A WORLD TO EXPLORE A LEARNING

Logo is the computer language which children can use. It is not a computer game, but an ingenious educational aid that will stimulate and stretch the minds of children from as young as four years old.

At the same time, working with Logo is fun. It combines the basic concepts of geometry, language and numbers with musical sound and colourful displays to provide an exciting learning environment which children find totally absorbing. The system encourages the child to experiment, which stimulates imaginative and logical thinking, and in the process it introduces young minds to the creative and practical process of writing computer programs.

In addition to developing an awareness of geometrical shape and providing limitless scope for exciting designs, Logo introduces numerical concepts which help children to use numbers purposefully and with understanding. A third important educational feature of Logo is the facility to play with words, through which techniques for exploring language can be practised.

Acornsoft Logo is the fullest possible version of this exciting computer language, available for both the BBC Microcomputer and the Acorn Electron.

LOGO helps children learn to think logically

LOGO develops language and number skills

Logo in the classroom. Acornsoft Logo provides an educational environment that children find irresistible. Working with Logo teaches them a wide variety of skills basic to literacy and numeracy as well as providing limitless scope for imaginative design. Sound, colour, words and numbers combine to educate the child in a way that makes learning fun, while the system also gives children a valuable beginning in the world of computer technology.

The floor turtle, which plots drawings or designs according to commands from the workstation, adds a further exciting dimension to the potential of Acornsoft Logo as an educational aid.

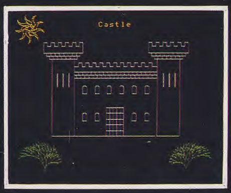
Logo in the home. Logo is as relevant in the home as it is in the classroom. Used as a system for creative play it provides an educational microworld that fascinates the whole family. In addition, Logo in the home gives children the opportunity to further explore the possibilities discovered at school.

1, 2 & 3. The turtle is the triangular cursor which moves around the screen to plot images. This is the friendly character at the heart of Logo's drawing facility. Images are built on the screen by writing simple programs which tell the turtle which way to move and as the turtle travels it leaves a trail behind it. As you can see

LOGO enables a child to practise at home what was learned at school

Every effort is made to ensure that the information in this poster is correct, but we reserve the right to make alterations at any time.

1



T SCHOOL AND AT HOME WITH LOGO

here, turtle graphics can be used to draw just about anything.

Once it is programmed to produce a particular geometrical shape, Logo can be told to repeat that shape over and over to produce developing patterns, such as spirals. A process called 'recursion' allows a modified version of the same procedure to be put to work in producing more representational figures, like the trees shown in picture 1.

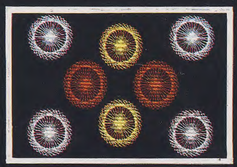
The child can add the finishing touch to a picture by giving it a title, because text can be incorporated anywhere on the screen.

4. Here it is possible to see that the size, shape and colour of the turtle can be altered and that animated shapes can be produced.

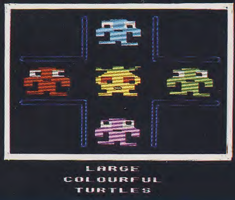
5. As many as 32 turtles can be employed together at one time by writing a simple program to HATCH as many as required. Each turtle is given its instructions by the command TELL and they go to work to produce patterns of limitless possibility. The curves and loops shown here are being generated by simple SIN and COS operations.

LOGO helps children learn elementary programming skills

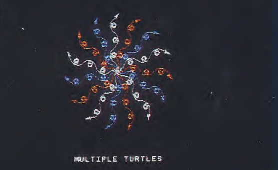
3



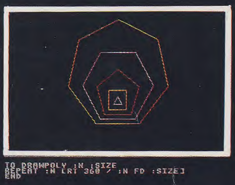
4



5

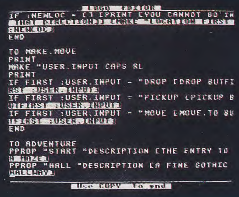


6



```
TO DRAWPOLY :N :SIZE
REPEAT :N [RT 360 / :N FD :SIZE]
END
```

7



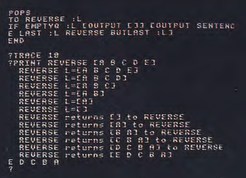
```
                LOGO EDITOR
IF :NEWLOC = [] [PRINT [YOU CANNOT GO IN
 THAT DIRECTION]] [MAKE "LOCATION FIRST
:NEWLOC]
END

TO MAKE.MOVE
PRINT
MAKE "USER.INPUT CAPS RL
PRINT
IF FIRST :USER.INPUT = "DROP [DROP BUTFI
RST :USER.INPUT]
IF FIRST :USER.INPUT = "PICKUP [PICKUP B
UTFIRST :USER.INPUT]
IF FIRST :USER.INPUT = "MOVE [MOVE.TO BU
TFIRST :USER.INPUT]
END

TO ADVENTURE
PPROP "START "DESCRIPTION [THE ENTRY TO
A MAZE]
PPROP "HALL "DESCRIPTION [A FINE GOTHIC
HALLWAY]

              Use COPY  to end
```

8



```
POPS
TO REVERSE :L
IF EMPTYQ :L [OUTPUT []] [OUTPUT SENTENC
E LAST :L REVERSE BUTLAST :L]
END

?TRACE 18
?PRINT REVERSE [A B C D E]
  REVERSE L=[A B C D E]
  REVERSE L=[A B C D]
  REVERSE L=[A B C]
  REVERSE L=[A B]
  REVERSE L=[A]
  REVERSE L=[]
  REVERSE returns [] to REVERSE
  REVERSE returns [A] to REVERSE
  REVERSE returns [B A] to REVERSE
  REVERSE returns [C B A] to REVERSE
  REVERSE returns [D C B A] to REVERSE
  REVERSE returns [E D C B A]
E D C B A
?
```

LOGO encourages children to be accurate

6. Turtle graphics provide a clear and simple way to teach the fundamentals of geometry. Logo can continue developing shapes as simple or as complex as required. Here is a program written to illustrate the relationships between the number of sides in a regular polygon and the angles which occur in it.

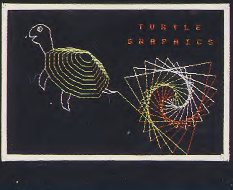
7 & 8. The Logo Editor can be used to change one, several or all procedures at once, using simple commands. The other screen here illustrates Logo's powerful trace facility which is invaluable for locating any mistakes which may have occurred during programming. Sixteen different levels of tracing allow procedure calls, statements and/or assignments to variables to be listed as they are carried out.

Acornsoft Limited
Betjeman House
104 Hills Road
Cambridge CB2 1LQ

Telephone
(0223) 316039

NSOFT

2



APP12

Optimus Graphic Design Limited, Cambridge

LOGO gives children an exciting introduction to modern technology

ACORNSOFT LOGO IS

ARITHMETIC
ASN <number> Returns the angle (in degrees) whose sine value is <number>.
ATN <number> Returns the angle (in degrees) whose tangent is <number>.
COS <number> Returns the cosine of <number> degrees.
DECS Returns the number of decimal places currently being worked to.
EXP <number> Returns the exponential function of <number>.
HEX <hexword> Returns the decimal value of <hexword>.
HIBYTE <integer> Returns the high byte of the 2-byte value <integer> ie QUOTIENT <integer> 256.
INT <number> Returns the integer part of <number>, any decimal part being stripped off.
LOBYTE <integer> Returns the low byte of the 2-byte value <integer> ie REMAINDER <integer> 256.
LN <number> Returns the natural logarithm of <number>.
PI Returns the value of pi.
†PRODUCT <number1> <number2> . . . Returns the product of the numbers input.
QUOTIENT <number1> <number2> Returns the integer part of <number1>/<number2>. If <number2> is zero an error is generated.
RANDOM <integer> Returns a random non-negative integer less than <integer>.
REMAINDER <number1> <number2> Returns the remainder of <number1>/<number2>. If <number2> is zero an error is generated.
RERANDOM <integer> Seeds the random number generator with <integer> to produce a repeatable sequence of random numbers. If no parameter is given then a random value is used to seed it.
ROUND <number> Returns the value of <number> rounded to the nearest integer.
SETDECS <integer> Controls the handling of numbers by setting the number of decimal places to <integer> if <integer> is in the range 0 to 8.
SIN <number> Returns the sine of <number> degrees.
SQRT <number> Returns the square root of <number>.
†SUM <number1> <number2> . . . Returns the sum of the numbers input.
TAN <number> Returns the tangent of <number> degrees.
+ Adds the numbers on either side and returns result.
- Subtracts the number on the right from the number on the left and returns result.
* Returns the product of the numbers on either side.
/ Divides number on left by number on right and returns result.
> Returns TRUE if the number on the left is greater than the number on the right and FALSE otherwise.
< Returns TRUE if the number on the left is less than the number on the right and FALSE otherwise.
= Returns TRUE if the objects on the left and right are equal and FALSE otherwise.

COMMENTS
\ Causes the rest of the line to be treated as a comment.

DEBUG
TC Shows the chain of current procedure calls along with their inputs.
TRACE <integer> Tells the system to trace parts of the program:
TRACE 1 traces every line
TRACE 2 traces every procedure call
TRACE 4 traces every primitive and buried procedure
TRACE 8 pauses between trace messages
These can be combined.

DEFINING and ERASING
BURY <name or list> Prevents the procedure(s) specified being listed, edited or saved.
BURYALL Prevents all procedures being listed, edited or saved.
COPYDEF <newname> <fromname> Copies the definition of the procedure <fromname> and calls it <newname>.
DEFINE <name> <list> Allows you to write procedures that define other procedures, <name> is the procedure to be defined; <list> helps with the definition and consists of a series of sublists.
EDALL Edits all procedures and names in workspace.
EDIT (ED) <procname or list> Puts the procedure(s) specified into the edit buffer so allowing you to edit them. If the input is absent the current contents of the edit buffer will be displayed.
EDN <varname or list> Edits the variable(s) specified.
EDNS Edits all the variables in the workspace.
EDPS Edits all the procedures in the workspace.
END Defines the end of a procedure.
ERALL Erases all procedures and variables from the workspace.
ERASE (ER) <procname or list> Erases the most recent invocation of the procedure(s) specified from the workspace.
ERN <varname or list> Erases the most recent invocation of the variable(s) specified from the workspace.
ERNS Erases all invocations of all variables from the workspace.
ERPS Erases all procedures from the workspace.
NOREDEF Prevents primitives being redefined.
REDEF Allows primitives to be redefined.
REDEFQ Returns TRUE if primitives can currently be redefined and FALSE otherwise.
TEXT <procname> Returns the definition of <procname> as a list of lists.
TO <procname> <parameters> Tells Logo that you are defining a procedure <procname> which has the inputs <parameters>.
UNBURY <procname or list> Allows the procedure(s) specified to be listed, edited or saved.
UNBURYALL Allows all procedures in workspace to be listed, edited or saved.

EDITING COMMANDS
arrow keys Allow the cursor to be moved around the screen.
CTRL/FUNC left Moves the cursor to the start of the current Logo line.
CTRL/FUNC right Moves the cursor to the end of the current Logo line.
CTRL/FUNC up Moves the cursor to the top of text.
CTRL/FUNC down Moves the cursor to the bottom of the text.
DELETE Deletes the character before the cursor.
CTRL/FUNC D Deletes the character at the cursor position.
CTRL/FUNC U Deletes the current Logo line.
CTRL/FUNC L Deletes from the current cursor position to the end of the current Logo line.
CTRL/FUNC N Inserts a new line below the current cursor position.
COPY Exits from the editor, preserving any changes made.
ESCAPE Exits from the editor without altering the original procedure(s)/name(s).

FILES
CAT Catalogues the current filing system.
ERFILE <filename> Deletes <filename> from the current filing system.
LOAD <filename> Loads the contents of the file <filename> into your workspace.
READPICT <filename> Copies the picture in the file <filename> on to the screen, changing the screen mode, number of lines of text, palette and type of screen if necessary.
SAVE <filename> <procname or list> Creates the file <filename> and saves into it all variables and property lists held in your workspace. If the second input is present then the procedures specified will be saved, otherwise all procedures will be saved. If you call a procedure LOADINIT and save it, then when it is loaded again it will be executed automatically.
SAVEPICT <filename> Creates the file <filename> and saves into it the graphics part of the screen.

FLOOR TURTLES
BACK (BK) <number> Moves the turtle back by <number> steps.
EXPLORE <number> Moves the turtle forward by <number> steps or until it hits something and returns the number of steps which it managed to cover.
FLOOR Tells Logo that all subsequent commands apply to the floor turtle rather than the screen turtle.
FORWARD (FD) <number> Moves the turtle forwards by <number> steps.
HOOT Generates a brief sound from the turtle's speaker, if one exists, otherwise causes a BEEP from the computer.
LEFT (LT) <number> Turns the turtle left by <number> degrees.
PENDOWN (PD) Lowers the pen so that the turtle leaves a trail behind it when it moves.
PENUP (PU) Lifts up the pen so that the turtle does not leave a trail behind it when it moves.
PENUPQ Returns TRUE if the turtle's pen is up and FALSE otherwise.
RIGHT (RT) <number> Turns the turtle right by <number> degrees.
SCREEN Tells Logo that all subsequent commands apply to the screen rather than the floor turtle.
SCREENQ Returns TRUE if the screen turtle is in use and FALSE otherwise.
SENSE <number> Returns the value TRUE if the turtle sensor <number> is touching anything and FALSE otherwise.

KEYBOARD
CI Clears the keyboard input buffer. Any keys pressed before CI is issued will be forgotten.
INKEY <integer> If <integer> is in the range 0 to 3276 INKEY waits for that number of tenths of seconds or until a key is pressed. If no key was pressed, the empty word is returned; if a key was pressed the one-character word CHAR <code> is returned, where <code> is the ASCII value of the key. If <integer> is negative a specific key is tested and the value TRUE returned if that key is currently pressed, and FALSE otherwise.
KEYQ Returns the value TRUE if a key has been pressed and its value has not been used by RC, READWORD or READLINE, and FALSE otherwise.
RC Reads the next character from the keyboard; waits for one to be typed if none is available.
READLIST (RL) Reads the following line from the keyboard in the form of a list.
READWORD (RW) Reads the first word of the line entered from the keyboard.

LOGICAL
†ALLOF <t/f> <t/f> . . . Returns TRUE if all expressions are true and FALSE otherwise.
†ANYOF <t/f> <t/f> . . . Returns TRUE if at least one of the expressions is true and FALSE otherwise.
NOT <t/f> Returns TRUE if the expression is false and FALSE if the expression is true.

MANY TURTLES
ALIVEQ <integer> Returns TRUE if turtle <integer> is 'alive' and FALSE otherwise.
FORGET <integer or list> Deletes the turtle or turtles specified from the list of turtles currently 'alive'. TURTLE 0 cannot be deleted.
†HATCH <integer or list> <integer2 or list2> Creates the turtle or turtles with the numbers given by the first input at the current turtle position, with the shape of the current turtle or with a shape given by SETSH of the second input if one is given. Each input must be different from all identifiers of currently 'alive' turtles and must be in the range 1 to 32.
TELL <integer or list> Determines which turtles all the subsequent primitives will apply to. The effect of TELL is cancelled by another TELL.
TURTLES Returns a list of all turtles currently alive.
WHO Returns a list of all turtles currently being talked to.

MISCELLANEOUS
CALL <address> Calls the machine code routine at <address>. On entry to the machine code, the A, X and Y registers are set up from bytes 0, 1 and 2 respectively of DATAAREA. On return bytes 0 to 3 are set up from the A, X, Y and P flags/registers respectively.
DASIZE Returns the size of the data area in bytes.
‡DATAAREA <integer> Returns the byte address of a data area reserved by Logo of size <integer> bytes.
DEPOSIT <address> <byteinteger> Places the value <byteinteger> in the location with address <address>.
EXAMINE <address> Returns the contents of the location <address>.
HIBYTE <integer> Returns the high byte of the 2-byte integer value <integer> ie QUOTIENT <integer> 256.
LOBYTE <integer> Returns the low byte of the 2-byte integer value <integer> ie REMAINDER <integer> 256.
OSBYTE <integer> <integer> <integer> Calls the operating system general purpose routine with the A register and optionally the X and Y registers. The contents of the X and Y registers are returned as the low and high byte of the result.

OTHER INPUT AND OU
ADVAL <integer> If <i
4 it returns the value of th
converter channel.
BEEP Generates a brief s
loudspeaker.
BUTTONQ <integer> I
the button on the appropr
pressed and FALSE other
or 2 then an error is gener
ENVELOPE 14*<integ
and pitch of sounds creat
operation.
PRSCREEN Copies the c
the printer unless the scre
which case it does nothin
SOUND <channel> <a
<duration> Produces a
loudspeaker.
TIME Returns the time in
the computer was switch
operation was last used. T
zero at 26214 (approx 43 m
TIMERESET Resets the t
WAIT <tenths of secs>
running for the number of
or until ESCAPE is presse
WS Returns a list of two in
total number of bytes free
the maximum workspace
individual item.

PROGRAM CONTROL
BREAK Breaks out of REF
loops.
CATCH <catch label>
THROW <catch label> is
execution, control returns
<catch label>.
CATCH "TRUE <list> C
CATCH "ERROR <list>
suppresses error message
CATCH "ESCAPE <list
ESCAPE key.
CONTINUE (CO) Resum
has been executed or ESC
DOFOREVER <list> Re
until a BREAK, LOOP, OU
encountered, an error occ
executed and moves cont
ERRMSG <list> Prints t
message when <list> co
form given by ERROR.
ERROR Returns informat
occurred whilst a CATCH
information is in the form
the error number and the t
error or empty lists if non-
GO <label> Transfers co
following <label> in the s
IF <t/f> <list1> <list2
TRUE then <list1> is exe
is executed if it exists.
IFFALSE <list> If the re
TEST in the current proce
is executed otherwise it d
IFTRUE <list> If the res
TEST in the current proce
executed otherwise it doe
LABEL <label> Used in
primitive – GO <label> p
instruction following <lab
LOOP Returns control to t
REPEAT or DOFOREVER
REPEAT, increments the
OUTPUT (OP) <object>
when control is passed ba
which called it.
PAUSE Suspends the exe
until CONTINUE is typed i
instructions to debug your
REPEAT <integer> <li
<integer> times unless in
DOFOREVER.
RUN <list> Runs <list>
in directly.
SETERR <list> Regener
been trapped by CATCH "
to take action about it you
STOP Is only allowed with
the procedure and returns
which it was called.
TEST <t/f> Tests wheth
TRUE or FALSE and reme
subsequent IFFALSE and
THROW <catch label>
primitive to dictate contro

AT YOUR COMMAND

TPUT
nteger> is between 1 and
: analogue to digital

und from the machine's

eturns the value TRUE if
ate joystick is being
vise. If <integer> is not 1
ted.
er> Controls the volume
d with the SOUND

ntents of the screen to
en is in modes 3, 6 or 7 in
.
nplitude> <pitch>
ound from the internal

tenths of a second since
d on or the TIMERESET
he time 'wraps round' to
inutes 41.44 seconds).
me counter to zero.
Stops the program
tenths of a second input
l.
tegers, the first being the
n workspace, the second
available for one

EAT or DOFOREVER

<list> Runs <list> and if
called during its
to the primitive after

atches any THROW.
Catches errors and
s.
> Catches any use of the

es running after a PAUSE
APE has been pressed.
peats <list> forever or
TPUT or STOP is
irs or a THROW or GO is
ol out of the list.
ne appropriate error
tains information in the

on about an error that has
'ERROR is in effect. The
f a list with two items,
wo parameters of the
xistent.
ntrol to the instruction
ame procedure.
> If the expression is
cuted otherwise <list2>

ult of the most recent
lure was FALSE, <list>
es nothing.
lt of the most recent
lure was TRUE, <list> is
s nothing.
conjunction with the GO
sses control to the
el>.
he beginning of the
ist and, in the case of
epeat count.
Returns <object>
k to the procedure

ution of a procedure
n, allowing you to enter
procedure.
st> Runs <list>
structed otherwise as in

as if it were being typed

ates an error which has
ERROR if you decide not
self.
in a procedure. It stops
control to the point at

r the expression is
nbers the result for
FTRUE instructions.
s used with the CATCH
during execution.

THROW "LEVEL Returns control to the most recent command level.
THROW "TOPLEVEL Returns to the highest command level.
TIDY Forces a garbage collection to be carried out.

PROPERTY LISTS
ERPLIST <name or list> Erases the property name(s) specified, along with their properties.
ERPLISTS Erases all property names and their properties.
GPROP <name> <propname> Returns the value associated with a specific property <propname> of the word <name>. If there is no such <name> or no such property of <name> it will return the empty list.
PLIST <name> Returns the property list of the word <name>, if there is no such property list it will return the empty list.
PPALL Prints the property list of every name.
PPROP <name> <propname> <word or list> Gives the word <name> a specific property <propname> with the value <word or list>.
PPS <name or list> Prints the property list(s) associated with the name(s) specified.
REMPROP <name> <propname> Removes the property <propname> from the property list of the word <name>.

SCREEN
CT Clears the text area of the screen and puts the cursor at its top left hand corner.
CURSOR Returns the text cursor position as a list of its x and y coordinates.
MODE Returns the current screen mode.
PAL <integer1> <integer2> Sets the logical colour <integer1> to the physical colour <integer2>.
PM <integer> Ensures that sufficient space is reserved in memory for you to be able to change to screen mode <integer>.
†PRINT (PR) <word or list> . . . Outputs the word(s) specified at the text cursor position, separated by spaces and followed by a carriage return.
SCR Returns the value of the screen's aspect ratio.
SETCURSOR <list> Places the text cursor at the position represented by <list>, which consists of the column number followed by the line number.
SETMODE <integer> Changes the current screen mode to MODE <integer>.
SETSCR <integer> Sets screen aspect ratio to <integer>.
SHOW <object> Prints the contents of <object> on the screen, followed by a carriage return.
TS Reserves the entire screen for text and clears it.
†TYPE <word or list> . . . Outputs the word(s) specified at the text cursor position. It does not insert spaces between them nor a carriage return at the end.
†VDU <number> or "; or <list> . . . Allows you to send control codes to the VDU driver.

SCREEN PRINT
PO <procname or list> Prints the definition of the procedure(s) specified.
POALL Prints the definition of every procedure and the contents of every variable that is currently in your workspace.
PONS Prints the name and value of every variable that is currently held in your workspace.
POPS Prints out the definition of every procedure in your workspace
POTS Prints out the title line of every procedure in your workspace.

SPECIAL WORDS
"ERROR "ESCAPE
"FALSE "TRUE
"LEVEL "TOPLEVEL

TURTLE GRAPHICS
BACK (BK) <number> Moves the turtle backwards by <number> steps.
BG Returns an integer which represents the logical background colour.
CLEAN Clears the graphics area, leaving the turtle where it is.
CS Clears the graphics area and returns the turtle to the centre of the screen.
DISTANCE <list> Returns the distance from the current turtle position to the point on the screen addressed by <list> which is in the form [x,y].
DOT <list> Returns an integer which represents the colour of the dot at the position specified by <list> which is in the form [x,y].
DRAW <integer> Resets the screen and reserves <integer> lines at the bottom of the screen for text (the default being 6).
FENCE Sets a fence around the graphics area and displays an error message if the turtle hits it.
FORWARD (FD) <number> Moves the turtle forward by <number> steps.
‡HEADING <integer> Returns the direction in which the turtle <integer> is pointing in degrees.
HIDETURTLE (HT) Hides the turtle from view until SHOWTURTLE is used.
HOME Returns the turtle to the centre of the screen, leaving a track if the pen is down.
LEFT (LT) <number> Turns the turtle left by <number> degrees.
‡PC <integer> Returns an integer which represents the current pen colour of turtle <integer>.
PE Tells the turtle to erase all lines over which it passes as it moves. The eraser can be removed by using PENDOWN, PENUP, PENRESET or PX.
‡PEN <integer> Returns the current pen parameters of turtle <integer> in the form of a list:
penstate – either PU, PD, PE or PX
shown – TRUE if turtle is visible, FALSE otherwise
colour – pen colour
nib – current graphics option
pentype – colour option
PENDOWN (PD) Tells the turtle to draw lines when it moves.
PENRESET Resets the turtle state, so that the turtle is shown, the pen is down, colour is 7, nib is 8 and pen type is 0.
PENUP (PU) Lifts the turtle's pen up so that no lines are drawn when it moves.
PENUPQ Returns TRUE if the turtle's pen is up and FALSE otherwise.
‡POS <integer> Returns the position of turtle <integer> in the form of a list.
PX Sets a reversing pen.
RIGHT (RT) <number> Turns the turtle right by <number> degrees.
SECT <number1> <number2> <number3> Draws a sector through angle <number2> with radius <number1> and thickness <number3>.
SETBG <integer> Sets the background to the colour represented by <integer>.
SETDOT <list> Puts a dot at the position represented by <list> which is in the form [x,y], in the current pen colour and without moving the turtle.
SETHEADING (SETH) <number> Turns the turtle so that it is pointing in the direction <number> degrees.
SETNIB <integer> Sets the BASIC PLOT code value to <integer> to give dotted lines, triangles etc.
SETPC <integer> Changes the logical pen colour to the colour represented by <integer>.
SETPEN <list> Sets the pen state to the condition determined by <list> which has five parameters: penstate, shown, colour, nib and pentype.
SETPOS <list> Moves the turtle to the position specified by <list> which is in the form [x,y].
SETPT <integer> Defines the way in which colours are to be used, eg Exclusive-ORed or ANDed on to the screen.
SETSH <integer or list> Allows the turtle to be redefined by sending one or a list of VDU commands describing what you want it to be.
SETX <number> Moves the turtle horizontally to the point with the x-coordinate <number>.
SETY <number> Moves the turtle vertically to the point with the y-coordinate <number>.
‡SH <integer> Returns the list of VDU parameters which define the current shape of turtle <integer>.
SHOWTURTLE (ST) Makes the turtle visible.
STAMP Causes an image of the turtle to be left on the screen at its current position.
†TITLE <word or list> . . . Prints the object(s) you give it at the current turtle position.
TOWARDS <list> Returns a value which indicates the heading needed to make the turtle face the position given by <list> which is in the form [x,y].
WINDOW Turns the screen into a window which shows only part of the field in which the turtle can move. If the turtle moves out of this window it will still move as instructed but will not be visible.
WRAP Places a fence around the screen so that when the turtle hits the fence it reappears on the opposite side of the screen.
‡XPOS <integer> Returns the x-coordinate of the current position of turtle <integer>.
‡YPOS <integer> Returns the y-coordinate of the current position of turtle <integer>.

TESTS ON OBJECTS
BURIEDQ <procname> Returns the value TRUE if the procedure <procname> is buried and FALSE otherwise.
DEFINEDQ <name> Returns TRUE if <name> is the name of a procedure or primitive and FALSE otherwise.
EMPTYQ <object> Returns TRUE if <object> is the empty word of empty list and FALSE otherwise.
LISTQ <object> Returns TRUE if <object> is a list and FALSE otherwise.
MEMBERQ <object1> <object2> Returns the value TRUE if <object1> is an element of <object2> and FALSE otherwise.
NUMBERQ <object> Returns TRUE if <object> is a number and FALSE otherwise.
PRIMITIVEQ <name> Returns TRUE if <name> is a primitive and FALSE otherwise.
THINGQ <name> Returns TRUE if <name> has some value and FALSE otherwise.
WORDQ <object> Returns the value TRUE if <object> is a word and FALSE otherwise.

VARIABLES
LOCAL <name> <value> Hides any previous invocation of <name> from the current procedure or list and replaces it with a new one containing <value>. The previous value is restored on leaving the procedure or list, when THROW transfers control to a procedure at a higher level, when ERN is used to erase it or when an error is encountered.
MAKE <name> <value> Assigns the value <value> to <name>.
THING <name> Returns the contents of the variable <name>.

WORDS AND LISTS
ADDITEM <integer> <object1> <object2> Returns an object made up of <object1> with <object2> added at position <integer>.
ASCII <word> Returns the ASCII value of the first character of <word>.
BUTFIRST (BF) <object> Outputs everything except the first element of <object>. Using it on empty words and lists will generate an error.
BUTLAST (BL) <object> Outputs everything except the last element of <object>. Using it on empty words and lists will generate an error.
CAPS <object> Converts the letters of <object> to capitals.
CHAR <integer> Returns a one character word whose ASCII code is <integer>.
COUNT <object> Returns the number of elements in <object>.
ERITEM <integer> <object> Returns an object which is <object> with the element at position <integer> removed.
FIRST <object> Returns the first element of <object>. Using an empty word or list will generate an error.
FPUT <object1> <object2> Produces a new list by putting <object1> at the beginning of <object2>.
ITEM <integer> <object> Returns the element in position <integer> of <object>. If the <integer>th element doesn't exist then an error is generated.
LAST <object> Returns the last element of <object>. Using an empty word or list will generate an error.
†LIST <object> <object> . . . Returns a list whose elements are the objects specified.
LPUT <object1> <object2> Produces a new list by putting <object1> at the end of <object2>.
MEMBER <object1> <object2> If <object1> is an element of <object2> it returns the element number, otherwise it returns zero.
†SENTENCE (SE) <object> <object> . . . Combines the objects specified to form one list.
SETITEM <integer> <object1> <object2> Returns an object which is <object1> with element <integer> changed to <object2>.
†WORD <word> <word> . . . Returns a word that is built up from the words specified.

† The inputs to these primitives may be repeated one or more times.
‡ If the input shown is used then the primitive and the input must be enclosed in brackets. The input defaults to 0.