DRAGON

MICRO GUIDE

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

DRAGON

MICROGUIDE
FOR THE
DRAGON

Professor Peter Morse
Brian Hancock

CENTURY
COMMUNICATIONS

London

CONTENTS

SECTION 1
BASIC KEYWORDS

Keyword	Brief meaning	Section
ABS	absolute value	4
AND	logical AND	2
ASC	american standard code	4
ATN	arc tangent	4
AUDIO ON	audio on	7
AUDIO OFF	audio off	7
CRH$	character string	4
CIRCLE	draw circle	11
CLEAR	clear variables	3
CLOAD	load from cassette	7
CLOADM	load machine code from casette	7
CLOSE #	close file	7
CLS	clear screen	3
COLOR	colour	11
CONT	continue execution	6
COS	cosine	4
CSAVE	save on cassette	7
CSAVEM	save machine code on cassette	7
DATA	data	3
DEF	define function	3
DEL	delete lines	6
DIM	dimensions array	3
DRAW	draws line	11
EDIT	edits line	6
ELSE	else	3
END	end of program	3
EOF(-1)	end of file test	7
EXP	exponential	4
EXEC	execute machine code	3
FIX	rounds down	4
FN	function	3
FOR	for loop	3
GET	store graphics	11
GOSUB	go to subroutine	3
GOTO	go to address	3
HEX$	hexadecimal	4
IF	if	3
INKEY$	gives character of the key pressed	4
INPUT	inputs data	3
INPUT #	inputs information from file	5
INSTR	in string	4
INT	integer	4
JOYSTK	gives x and y values for the position	4
LEFT$	left string	4
LEN	length of string	4
LET	let	3
LINE	draws line/rectangle	9
LINEINPUT	inputs a line of text	3
LIST	lists program on screen	1
LLIST	lists program on printer	1,6
LOG	natural logarithm	4
MID$	middle of string	4

Keyword	Brief meaning	Section
MOTOR ON	cassette motor on	5
MOTOR OFF	cassette motor off	5
NEW	erase old program	3
NEXT	next	3
NOT	logical inverse	2
ON GOSUB	on condition go to subroutine	3
ON GOTO	on condition go to address	3
OR	logical OR	2
OPEN #	opens cassette file	5
PAINT	paint picture	9
PCLEAR	reserve graphics pages	9
PCLS	clear high-resolution screen	9
PCOPY	copies one graphics page to	9
PLAY		8
PEEK		3,4
POINT	point colour (low-resolution)	4
POKE	poke	3,4
POS	print position	4
PPOINT	point colour (high-resolution)	4
PRESET	reset high resolution point	9
PRINT	prints to screen	3
PRINT #	writes to file or printer	5,6
PRINT@	controlled print to screen	3
PRINT USING	formatted print to screen	3
PRINT #, USING	formatted print to file or printer	6
PSET	sets high resolution pixcell	9
PUT	puts stored picture on the screen	9
READ	read data	3
REM	remark	3
RENUM	renumber	1
RESET	reset low-resolution cell	9
RESTORE	restore data pointer	3
RETURN	return from subroutine	3
RIGHT$	right string	4
RND	random number	4
RUN	executes program	1
SCREEN	selects screen	9
SET	sets low-resolution cell	9
SGN	sign	4
SIN	sine	4
SKIPF	skips program on tape	5
SOUND	sound	8
SQR	square root	4
STEP	step	3
STOP	stops execution	1
STR$	string representation	4
STRING$	multiple strings	4
TAB	tabulation	3
TAN	tangent	4
TIMER	timer	4
TRON	program trace on	1
TROFF	program trace off	1
USR	user	3
VAL	value	4
VARPTR	variable pointer	4

Arithmetic operators

Symbol	Operation
+	Addition
−	Subtraction
*	Multiplication
/	Division
	Unary minus
↑	Exponentiation

The only operator that can be used with strings is "+". This is used to join strings together, and the process is known as concatenation.

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

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

List
A one dimensional array.
eg A(15), A$(15) contain 16 numbers and 16 strings respectively.

Logical operators

Symbol	Operation
AND	logical AND
eg IF A=0 AND B=0 THEN PRINT "ZERO"	
NOT	logical NOT
eg IF NOT A=0 PRINT "NOT EQUAL TO ZERO"	
OR	logical OR
eg PRINT "NOT A TEENAGER"	

Number
A positive or negative decimal quantity which is significant to about 8 digits, and whose magnitude is between an approximate minimum of:
$\pm 3 \times 10^{-39}$
and maximum of:
$\pm 10^{+38}$

Numeric expression
An expression having a numeric value. When an integer is required, the number represented by the expression is truncated to give an integer value.

Numeric variable
A variable name composed of a single letter or a single letter followed by a single digit or a letter followed by another letter,

which names a numeric value or a collection of numeric values. If more than two characters are used, the characters after the second one are used as reference by the programmer for identifying the variable name. Care must be taken in naming a variable, that is, the name should not contain any of the Dragon basic commands. For example, TOTAL used as a variable name will give SN error since TO is a basic command.

Print-list
A list of items separated by commas, or semicolons. The items can be variables, expressions or string constants.
eg A, AB, A1, SUM, NUMBER

Relational operators

Symbol	Operation	Priority
=	Equal to	5
<	Less than	5
<=	Less than or equal to	5
>	Greater than	5
>=	Greater than or equal to	5
<>	Not equal to	5

String
A sequence of characters each of which is a letter, digit, space, or some character other than a line terminator or carriage return. ASCII code is used to represent these characters in computer.

String constant
A string enclosed in double quotes.
eg
"ENTER A NUMBER"

String expression
An expression having a string value.
eg
LET A$=B$+C$

String variable
Used to name a string or collection of strings. The rules for naming string variables are similar to that of numeric variables except that the variable name must be followed by a dollar sign ($).
eg
A$, AB$, A1$, SUM$, NUMBER$

Table
A two dimensional array.
eg
A(10,10), A$(10,10)

SECTION 3
BASIC STATEMENTS

Meaning

CLEAR n, ba
Clears all variables from memory and reserves n bytes of space for string storage. The highest BASIC address is given by ba, after which machine language routines will be stored.
eg 10 CLEAR 1000
clears 1000 bytes for string storage
10 CLEAR 1000, 14000
clears 1000 bytes and sets highest BASIC address to 14000

CLS c
Clears screen to the specified colour, given by c. The default value is 1.

c	Colour
0	Black
1	Green
2	Yellow
3	Blue
4	Red
5	Buff
6	Cyan
7	Magenta
8	Orange

DATA data1, data2, data3, .
Provides constant data (numbers or strings) for READ statements.
eg
```
 5 DATA 35,-20, 7.25,-2.5, 18
10 FOR I=1 TO 5
20 READ A(I)
30 NEXT I
```
will assign 35 to A(1), −20 to A(2), 7.25 to A(3), and so on.

DEF FNx (dummy variable)=Expression
Defines a user-specified function. x is any letter from A to Z, and the dummy variable is a letter, which will be replaced by the function argument when the function is called.
eg
```
10 DEF FNA(y)=y*y+1
30 LET x=3
40 LET B=FNA(x)
50 PRINT B
```
will print 10 on the screen. The reason for this is that the dummy variable y is replaced by x when the function FNA is called at line 40.

DEFUSRn=address
Defines entry point for user defined machine language subroutine n, where n=0−9 and the address is between 0-65535 and contains the entry address for USRn.
eg
DEF USR1=30000

DIM variable (subscript), . .
Specifies the space to be allocated for a list or a table. If a list or a table is not specified in a DIM statement, the default dimension for a list is 10 elements and the default for a table is 10 (rows) by 10 (columns).
ie
100 elements
eg
DIM A(5)
DIM x(10,20), xy$(10,12), A(5)

END
Indicates the end of program and usually is the statement with the highest line number (ie the end of program).
1000 END

EXEC address
Transfers control to machine language program at address specified.
eg
50 EXEC 30000

FOR numeric variable = n TO m STEP s
Specifies a loop and must be used with a NEXT statement. The loop is executed for a range specified by n and m. S indicates increment in step (ie next n=n+s), if omitted increment is by 1. n, m and s may be numbers or numeric expression.
eg
```
 10  FOR I=1 TO 20 STEP 2
 50  NEXT I
100  FOR J=0 TO 2 STEP 0.2
110  PRINT J
120  NEXT J
```

GOSUB line number
Enters a subroutine at the specified line. The subroutine is exited from by executing a RETURN statement.
eg
100 GOSUB 1000

GOTO line number
Transfers control to the statement at the specified line.
eg
100 GOTO 500

IF condition THEN task1 ELSE task2
The condition is evaluated and control is transferred to THEN portion (task1) if the condition is true. If condition is not true, then the ELSE portion (task2) is executed. Omitting ELSE portion of the statement will cause the control to be transferred to next statement in the program if condition is not true.
eg
```
100  IF A$ <> "END" THEN GOTO 10 ELSE STOP
180  IF A >=50 THEN LET A=2
190  LET A=A+1
```

INPUT "PROMPT"; variable, variable, . . .
Allows data to be entered from the keyboard. This statement causes a "?" to be output on the screen so that you can respond by typing in values for the requested variables. When inputing a string, leading spaces before first character will be ignored and if a comma is entered as part of a string, the characters after "," will be ignored.
eg
```
INPUT A,FRED, TEL$, B$
INPUT "ENTER VALUE FOR x";x
INPUT "ENTER NAME" A$
```

LET variable = expression
Assigns the value of the expression to the specified variable. The word LET may be omitted.
eg
```
10  LET A=3.5
20  LET B=A*3+A2
30  D=A+B
```

LINE INPUT "prompt"; string variable
Allows entry of an entire line (255 character long) into a string variable from the keyboard. The text entered may contain leading spaces and commas as they cannot be entered using INPUT statement. Line is terminated by pressing [ENTER]. Only one variable can be used.
eg
```
LINE INPUT "ENTER NAME AND ADDRESS";A$
```

ON expression GOSUB line number 1, line number 2, .
Jumps to subroutine at line given by line number 1 if the expression evaluates to 1, and to subroutine at line given by line number 2 if the expression evaluates to 2 and so on. If the value of the expression is real (ie with decimal point), then the value is truncated and the integer portion is considered.
eg
```
 10  D=RND(5)
 20  ON D GOSUB 1000, 2000, 3000, 4000, 5000
100  ON (x+3)/y GOSUB 500, 1000, 1500
```
If value is negative, an error message will result, if value = 0 or value greater than the number of lines in the line number-list then the statement will be ignored.

ON expression GOTO line number 1, line number 2,
As in GOSUB, Control is transferred to the statement at the line given by line number-list depending on the value (1, 2, 3, . .) of the expression.
eg
```
 50  ON G GOTO 50, 80, 110, 140
100  On (x*3)/G GOTO 100, 300
```

PEEK (address)
Returns the contents of the specified memory location. The address is in the range of 0-65535.
eg
```
10  B=PEEK (65280)
```

20 PRINT B
assigns the contents of memory location 65280 which contains the result of checking joystick button. If right joystick button is pressed, the value will be 126 or 254, if left joystick button is pressed, the value will be 125 or 253.

POKE address, value
Is a command that alters the contents of the specified memory location (ie the specified value is put in memory location specified by address). Value must be between 0-255.
eg
10 FOR I=1024 To 1535
20 POKE I,42
30 CONTINUE
will fill the screen with *.42 is the ASCII code for *, and 1024-1535 is the text screen memory.

PRINT print-list
Types out results of computation, messages, and/or types out a blank line. The print-list consists of strings, variables, etc separated by format control characters. The format control characters are:
Comma(","):– causes the output to be printed in the next print zone (each print zone = 16 character position).
Semicolon (";"):– causes the output to be printed in a close packed form (ie print position remains in the current position).
Omitting print-list, prints a blank line.
eg
10 PRINT
20 PRINT A,B
30 PRINT "SUM=";S,"ITEM= ";IT$

PRINT TAB(C);
Moves the print position to specified column position.
eg
10 PRINT TAB(5);A;TAB(7);B$;TAB(S);C

PRINT USING "format"; output-list
Prints the output list according to the specific format. The format consists of string of characters describing the form of the output and placement of the output on the output device. If PRINT #–1 USING is used, the output device will be cassette recorder, if #–2 is used, the output will be printed on printer. Formats used are as follows:

#
Formats numbers
eg PRINT USING "###";147·76 *result* 148

Decimal point
eg PRINT USING "###·#";147·76 *result* 147.8

Inserts a comma to the left of every third character.
eg PRINT USING "###,#";14776 *result* 14,776

Fills leading spaces with *
eg PRINT USING "**#####.###";147.76 *result* 147.760

$
Places $ ahead of number
eg PRINT USING "$#####.#";147.76 *result* $▽▽147.8

Prints in exponential form
eg PRINT USING "###.##";147.76 *result* 14.78E+01

PRINT @ location, output-list
Prints the output-list at specified location on the screen. Location can be a number or an expression and must evaluate to a number between 0-511.
eg
PRINT @ 68, "YOU HAVE";SUM;"DOLLAR"
PRINT @ C+32*L, "TEL. NO.";TEL
where c = column (0-31), L = line (0-15)
PRINT @ 5+32*2,"JOHN";AGE
prints at column 5, line 2 the prompt JOHN followed by his age.

READ variable, variable,
Assigns the data (numeric or string) in DATA statements to the specified variables. This statement must be used with one or more DATA statements.
eg
10 READ A, B, C$
20 PRINT A;B;"▽▽";C$
30 DATA 10, 20, SUM

REM
Inserts comment lines in the program. Everything after REM is ignored by the computer and its presence is to aid the user. Single quotes (ie ' obtained by pressing [SHIFT][7]) has the same effect.
eg
1000 REM START OF SUB1
1000 'START OF SUB1

RESTORE
Allows data in DATA statements to be read more than once. This statement sets the data block pointer back to the beginning of the collection of data values.
eg
80 RESTORE

RETURN
Exits the subroutine and directs BASIC to go to the statement following the last GOSUB from which it transferred. A subroutine must at least have one RETURN statement.
eg
1100 IF AN$="N" THEN RETURN

1200 RETURN

STOP
Stops program execution at line containing the STOP statement. Program execution can be resumed by typing CONT [ENTER], which will cause the program to continue execution at line following the STOP statement.
eg
```
100  IF ANS$<>"y" THEN STOP
110  LET I=I+1:GOTO 10
```

SECTION 4
BASIC FUNCTIONS

Section A: Numeric Function

ABS(X)
Returns the absolute value of X, where X can be a number, a variable, or an expression.
eg
```
PRINT ABS(-3.5) Prints 3.5
PRINT ABS (A)
PRINT ABS (A*B+3*C)
```

ATN(X)
Returns the arctangent (ie $\tan^{-1}$) of X in radians, where X can be a number, a variable or an expression.
eg
```
y=ATN (5.8)
PRINT ATN (x+5)
```

COS(X)
Returns the cosine of angle X given in radians.
eg
```
10  INPUT ANGLE
20  PI=4*ATN(1)
30  PRINT COS(ANGLE*PI/180)
```
line 20 gives value for PI = 3.1415927, and line 30 prints the cosine of the given angle. ANGLE*PI/180 converts from degrees to radians.

EXP(X)
Returns the result of e to the power specified by X.
eg
```
PRINT EXP(0)    prints 1 ie e^0=1
PRINT EXP(1)    prints 2.71828183
PRINT EXP(2)    prints 7.3890561
```

FIX(X)
Returns the integer part of the number represented by X (ie trancates all digits after the decimal point).
eg
```
10  LET A=FIX (3.999)
20  PRINT A (prints 3)
```

INT(X)
As in FIX if X is positive, if X is negative, it rounds it down.
eg
PRINT INT(3·99) prints 3
PRINT INT(−3·099) Prints −4

JOYSTK(X)
Returns the horizontal, or vertical coordinate of the left or right joystick. X must have value between 0 and 3.
X=0 gives horizontal (x) coordinate of left joystick
X=1 gives vertical (y) coordinate of left joystick
X=2 gives horizontal (x) coordinate of right joystick
X=3 gives vertical coordinate of right joystick
eg
```
10  LET HL=JOYSTK(0):LET VL=JOYSTK(1)
10  LET HR=JOYSTK(2):LET VR=JOYSTK(3)
```

LOG(X)
Returns the logarithm to the base E (ie natural log) of value representing X.
eg
```
PRINT LOG(1·7)+3*LOG(2·8)
```

PEEK (address)
Returns the contents of the specified memory location.
eg
```
10  FOR I=1024 TO 1535
20  PRINTI,PEEK (I):NEXT I
```
will print the contents of the text screen memory.

POINT (Xcoord, Ycoord)
Checks to see if the low resolution graphic/text cell at position specified by Xcoord (ie horizontal coordinate, 0≤63), and Ycoord (vertical coordinate, 0≤31) is on or off. If the cell is on, then the colour code (0-8) of the cell is returned. If the cell is off (ie RESET), then 0 is returned, and −1 is returned if the cell contains a text character.
eg
```
10  SET (25,25,3)
20  P1=POINT(25,25):P2=POINT(30,40)
40  PRINT P1, P2
50  RESET (25,25):P3=POINT(25,25)
60  PRINT P3
```
line 40 will print 3 and −1 respectively and line 50 switches the cell off, and therefore 0 is printed as result of line 60.

POS (device)
Returns the print position of the specified device.
device=0 screen display
device=−2 printer
eg
IF POS(0)>16 THEN PRINT, ELSE PRINT A
moves to next print zone if condition is true else prints value of A at the same line.

PPOINT (Xcoord, Ycoord)
As in point, but a high resolution cell is checked to see if on or off. Returns value 0 or colour code only, since text cannot be used in high resolution graphics.
0≤Xcoord≤127-255
0≤Ycoord≤127-191

depends on mode used
eg
```
10  LET C=PPOINT (180-120)
```

RND(X)
Generates a random number according to the integer value given by X. If X=0, the random number generated will be between 0 and 1 (ie 0≤ random number <1).
If X>1, then number generated will be between 1 and the specified range (ie 1≤ random number ≤X).
eg
```
LET D=RND(0)
LET D=RND(100)
```

SGN(X)
Returns value of 1 if the value representing X is positive, 0 if X has value of zero, and −1 if the value of X is negative.
eg
```
10  IF SGN(A*B−6)=1 THEN PRINT "+VE" ELSE
IF SGN(a*b−6)=−1 THEN PRINT "−VE" ELSE
PRINT "ZERO"
```

SIN(X)
Returns the sine of angle X. (X is in radians), (X radians = 0 degrees *π/180).
eg
```
10  INPUT ANGLE
20  PI=4*ATN(1)
30  PRINT SIN(ANGLE*PI/180)
```
line 30 converts angle given in degrees to radians and prints the sine of the given angle.

SQR(X)
Returns square root of the value representing X.
eg
```
PRINT SQR(16)
LET R=(−B+SQR(B**2−4*A*C))/2*A
```

TAN(X)
Returns the tangent of the value representing X. X must be in radians.
eg
```
PRINT TAN (ANGLE*PI/180)
```

TIMER
Returns the contents of the timer which increases by one every fiftieth (1/50) of a second. The reset the timer USE TIMER=0
eg
```
 5  CLS
```

```
10  TIMER=0
20  TM=TIMER/50:PRINT@75,TM
30  GOTO 20
```
line 20 prints value of TM on the screen which is increment by 1 every second.

USR(X)
Transfers control from BASIC to a machine language routine. The address at which the machine language starts is given by DEF USR(X) statement. X is an integer value, refering to subroutine number.

VARPTR (variable name)
Returns address of pointer to specified variable (eg a pointer to a BASIC variable can be used as an argument by a USRn function. This would allow a USR function to access elements of an array).
eg
```
USR0(VARPTR(A))
```

SECTION 5
Section B: String Functions

ASC(string$)
Returns the ASCII code number for the first character of the string variable.
eg
```
IF ASC(A$)>=48 AND ASC(A$)<=57 THEN
PRINT "NUMERIC CHARACTER"
```
checks to see if the character is a number between 0 and 9.

CHR$(n)
Returns the character whose code is given by n, where n is an integer number between 0 and 255.
eg
```
 5  CLS
10  FOR I=32 TO 255
20  PRINT @ 96,"CODE=";I;"CHARACTER=";CH
    R$(I)
25  FOR J=1 TO 500:NEXT J
30  NEXT I
```
This will print the printable character set of the Dragon, with a delay loop at line 25 enabling you to see characters before they change.

HEX$(n)
Returns the hexadecimal equivalence of n, where n is an integer value.
eg
```
10  FOR I=1 TO 16
20  PRINT HEX$(I)
30  NEXT I
```
prints Hex equivalence of 0-16; which is 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10

INKEY$
Checks the keyboard and returns the string character of the key pressed. The string character returned is normally assigned to a string character. This command is normally used in game programmes to move an object, 'say a ship' around.
eg
```
 10  LET A$=INKEY$
 20  IF A$=CHR$(8) THEN X=X-2
100  IF A$="S" THEN STOP
```

INSTR(n, string$, substring)
Searches within the string specified by the string$ for the specified substring and returns the position number of the first character of the match. The n specifies the character position in string$ at which search begins. Zero returned if there is no match.
eg
```
10  LET F=INSTR (5, "ABCDEFGH", "FH")
    F will be assigned value 6.
10  LET A$="ABCDEFGH"
20  LET F=INSTR (5, A$, "FG")
```

LEFT$(String$,n)
Returns a substring of the specified string. The substring begins at the leftmost character of the string$ and contains the number of characters specified by n, where n is an integer value.
eg
```
10  LET A$="ABCDEFGH"
20  PRINT LEFT$(A$,5)
    Prints ABCDE
30  PRINT LEFT$("ABCDEFGH",5)
```
will also print ABCDE.

LEN (String$)
Returns the number of characters in the specified string (ie length of the string).
eg
```
10  LET A$="ABCDEFGH"
20  PRINT LEN(A$)
    Prints 8
30  PRINT LEN ("ABCDEFGH")
    also Prints 8.
```

MID$(String$, n, m)
Returns the substring of the specified string, starting at character position specified by n, and containing the number of characters specified by m, where n and m are both integer values. Omitting m will cause the substring to continue to the end of the specified string.
eg
```
10  LET A$="ABCDEFGH"
20  PRINT MID$(A$,3,6)
    Prints CDEF
```

RIGHT$(String$, n)
Returns a substring of the specified string which ends at the right most character of the string and contains the number of

characters specified by n, where n is an integer value.
eg
```
10  LET A$="ABCDEFGH"
20  PRINT RIGHT$(A$,2)
    Prints GH
```

String$(n, String$)
Returns n copies of the first character of the specified string. It is also possible to specify the ASCII code of the required character instead of the string.
eg
```
10  PRINT STRING$(32,42)
20  PRINT STRING$(32,"*")
```
both lines print 32 asterisks.

STR$(n)
Returns the string representation of the integer value given by n (ie converts numbers to strings).

eg
```
10  LET $="DECEMBER▽"+STR$(26) (▽=SPACE)
```

VAL (String$)
A$ will contain "DECEMBER 26". Returns the numeric characters of the specified string as a number.
eg
```
10  LET NUM=VAL("123")
20  LET A$="111"
30  PRINT NUM+VAL(A$)
    Prints 234
```

SECTION 6
CONTROL KEYS

[BREAK]
Stops program execution and transfers control to keyboard.

[CLEAR]
Clears the screen.

[ENTER]
Indicates the end of input.

[SPACEBAR]
Blank character (space).

[←]
Backspace. Rubs out last character input.

[SHIFT][←]
Rubs out the whole line.

[SHIFT][@]
Causes execution of program to pause until a key is pressed. Used to pause output to screen.

[SHIFT][0]
Pressed once switches to lower case (inverse video), press again to switch to upper case.

SYSTEM COMMANDS

CONT
Continues program execution that is interrupted by either pressing [BREAK] or as result of STOP statement in the program. The execution continues from the line of interrupt.

DEL line number 1, line number 2
Deletes lines between the specified range.
eg
DEL 10 deletes line 10
DEL 10-100 deletes lines 10 to and including 100
DEL –50 deletes from current line to line 50
DEL 100– deletes from line 100 to the end.

EDIT line number
Allows alterations to be made in the specified line. To enter EDIT mode type:
EDIT line number [ENTER]
Once in this mode the following subcommands can be used to alter contents of the line.

Subcommands
C char
Changes the current character to character specified by char.

nC char
Changes the next n characters from the current position to the specified character.

D
Deletes current character.

nD
Deletes the next n characters from the current position.

H
Deletes rest of the line from the current position, and goes to insert mode (ie waits for you to enter new characters).

I chars
Inserts characters at the current position.

K
Deletes rest of the line from current position.

nK char
Deletes characters from current position to nth occurrence of the character specified by char.

L
Lists the current state of the line.

S char
Searches for the first occurrence of the character specified by char.

nS char
Searches for the nth occurrence of the specified character.

X
Used to extend the line, the cursor moves to the end of the line and waits for characters to be input.

[SPACEBAR]
Moves the cursor one position forward (ie to the right).

n [SPACEBAR]
Moves the cursor n positions forward.

[←]
Moves the cursor one position back (ie to the left). If in insert mode deletes the character of the current position.

n [←]
Moves the cursor n positions backwards.

[SHIFT][↑]
Leaves insert mode and returns to edit mode.

[ENTER]
Leaves either, and stores the altered line.

LIST line number 1 – line number 2
Lists lines between the specified range on the screen. If line number 2 is omitted, only line number 1 will be listed.
eg
LIST 10 lists line 10.
LIST 10– lists lines 10 to the end of program.
LIST –100 lists from current line to line 100.
LIST 110-310 LISTS lines 110 to and including line 310.

LLIST line number 1, line number 2
As in list, but listing will beon printer instead of screen.

NEW
Clears computer memory, ie deletes program and variables.

RENUM new start line, old line number, increment
Renumbers the lines of program from the specified old line number to the end of program.
eg
RENUM renumbers entire program as: 10,20,30, .
RENUM,,5 renumbers the entire program as: 5,10,15, .
RENUM 100 renumbers the entire program as: 100,110,120, .
RENUM 50,10,5 renumbers the program as: 50,55,60, . lines before 10 remain unchanged.

RUN line number
Executes program from the line specified. Omitting the line number will execute the program from the lowest line number (ie start of program). Direct command GOTO line number can also be used to execute a program.
eg
RUN
RUN 100
GOTO 100

TRON
Used in debugging (ie finding bugs) in a program. It turns on program flow trace, and causes the line number of the statement being executed to be printed on the screen.

TROFF
Turns off the program flow trace.

SECTION 7
CASSETTE RECORDER CONTROL STATEMENTS

AUDIO OFF
Disconnects the cassette output to TV speaker.
eg
AUDIO OFF

AUDIO ON
Connects the cassette output to TV speaker. This enables you to play music while playing a game or add comments to an educational program. It can also be used to find a stored program on tape if the name is specified by you before storing your program.
eg
AUDIO ON

CLOAD "file-name"
Loads the specified file from cassette into computers memory. If the file is not at the current tape position, it will search until the name is encountered. At this stage, the Find cursor F will replace the S cursor and OK message will appear after loading. Omitting the file name leads the first file encountered on tape. The file name must be up to 8 characters long.

eg
CLOAD " "
CLOAD "GAME A"

CLOADM "file name", offset
Loads the machine language program from cassette into memory at an address given by start addres + offset.
eg
CLOADM"GAME A",1500

CLOSE #−1
Closes a data file that has been opened for either input or output.
eg
200 CLOSE #−1
line 200 closes input/output to a cassette.

CSAVE "file name"
Outputs the specified program to cassette recorder. If the cassette recorder is in record mode, then a copy of the program will be made on the tape provided the tone and volume of the recorder is adjusted properly. File name must not be more than 8 characters. NB Set tone to ¾ of maximum and volume to just above ½ the maximum.
eg
CSAVE "GAME A"

CSAVEM "file name", start, end, entry
Outputs the specified machine language program to cassette recorder. Start gives the starting address of program in memory, end gives the last address occupied by the program and entry gives the program entry point.
eg
CSAVE 3900,39FF,390F

EOF(−1)
Used when inputing data from cassette recorder. Checks to see if the end of the specified file on cassette is reached. Returns true if the end of file is reached.
eg
```
10   OPEN "I", #−1, "GAME A"
20 IF EOF(−1) THEN GOTO 40
30   INPUT #−1, A,B:PRINT A,B:GOTO 20
40   CLOSE #−1
```

INPUT #−1, print list
Inputs data from cassette file. Before using this statement the required file must be opened (using OPEN statement).
eg
```
10   OPEN "I",#−1, "GAME A"
20   INPUT #−1,A,B, .
```

MOTOR ON
Turns cassette motor on. This is useful if cassette recorder is remote controlled (ie EAR, AUX, and REM plugs are connected to the computer). Another words it returns control to the cassette keys.

MOTOR OFF
Turns cassette motor of (ie returns to remote control mode).

OPEN "I",#−1, "file name"
Used before inputing data from a cassette file, and opens an input channel to cassette recorder. The file must be closed after end of file is reached.
eg
OPEN "I",#−1,"GAME A"

OPEN "O",#−1, "filename"
Used before outputing (ie creating data file) data to cassette recorder. This creates a data file whose name is given by file name. When inputing (ie OPEN "I") the same filename must be used.
eg
OPEN "O", #−1, "GAME A"

PRINT #−1, print list
Writes the data representing print list to a file named by OPEN "O" statement on cassette recorder.
eg
10 OPEN "O",#−1,"GAME A"
20 PRINT #−1, A,B

SKIPF "file name"
Skips to end of the specified program. Omitting file name, will make the cassette head SKIP to the end of next program on the tape.
eg
SKIPF " "
SKIPF "GAME A"

SECTION 8
PRINTER CONTROL STATEMENTS

LLIST n-m
Prints lines of program from line n to line m on the printer. Omitting n and m will cause the entire program to be printed on the printer.
eg
LLIST prints entire program
LLIST 100- prints all the lines from line 100
LLIST 10-200 prints from line 10 to line 200

OPEN "O",#−2,"file name
Used before outputting data (results) to printer, and opens output channel to printer.
eg
10 OPEN "O",#−2, "GAME A"

PRINT #−2, output-list
Outputs data representing output list to printer. Channel must be opened before outputing.
eg
```
100  OPEN "O",#−2, "GAME A"
110  PRINT #−2, A,B
```

PRINT #−2, USING "format"; print-list
Used to output results on printer in the form specified by format. The format is as in PRINT USING statement.
eg
```
PRINT #−2, USING "###·##"; A, B+0·6556
```

SECTION 9
ERROR CODES

10
Dividing by 0. Attempting to divide a number by zero.

AO
Already Open. Trying to open a file that is already open. Normally caused by stopping a program before the end of an opened file is reached, and executing the program again will produce this error. Use direct command close #−1 to close the file.

BS
Bad Subscript. Value of the subscript is greater than the declared dimension.

CN
Can Not Continue. Trying to continue execution of program (ie using CONT) after program has reached the end. CONT can only be used as result of pressing [BREAK] key or using a STOP command in the program.

DD
Trying to redimension an array. Arrays can only be dimensioned once.

DS
Direct Statement. This type of error is normally caused by trying to CLOAD a data file.

FC
Illegal Function Call. The specified parameters are too large or of wrong variable type.

FD
Faulty Data. Trying to assign string data to numeric variable or vice versa, when loading data from a data file.

FM
Bad File Mode. Trying to output to a file that is designated as an input file, or to input to a file that is designated as an output file.

ID
Illegal Direct statement. Caused by using a statement as a direct command, where it can only be used as a statement in program, eg using INPUT with no line number will give ID error.

IE
Attempting to input data after the end of the specified data file. The statement IF EOF(−1) is used to avoid such error.

IO
Input/Output error. Caused by several factors, such as:
1 cassette tone and volume not adjusted properly (adjust tone to ¾ of maximum and volume to just above ½ of maximum).
2 The cassette head is not positioned at the start of program.

LS
String too Long. Trying to input a string that is more than 255 characters long (ie maximum allowable string is 255 characters).

NF
NEXT without FOR. This occurs when the FOR statement of a FOR . . .NEXT loop is missing.

NO
File Not Open. Trying to read or write a data file that is not opened.

OD
Out of Data. Trying to read more data when the data block pointer has reached the end of data.

OM
Out of Memory. All available RAM is being used or has been reserved. Use pclear to release some pages of graphics memory that are not required.

OS
Out of String space. Use CLEAR n statement to reserve more string storage if available.

OV
Overflow. Number too large to be handled with computer. maximum value $= \pm 10^{38}$

RG
Return without GOSUB. Caused by using a GOTO statement which branches into a subroutine.

SN
Syntax error. Caused by typing errors or incorrect punctuations.

ST
String formula (expression) too complex. To avoid, break the expression into smaller expressions.

Tm
Type Mismatch. Caused by assigning string data to numeric variable or by assigning a numeric data to a string variable.

UL
Undefined Line. Caused by branching to a line which does not exist.

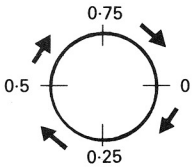
SECTION 10
GRAPHIC STATEMENTS

CIRCLE (X,Y), rad, colr, hwr, start, end
Draws a circle of radius given by rad, and whose centre is at point (X,Y). The remaining parameters are optional, and are usually used to draw ellipses, semicircles, arcs, etc.

The colour gives the code of the colour that the circle will be drawn in, if omitted, foreground colour is used.

The hwr gives the height to width ratio of the radius, and is used to draw squashed circles or ellipses. This is done by providing a value for the hwr other than 1.

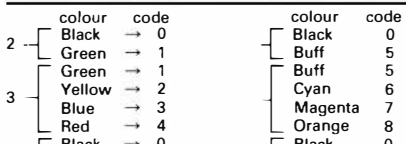
The start and end parameters are used to produce arcs and semi-circles. These two parameters can have values:
0, 0.25, 0.5, 0.75



eg
10 CIRCLE (100,85),50,3 draws a blue circle of radius 50, centre at X=100 and Y=85
20 CIRCLE (100,85),50,3,2 draws an ellipse centred at given coordinates, whose horizontal radius is 100(2×50), and vertical radius is 50 (ie ratio is 2:1)
30 CIRCLE (100,85),50,3,1,0,0.5 draws a horizontal semi-circle below the diameter (ie ∪)

COLOR foreground, background
Sets the colours to be used for the foreground and background from the available colour set.

MODE	colour set 0 screen 1,0 colour	code	colour set 1 screen 1,1 colour	code
0	Black →	0	Black	0
	Green →	1	Buff	5
1	Green →	1	Buff	5
	Yellow →	2	Cyan	6
	Blue →	3	Magenta	7
	Red →	4	Orange	8



eg
20 colour 5,7

DRAW "string"
Draws lines according to commands contained in the specified string. The draw commands are as follows:

M Xcord, Ycord
Moves to the specified point, drawing a line from last position to the new position.

Un
Moves Up n points

Dn
Moves Down n points

Ln
Moves Left n points

Rn
Moves Right n points.

En
Moves 45° n units. °=degrees

Hn
Moves 135° n units.

Gn
Moves 225° n units.

Fn
Moves 315° n units.

AK
Changes Angle of drawing. K=0=0°, K=1=90°, K=2=180°, K=3=270°.

Cn
Changes the Colour of a point to the colour specified by n (0-8).

Sn
Changes the scale of drawing to n/4, where n is a value between 1 and 62.

N
No update of position at the end of the line. ie the end of the line is not used as the new position.

B
No line will be drawn by the next move. The N and B commands can be used anywhere in the draw string. For example BM50,50, will move to position X=50, Y=50 without drawing a line from the last position to the new position.

eg
DRAW "BM100,100;U40E30F30D40L40"
DRAW "BM150,135;NU40ND40NR. NL"

GET (X_1, Y_1) – (X_2, Y_2), array name, G
Reads an area of the screen enclosed by the rectangle whose coordinates are given by (X_1, Y_1) and (X_2, Y_2) into the specified array. The (X_1, Y_1) gives the top left coordiantes of the rectangle and (X_2, Y_2) gives the bottom left coordinates. The G parameter is optional and determines the amount of picture stored, it is necessary to include G when using PMODEs 0, 1 and 3.

eg
GET (100,100)–(150,135),A,G

LINE (X_1, Y_1)–(X_2, Y_2),a,b
Draw lines from point (X_1, Y_1) to (X_2, Y_2). If (X_1, Y_1) is omitted, last end point is used as the start point. The a) parameter must be either PSET or PRESET. If PSET is used, the line is drawn in the current foreground colour. If PRESET is used, then the line is drawn in the background colour. The b) parameter is optional. If used it can be either B (a rectangle is drawn) or BF (a rectangle is drawn and filled with foreground colour).

eg
LINE (100,100)–(150,135),PSET, BF
LINE (100,100)–(150,135),PSET, B
LINE (10,10)–(35,35),PSET

PAINT (X,Y), C_1,C_2
Paints picture starting at point given by (X,Y) with colour specified by C_1 and stopping at the border of the drawing whose colour is given by C_2.

eg
PAINT (100,125),4,1

PCLEAR n
Reserves specified number of graphics memory pages. (n has value of between 1-8). The default value of n is 4.

PCLS C
Clears the high resolution graphics screen to colour specified by C (0-8).

PCOPY P_1 TO P_2
copies the contents of page P_1 to Page P_2 (P_1 & P_2 must be a number between 1 and 8, and should refer to pages reserved with PCLEAR command). This is used to produce a number of displays all of which are based on the original display.

PMODE mode, start page
Determines the high resolution modes (0-4). The value of start page determines which page (1-8) in memory will be affected.

PMODE	no. of pixels	pages used	screen 1,0	screen 1,1
0	128×96	1	Black, Green	Black, Buff
1	128×96	2	Green, Yellow, Blue, Red	Buff, Cyan, Magenta, Orange
2	128×192	2	Black, Green	Black, Buff
3	128×192	4	Green, Yellow Blue, Red	Buff, Cyan Magenta, Orange
4	256×192	4	Black, Green	Black, Buff

PRESET (X, Y)
Resets the specified high resolution graphics point to background colour.

PSET (X, Y, C)
Sets the specified high resolution graphics point to the specified colour.

PUT (X_1, Y_1)–(X_2, Y_2), array name, action
Puts the contents of the specified array as graphics points into the rectangle whose top left hand point is given by (X_1, Y_1) and the bottom right hand point is given by (X_2, Y_2). The parameter 'action' is optional and if specified, must be one of the following words:

PSET
Sets points to their original colour.

PRESET
Resets each point that is set in the original picture (ie reserves the foreground and background colours.)

AND
Sets a point if the point is already set and was set in the original picture (ie if source AND destination points are both set it sets that point).

OR
Sets a point if the point is already set or was set in the original picture.

NOT
Reverses each point in the display area.

RESET (X,Y)
Sets low resolution graphics cell at (X,Y) to background colour.

SCREEN type, colour set
Selects screen type (see COLOR command)
type=0, for text
type=1, for graphics

SET (X,Y,C)
Sets low resolution graphics cell at (X,Y) to the specified colour(C).

The authors gratefully acknowledge the permission of Dragon Data Ltd. to include the copyright material from their *User Guide* pages 136 and 143

First published in Great Britain in 1984
by Century Communications Ltd
Portland House, 12-13 Greek Street,
London W1V 5LE

ISBN 0 7126 0352 2

Printed in Great Britain