
Allows several possible transfers of control to a line or subroutine, depending on the value of **e**.
Example:
ON X GOTO 100, 200
passes control to line 100 if X = 1, and to line 200 if X = 2

OUT add,i
Outputs the value of **i** (0–255) to the I/O address **add**.

PAPER [#st,] [ink]
Sets character background **ink** colour (0–15). MODE dependent.

PEN [#st,] [ink] [,background mode]
Sets **ink** (0–15) to be used when writing to specified screen **st**. Background mode can be either 1 (transparent) or 0 (opaque).

POKE add,i
Alters contents of memory location **add** to value **i** (1–255).

PRINT [#st,] ["prompt";] [v[$]]
Items to be printed may be separated by commas (,), causing each to be printed in next print zone *(see ZONE below)*; or by semicolon (;), causing each to be printed without additional spacing. Abbreviates to ?.

Example:
PRINT 5,6;"HELLO" prints 5 6HELLO

PRINT[#st;]SPC (i)
Moves print position i spaces to right before printing output.

PRINT[#st] TAB (i)
Moves the print position to column i before printing output.

PRINT [#st,]USING "format";v
Enables automatic formatting of printed output, useful for producing tables, forms and accounts. **format** is given as a string constant or a string variable (maximum 20 characters) containing instructions as to how the output is to be printed.

The format characters used with PRINT USING*:*

#	formats numbers "###" *with* 147.2 *gives* 147
.	decimal point position "##.##" *with* 34.678 *gives* 34.68
,	displays comma to the left of every third character "######,.#" *with* 123456 *gives* 123,456.0
**	fills leading spaces with asterisks "**###.###" *with* 1.47 *gives* ****1.470
££	prints £ sign before the first digit "££####.##" *with* 12.689 *gives* £12.69
$$	prints $ sign before the first digit "$$####.#" *with* 12.689 *gives* $12.7
**£	places asterisks before the £ sign "**£#####.##" *with* 12.689 *gives* ****£12.69
**$	places asterisks before the $ sign "**$#####.##" *with* 12.689 *gives* *****$12.69
+	In first position prints + or - before the number "-##.##" *with* -1.269 *gives* -1.27
+	in last position prints + or - after the number "##.##-" *with* -1.269 *gives* 1.27-
-	in last position prints - sign after a negative number "##.##-" *with* 1.269 *gives* 1.27-
↑↑↑↑	prints in exponent format "##.###↑↑↑↑" *with* 12.681 *gives* 1.269E-01
!	prints only the first character "!" *with* "CREDIT" *gives* C
\ \	prints string to length of number of spaces + 2 "\ \" *with* "CREDIT" *gives* CRED

& SPACE prints the entire string
"& " *with* "CREDIT" *gives* CREDIT

RAD
Selects radian mode for calculations. This is the default mode.

RANDOMIZE [n]
Randomizes the number seed specified by **n**. If **n** omitted, the prompt Random number seed? appears: enter a value.

READ v[$] [,v[$]]
Assigns the data in a DATA statement to the specified variable(s). A DATA statement must be present somewhere in the program.
Example:

```
10 RESTORE 100 : READ AS , B , C , D%
100 DATA SMITH , 3 . 5 , 80 . 0 , 25
```

REM (or ' – single quote)
Inserts comment lines in the program. Everything after REM up to the end of the line is ignored by the BASIC interpreter.

RESTORE [ln]
Resets DATA read pointer to selected line number *(see READ)*. RESTORE alone sets data pointer to first data item in the program.

RESUME [ln]
Resumes execution of a program after an error has been trapped and processed by ON ERROR GOTO. If **ln** is omitted, the program re-starts execution from the line in which the error was trapped.

RESUME NEXT
Re-executes the program from the line following the line in which the error was trapped (by an ON ERROR ... statement).

RETURN
Terminates a subroutine and returns control to the line following the GOSUB call *(See GOSUB)*.

SPEED INK i1,i2
Sets rate of alternation between two ink colours specified in INK or BORDER commands. **i1** gives the time period (i/50 seconds) for the first colour, **i2** is the time period for the second colour.

SPEED KEY i1,i2
Sets the rate of keyboard auto repeat. The parameter **i1** gives the time (i/50) seconds before auto repeat starts. The parameter **i2** sets the time delay between repeats of a key.

STEP
Increment (or decrement) step values of a loop-counting variable in a FOR . . . NEXT loop *(see FOR)*.

STOP
Breaks program execution at line containing the STOP statement. The message BREAK in is output with the line number.

SYMBOL character number, list of variables
Allows redefinition of a character. Must be used after a SYMBOL AFTER command *(see below)*.
Example:

```
10  SYMBOL AFTER 68
20  ROW1=60:    REM 00111100
30  ROW2=126:   REM 01111110
```

```
40  ROW3=253:   REM 11111100
50  ROW4=248:   REM 11111000
60  ROW5=253:   REM 11111100
70  ROW6=126:   REM 01111110
80  ROW7=60:    REM 00111100
90  ROW8=0:     REM 00000000
100 SYMBOL 68,ROW1,ROW2,ROW3,ROW4,ROW5,
    ROW6,ROW7,ROW8:PRINT CHR$(68)
```

SYMBOL AFTER i
Specifies number of allowable user-defined characters. i specifies that all the characters numbered i to 255 may be redefined.

THEN *(see IF)*

TO *(see FOR)*

UNT (add)
Returns an integer(− 32768 to 32767) which is the two's complement of **add**.
Example:
PRINT UNT (&FF66) prints -154

USING *(see PRINT USING)*

WAIT add, i1[,i2]
Waits until the I/O port at **add** returns a value (0–255). The value returned is XORed with **i2** and then ANDed with **i1**. This is repeated until a non-zero result occurs.

WEND *(see WHILE below)*

WHILE le
Repeats execution of a section of program as long as specified condition is true. WHILE indicates start of loop, WEND the end.
Example:

```
10 WHILE A<>0
20 INPUT A
30 PRINT "NON-ZERO"
40 WEND:PRINT "ZERO"
```

WINDOW [#st,] left, right, top, bottom
Defines dimensions of a window on specified screen stream.
Example:

```
100 WINDOW 3,7,5,19:CLS:LIST
```

WINDOW SWAP #st1, #st2
Swaps the text window specified by **#st1** with **#st2**.

WRITE [#st,]v[$],v[$]]
Writes the values of the specified variable to the specified stream.
Example:

```
10 OPENOUT "EXPENSE"
20 INPUT A$,A
30 WRITE #9,A$,A:CLOSEOUT: REM Write to tape
```

ZONE i
Changes the width of the print zone *(see PRINT)*. Default is 13.

6 BASIC NUMERIC FUNCTIONS

ABS (n)
Returns the absolute value of **n** by ignoring the sign value.
Example:
PRINT ABS (-3.5) prints 3.5

ATN (n)
Returns the arctangent (ie. $\tan^{-1}$) of **n**.

BIN$ (i1 [,i2])
Returns binary representation of **i1** between –32768 and 65535. The number of binary digits (0s and 1s) is specified by **i2** (0–16).
Example:
PRINT BIN$ (66,8) prints 01000010

CINT (n)
Returns rounded up integer value of **n** between –32768 and 32767.
Example:
PRINT CINT (3.8) prints 4

COS (n)
Returns cosine of **n** in degrees or radians *(see DEG and RAD)*.

CREAL (n)
Converts integer **n** to real numeric variable *(see section 1)*.

DERR
Returns an error code number from the disk filing system.

EOF
Checks to see if end of specified file has been reached during input. Returns 0 (false) until end of file, then –1 (true).

ERL
Returns the line number of the last error encountered.

ERR
Returns the error code number of the last error encountered.

EXP (i)
Returns the result of calculating *e* to the power **i** (e^i).
Example:
PRINT EXP (1) prints 2.71828183

FIX (n)
Removes the fractional part of **n** *(see INT below)*.

FRE (n/se)
Returns the amount of unused memory, irrespective of the nature or value of the *dummy argument* inside the bracket.
Examples:
PRINT FRE (0) or PRINT FRE ("hello")

HIMEM
Returns address of the highest memory address used by BASIC.

INKEY (i)
Checks to see if key number **i** is being pressed.

Value returned	[SHIFT]	[CTRL]	*Specified key*
-1	*ignored*	*ignored*	up
0	up	up	down
32	down	up	down
128	up	down	down
160	down	down	down

Example:

```
10 IF INKEY(43)=0 THEN STOP ELSE GOTO 10
```

runs in an endless loop until Y key is pressed alone

INP (add)
Returns value read from the I/O address **add**.

INT (n)
As in FIX if **n** is positive; if **n** is negative, it rounds it down.
Example:

```
PRINT INT(3.99), INT(-3.99)
```

prints 3 -4

JOY (i)
Returns bit-significant value from specified joystick. **i** = 0 or 1.

Bit	*Value returned*
0 (up)	1
1 (down)	2
2 (left)	4
3 (right)	8
4 (fire 2)	16
5 (fire 1)	32

Example:

```
JOY(1)
```

returns 40 if right joystick fire button is pressed

LOG (n)
Returns the natural logarithm (to base *e*) of **n**.

LOG10 (n)
Returns the logarithm to base 10 of **n**.

MAX (list of n)
Returns the maximum value from the given list.
Example:

```
PRINT MAX(3,8,25,1,2,9)
```

prints 25

MIN (list of n)
Returns the minimum value from the given list *(see MAX above)*.

PEEK (add)
Returns the contents of the specified memory location (0–65535).

PI
Returns value of π (3.14159265).

POS (#st)
Returns column number of print position relative to left edge of text window on stream **st**. **st** must be specified.
Example:

```
PRINT POS(#0)
```

prints 7

REMAIN (i)
Returns count remaining in delay timer **i** (0–3) then disables it.

RND [(n)]
Generates the next random number in the current sequence if **n** is positive or omitted. If **n** = 0, the random number generated will be the same as the last random number generated.

ROUND (n[,i1])
Rounds **n** to a number of decimal places or to the power of ten specified by **i**. If **i** is negative, then **n** is rounded to give an absolute integer with **i** zeros before the decimal point.
Example:
`PRINT ROUND(1562.357,2):ROUND(1562.375,-2)` **prints** `1562.36 1600`

SGN (n)
Returns `1` if **n** is positive, `0` if **n** = 0, `-1` if **n** is negative.

SIN (n)
Returns sine of **n** in degree or radian mode *(see DEG and RAD)*.

SQ (channel)
Returns a bit significant integer showing state of the sound queue for specified channel where **channel** 1, 2, 3 = A, B, C.

Bits 0, 1 and 2	number of free entries in the queue
Bits 3, 4 and 5	rendezvous state at head of this queue
Bit 6	head of the queue is held
Bit 7	channel is currently active

SQR (n)
Returns the square root of **n**.

TAN (n)
Returns the tangent of **n**. The DEG and RAD commands can be used to force the result to either mode.

TIME
Returns time elapsed since the computer was switched on or reset. One second = `TIME/300`.

VPOS (#st)
Reports the current row (line) position of the text cursor relative to the top of the text window of the specified stream.

7 BASIC STRING FUNCTIONS

ASC (s)
Returns ASCII code number of first character of string **s**.

CHR$ (i)
Returns the character whose ASCII code is given by **i** (0–255).

COPYCHR$ (st)
Copies character from current position in specified stream.

DEC$ (n,format)
Returns the decimal string representation of **n**, according to the specified format *(see PRINT USING)*.

HEX$ (i1,i2)
Returns a string hexadecimal digit representation of **i1** (0–65535). The number of hex digits in the string is given by **i2** (0–16).

INKEY$
Checks the keyboard and returns the string character of the key pressed. The string character returned is normally assigned to a string variable. If no key pressed, a null string is returned.

LEFT$ (se,i)
Returns a substring of **se**. The substring begins at the left-most character of **se** and contains **i** characters.
Example:
`A$ = "ABCDEFG" : PRINT LEFT$(A$,3)` prints ABC

LEN (se)
Returns the number of characters in **se** (0–255).

LOWER$ (se)
Returns a copy of **se** in which all alphabetic characters are converted to lower case *(also see UPPER)*.
Example:
`PRINT LOWER$("A1B2c3")` prints a1b2c3

MID$ (se,i1[,i2])
Returns a substring of **se** of length **i2** characters, starting at character **i1**. If **i2** omitted, substring continues to end of **se**.
Example:
`PRINT MID$("ABCDEFG",3,4)` prints CDEF

RIGHT$ (se,i)
Returns a substring of length **i** (0–255) characters from **se**, ending at the rightmost character of **se**.
Example:
`PRINT RIGHT$("ABCDEFG",3)` prints EFG

SPACE$ (i)
Creates a string containing **i** spaces (0–255).

STR$ (n)
Returns the string representation of number **n**.

STRING$ (i,s)
Returns i copies of the string character specified by **s**.
Example:
`PRINT STRING$(3,"*")` prints ***

UPPER$ (se)
Gives copy of **se** with all alphabetic characters in upper case.

VAL (se)
Returns the numeric value (including signs) of first numeric character(s) in **se**. Returns 0 if **se** starts with a non-number.
Example:
`PRINT VAL("-12.34x"),VAL("A-12")` prints -12.34 0

8 GRAPHICS COMMANDS

CLG [ink]
Clears the graphics screen to colour specified by **ink**. If parameter **ink** is not specified then the graphics screen is cleared to the colour specified by the GRAPHICS PAPER statement.

DRAW x,y [,[i1][,i2]]
Draws a line from current graphics cursor position to position **x,y**. **i1** specifies colour, i2 is logical colour.

i2 = 0	normal colour	**i2** = 2	AND colour
i2 = 1	XOR colour	**i2** = 3	OR colour

Example:
CLG2 : DRAW 500 , 400 , 0 draws a line from 0,0 to 500,400

DRAWR xr,yr,[[i1][,i2]]
Draws a line from current graphics cursor position to current cursor **x position** + **xr**, current cursor **y position** + **yr**. **i1** and **i2** as DRAW.
Example:
MOVE 200 , 200 : DRAWR 100 , 100 , 0
draws a line from 200,200 to 300,300

FILL i
Fills an area of a graphics screen in colour i (0–15). Default value of **i** is current graphics pen colour.

FRAME
Smooths character and graphic movement and reduces flicker.

GRAPHICS PAPER i
Sets graphics paper (background) colour to i (0–15).

GRAPHICS PEN [i1][,i2]
Specifies drawing colour i (0–15) to be used when drawing lines and plotting points. **i2** specifies background mode with 0 giving opaque and 1 giving a clear background.

MASK [i1][,i2]
Sets bits in each adjacent group of 8 pixels on (1) or off (0) according to binary value of **i1** (0–255). **i2** determines whether the first point of the line is to be plotted (1) or not (0).
Example:
10 CLG 2 : MASK1 : MOVE 0 , 0 : DRAW 500 , 400
20 MASK 15 : MOVE 0 , 0 : DRAW 500 , 400

MOVE x,y[[,i1][,i2]]
Moves the graphics cursor to position **x,y**. The parameter **i1** may be used to change the pen (drawing) colour. The parameter **i2** specifies the logical colour, as in DRAW.

MOVER xr,yr [[,i1][,i2]]
Moves the graphics cursor to point at **xr,yr** relative to its current position (ie. current **x position** + **xr**, current **y position** + **yr**).

ORIGIN x,y[left, right, top, bottom]
Sets the graphics origin (0,0) to position **x,y**. Graphics window dimensions may also be set to the given parameters.

PLOT x,y[[,i1][,i2]]
Plots point **x,y** on graphics screen. Optional **i1** and **i2** as in DRAW.

PLOTR xr,yr[[,i1][,i2]]
Plots a point at **xr,yr** relative to current position *(as MOVER)*.

TAG [#st]
Allows text to print at graphics cursor position *(see TAGOFF)*.

TAGOFF [#st]
Directs text to stream **st** printing it at previous text cursor position.

TEST (x,y)
Moves the graphics cursor by **x** and **y** relative to its current position, and returns the value of the ink at that position.

TESTR (x,y)
Moves the graphics cursor by **x** and **y** relative to its current position, and returns the value of ink at that position.

XPOS
Returns the current horizontal (**x**) position of the graphics cursor.

YPOS
Returns the current vertical (**y**) position of the graphics cursor.

9 SOUND COMMANDS

ENT en [,**es**] [,**es**]
[,**es**] [,**es**]
[,**es**]

en = *envelope number*
es = *envelope section*

Used with SOUND command to set tone envelope of **en** (0–15). If **en** is negative, envelope repeats until end of duration of SOUND command. Each **es** may have either two or three associated parameters. If **es** has three parameters, then they are:

Number of steps	*Step size*	*Pause time*
number of different steps (0–239) of tone (tone) the sound passes through during the envelope section	specifies pitch of sound (–128 to 127). Negative steps give higher pitch; positive steps give lower pitch	specifies pausing time between steps in 1/50 second units. Must not be greater than that in SOUND command

If **es** has only two parameters, then they are:

Tone period	*Pause time*
gives new setting for the tone period	specifies pausing time in 1/50 second units. Must not be greater than that in the SOUND command

Example:

```
10 ENT 1, 10, -50, 10, 10, 50, 10
20 SOUND 1, 500, 255, 15, 0, 1
```

ENV en [,**es**] [,**es**]
[,**es**] [,**es**]
[,**es**]

en = *envelope number*
es = *envelope section*

Each **es** may have either two or three parameters. If it has three parameters then they are:

Number of steps	*Step size*	*Pause time*
specifies how many different volumes the sound passes through during the envelope section	specifies the step size, varying from a volume level (0–15) with respect to the previous step	specifies pausing time in 1/50 second units

If **es** has two parameters then they are:

Hardware envelope	*Envelope period*
specifies the value to be sent to the envelope shape register of the sound chip	specifies the value to be sent to the envelope period registers of the sound chip

ON SQ (channel) GOSUB ln
Transfers control to subroutine at **ln** when there is a free slot in the given sound queue. **channel** set to 1, 2 or 3 for A, B or C.

RELEASE channel
Used to release sound channels which have been put in a hold state by the SOUND command. **channel** values must be between 1 and 7.

1: release channel A
2: release channel B
3: release channels A and B
4: release channel C
5: release channels A and C
6: release channels B and C
7: release channels A, B, and C

SOUND cs, tp [,du[,vol[,ve[,te,np]]]]

cs channel status
tp tone period defines the pitch (ie. note) of the sound
du duration (0–255) sets the duration of the sound
vol volume (0–15) specifies the starting volume of a note
ve volume envelope (1–15) varies volume during note
te tone envelope (1–15) varies tone or pitch during note
np noise period (0–31) adds white noise to sound. 0 = none

Produces a sound. The parameter **cs** yields an integer (1–255). The parameter is bit significant, with each bit signifying the following:

Bit 0: sends sound to channel A (decimal value 1)
Bit 1: sends sound to channel B (decimal value 2)
Bit 2: sends sound to channel C (decimal value 4)
Bit 3: rendezvous with channel A (decimal value 8)
Bit 4: rendezvous with channel B (decimal value 16)
Bit 5: rendezvous with channel C (decimal value 32)
Bit 6: hold sound channel (decimal value 64)
Bit 7: flush sound channel (decimal value 128)

Example:
SC=68 means send to channel C(4), with a hold state (64)

10 GENERAL AMSDOS COMMANDS

|A
Selects drive **A** as default drive. Used when two drives attached.

|B
Selects drive **B** as default drive. Used when two drives attached.

|CPM
Selects CP/M as current DOS and loads it from system disk.

|DIR, se
Displays disk directory and free space on disk. The parameter **se** specifies type of filenames to be displayed (eg. .BAS and .BAK).
Example:
|DIR,"*.*" displays a directory of all the files on disk

|DISC
Selects disk for both input and output operation. Equivalent to the two commands|DISC.IN and|DISC.OUT.

|DISC.IN
Selects a disk as the file input medium.

|DISC.OUT
Selects a disk as the file output medium.

|DRIVE, "A" or "B"
Selects the specified drive to be the default drive.

|ERA, se
Erases R/W (read/write) file(s) specified by **se** from disk.
Examples:
|ERA,"XX.BAS" deletes the BASIC file, XX.BAS, from disk
|ERA,"*.*" deletes all the files from the disk

|REN, "newname.ext","oldname.ext"
Renames a disk file from **oldname.ext** to **newname.ext**.

|TAPE
Selects the tape for both input and output operation. Equivalent to the two commands|TAPE.IN and|TAPE.OUT.

|TAPE.IN
Selects tape as file input medium.

|TAPE.OUT
Selects tape as file output medium.

|USER i
Determines which of the 16 (0–15) user areas of the disk system will be affected by current disk-related commands.

11 AMSDOS & BASIC FILE COMMANDS

CAT
Displays the names of all existing programs on the tape or disk.
Examples:
CAT [ENTER] lists all disk files in alpha-numeric order
TAPE [ENTER]
CAT [ENTER] lists names of all tape files in their storage order

CHAIN "filename" [,ln]
Enables the specified program to be loaded and RUN automatically. If the optional parameter **ln** is specified, the program execution will commence from line **ln**.

CHAIN MERGE "filename" [,ln][,DELETE ln1–ln2]
Loads the specified program from tape or disk, merges it into the program in memory, and starts execution of the merged program. The

parameter **DELETE 1nl–1n2** is used to delete part of the original program before running it, if required.

CLOSEIN
Closes any input file (tape or disk).

CLOSEOUT
Closes any output file (tape or disk).

DERR
Gives the most recent error code number returned by DOS.

EOF
Tests for the end of file. Returns –1 when EOF found, otherwise 0.
Example:

```
WHILE NOT EOF . . . WEND
```

LOAD "filename", add
Loads a program from disk or tape into the computer's memory. Memory used by binary files starts at **add** if specified.

MERGE "filename"
Loads the specified program from disk or tape and merges it with the program currently in memory.

OPENIN "datafile"
Opens the specified data file for reading.

OPENOUT "datafile"
Opens the specified data file for writing.

RUN "filename"
Loads in a BASIC or binary program and starts execution.

SAVE "filename" [,type] [binary parameters]
Saves the program in memory on disk or tape. **type** is either A (ASCII mode), B (binary mode) or P (protected mode). **binary parameters** can include start address, file length and entry point.
Example:
SAVE "XX", P saves BASIC file on disk and protects it
SAVE "XX", A saves BASIC file in ASCII (instead of as tokens)
SAVE "XX", B, 8000, 3000 saves the file XX in binary mode. The program starts at address 8000 and 3000 bytes are to be saved

SPEED WRITE i
Sets speed at which data is written to tape (only). **i** = 0 sets speed to 1000 baud, **i** = 1 sets it to 2000 baud.

WRITE *(see section 5 – General BASIC Commands)*

12 CP/M COMMANDS

AMSDOS
Transfers control to BASIC (and consequently to AMSDOS).

BOOTGN
Used in two-drive systems. Copies disk sector 1 track 0 (the loader), and the configuration sector from one disk onto another.

CHKDISC
Used in two-drive systems. Checks destination disk against a source disk for differences. If one is found, computer displays:

```
Failed to verify destination disc correctly:
(track x sector y)
```

CLOAD "tape filename" disk filename
Transfers ASCII files from tape to disk. It loads the specified tape file and stores it on the disk. If the first filename is omitted, then the first file on tape will be loaded, if second filename is omitted then disk file will be given same name as tape file.

COPYDISC
Used in two-drive systems to make a backup copy of an entire disk. It will also format the destination disk.

CSAVE "disk filename" tape filename i
Transfers ASCII files from disk to tape. If **tape filename** is omitted, then tape file will take same name as disk file. **i** specifies the tape speed to be used *(see section 11 – SPEED WRITE)*.

DIR
Lists the directory of the disk in the default disk drive.

DISCCHK
Used in single-drive systems to check a destination disk against a source disk for differences. Slower than CHKDISC command as you must swap the two disks when instructed.

DISCCOPY
Used in single-drive systems to format a destination disk and make a backup copy onto it. Swap disks as instructed.

ERA se
Erases specified file entry from directory. Data remains on disk (but hard to find!) until overwritten by more data.
Example:
ERA *.BAS erases all files with the extension .BAS from disk

FILECOPY se
Used in single-drive systems to copy files between disks. Instructions are provided to select files and swap disks.

FORMAT
Formats the disk in the default disk drive.

MOVCPM i
Moves CP/M to any 256 byte boundary address in memory. The parameter **i** (64–179) specifies the 256 byte boundary to be used.

PIP destination = source
Allows transfer of information between the computer and attached peripherals. **source** and **destination** may be either a file name or device token. The following device tokens may be used:

Source	Destination
CON: keyboard	CON: screen
ROR: serial interface	PUN: serial interface
	LST: printer

Examples:
PIP LST:=XX.BAS outputs file XX.BAS to the printer
PIP B:=A:*.* copies all the files from disk A to disk B

REN newname.ext = oldname.ext
Renames a disk file from **oldname.ext** to **newname.ext**.

SETUP
Allows you to redefine the characteristics of the CPC 664 keyboard and disk drive section. Also allows you to invoke various actions when CP/M is first loaded *(see CPC 664 user manual)*.

STAT [se]
Gives current disk information: number of records, number of bytes and R/W (Read and Write) or R/O (Read Only) status. File details supplied if **se** used. *(see CPC 664 user instructions)*.
Example:
STAT *.BAS details sizes and status of .BAS files

SYSGEN
Copies the image created by MOVECPM onto the system track of a disk *(see CPC 664 user instructions)*.

TYPE filename.ext
Lists the specified file on the screen.

13 ERROR MESSAGES

1 Unexpected NEXT – Occurs when the FOR of a FOR . . . NEXT loop is missing.

2 Syntax Error – Typing error or incorrect punctuation.

3 Unexpected RETURN – Caused by entering a subroutine other than with GOSUB.

4 DATA exhausted – Trying to READ data when data pointer has reached end of data.

5 Improper argument – The argument for a function is not legal (eg. PRINT SQR(--10)).

6 Overflow – The computer cannot handle a number greater than 1.7E↑38.

7 Memory full – All available RAM is being used or has been reserved. Program too big or control structures too deeply nested.

8 Line does not exist – Attempt to RUN, GOTO or GOSUB a non-existent line number.

9 Subscript out of range – Value of a subscript in an array is greater than DIM declaration.

10 Array already dimensioned – Arrays can only be DIMensioned once within a program.

11 Division by zero – Trying to divide a number by zero.

12 Invalid Direct command – Using a statement as a direct command when it is not allowed outside a program.

13 Type mismatch – Trying to assign string data to a numeric variable or vice versa.

14 String space full – String memory area is full.

15 String too long – Strings may not exceed 255 characters.

16 String expression too complex – A string expression needs to be broken down into smaller expressions.

17 Cannot CONTinue – CONT can only be used if program was stopped by [ESC] or a STOP in program – not after END.

18 Unknown user function – A DEF FN must be executed before calling an FN function.

19 RESUME missing – End of program has been reached while in error processing mode. Use ON ERROR before RESUME.

20 Unexpected RESUME – RESUME is only used in error processing mode, ON ERROR GOTO statement must be used first.

21 Direct Command found – A line without a line number has been found while loading a file.

22 Operand missing – An incomplete expression has been found.

23 Line too long – The line contains too many statements.

24 EOF met – Trying to input data beyond end of data file.

25 File type error – Using a program file instead of a data file to read or write (or vice versa).

26 NEXT missing – The NEXT of a FOR . . . NEXT loop is missing.

27 File already open – Trying to open an open file. Use CLOSEIN or CLOSEOUT first.

28 Unknown command – Given when an unknown command follows a|.

29 WEND missing – The WEND part of the WHILE . . . WEND loop is missing.

30 Unexpected WEND – WEND encountered without a corresponding active WHILE.

31 File not open – Attempting to read from or write to a file without OPENing it first.

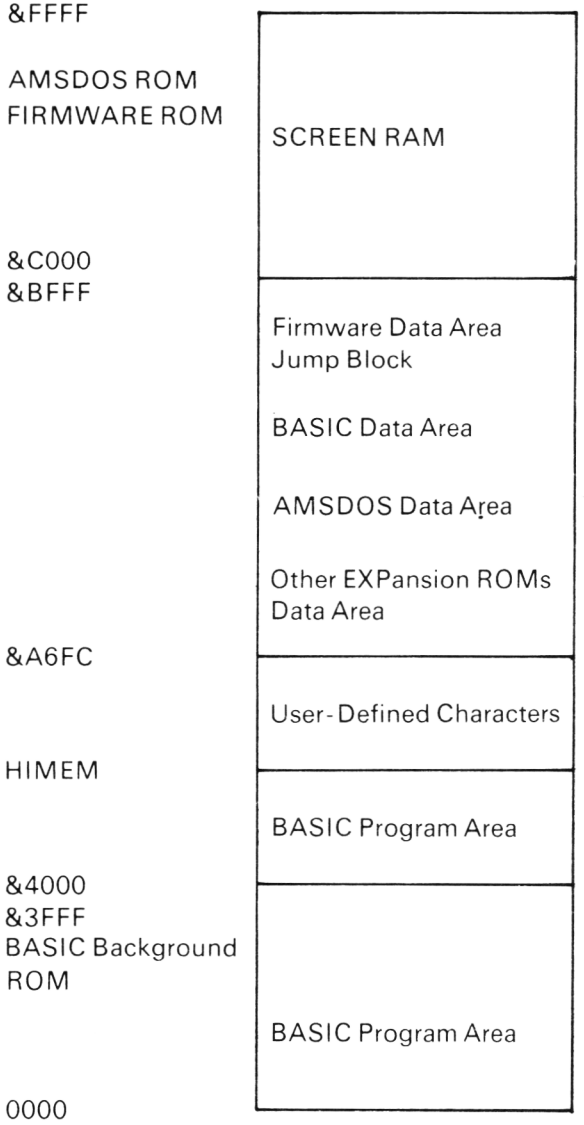
AMSDOS disk error messages

Error number	*DEER value*	*Description*
0	0 or 22	[ESC] has been pressed
14	142*	unsuitable stream state
15	143	hard end of file reached
16	144	bad command
17	145	file already exists
18	146	file does not exist
19	147	directory is full
20	148	disk is full
21	149	changed disk while files open

Error number	*DEER value*	*Description*
22	150	file is read/only
26	154	soft end of file detected

* 142 made up of 14 + 128

14 MEMORY MAP



15 LIST OF AMSTRAD KEYWORDS

| A
ABS
AFTER
AMSDOS
AND
ASC
ATN
AUTO
| B
BIN$
BOOTGN
BORDER
CALL
CAT
CHAIN
CHAIN MERGE
CHKDISC
CHR$
CINT
CLEAR
CLG
CLOAD
CLOSEIN
CLOSEOUT
CLS
CONT
COPYCHR$
COPYDISC
COS
| CPM
CREAL
CSAVE
CURSOR
DATA
DEC$
DEF
DEFINT
DEFREAL
DEFSTR
DEG
DELETE
DERR
DI
DIM
DIR
| DIR
| DISC
DISCCHK
DISCCOPY
| DISC.IN
| DISC.OUT
DRAW
DRAWR
| DRIVE
EDIT
EI
ELSE
END
ENT
ENV
EOF
ERA
| ERA
ERASE
ERL
ERR
ERROR
EVERY
EXP
FILECOPY
FILL
FIX
FN
FOR
FORMAT
FRAME
FRE
GOSUB
GOTO
GRAPHICS PAPER
GRAPHICS PEN
HEX$
HIMEM
IF
INK
INKEY
INKEY$
INP
INPUT
INSTR
INT
JOY
KEY
LEFT$
LEN
LET
LINE
LIST
LOAD
LOCATE
LOG
LOG10
LOWER$
MASK
MAX
MEMORY
MERGE
MID$
MIN
MOD
MODE
MOVECPM
MOVE
MOVER
NEXT
NEW
NOT
ON BREAK
ON ERROR
ON GOSUB
ON GOTO
ON SQ
OPENIN
OPENOUT
OR
ORIGIN
OUT
PAPER
PEEK
PEN
PI
PIP
PLOT
PLOTR
POKE
POS
PRINT
RAD
RANDOMIZE
READ
RELEASE
REM
REMAIN
REN
| REN
RENUM
RESTORE
RESUME
RESUME NEXT
RETURN
RIGHT$
RND
ROUND
RUN
SAVE
SETUP
SGN
SIN
SOUND
SPACE$
SPC
SPEED
SQ
SQR
STAT
STEP
STOP
STR$
STRING$
SYMBOL
SYMBOL AFTER
SYSGEN
TAB
TAG
TAGOFF
TAN
| TAPE
| TAPE.IN
| TAPE.OUT
TEST
TESTR
THEN
TIME
TO
TROFF
TRON
TYPE
UNT
UPPER$
USER
USING
VAL
VPOS
WAIT
WEND
WHILE
WIDTH
WINDOW
WRITE
XOR
XPOS
YPOS
ZONE

NOTES

The Century Microguide to the Amstrad is a conveniently sized, clearly laid out, quick reference guide for the busy Amstrad owner. It comprehensively summarizes all the essential information needed by the Amstrad enthusiast and includes:

- Special Keyboard Features
- Alphabetic Quick Reference
- Locomotive BASIC Commands
- Sound, Graphics, Text and Colour
- Numeric, Trigonometric and String Functions
- Input/Output Functions
- Arithmetic and Logic Operations
- File Handling Commands
- Indirection Operators
- Memory Maps
- Error Handling, Codes and Messages
- Operating System Commands
- Disc System Commands

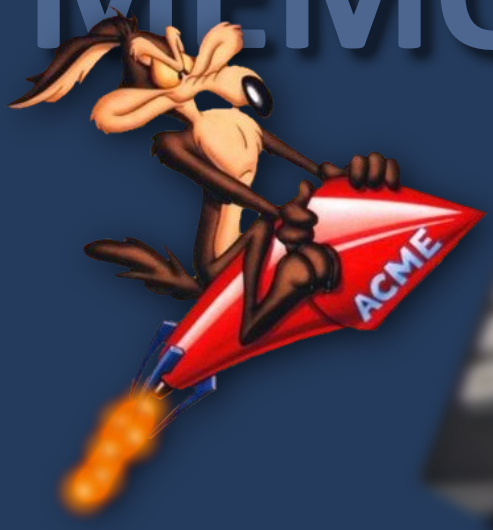
Each command is illustrated with simple examples to show how it is used in context and there are practical hints throughout the book.

CENTURY COMMUNICATIONS LTD.

Document numérisé avec amour par :

AMSTRAD CPC

MÉMOIRE ÉCRITE



https://acpc.me/