18 GOTO and GOSUB

There are four more instructions in Electron BASIC which can be used to tell the computer to continue executing the program at specified points.

These are:

GOTO
GOSUB . . .RETURN
ON . . . GOTO
ON . . . GOSUB

GOTO

The simplest of these instructions is GOTO.

```
10 PRINT "SCREENFUL"
20 GOTO 10
```

Each time the computer executes line 20 it is sent back to line 10 once again. This program never ends: it is a continuous loop. To stop the program you may press either **ESCAPE** or **BREAK**. If you press **ESCAPE** a message is printed giving the line number at which execution ceased.

GOTO instructions may send control of the program either forwards or backwards, but you must be careful not to use too many **GOTO** loops - they soon become impossible to follow, and very difficult to correct when a program does not function as you wish it to. It is far better to use procedures or **REPEAT . . . UNTIL** statements where possible.

GOSUB . . . RETURN

GOSUB stands for ‘Go To Subroutine’, and is really just a variation of **GOTO**. It is strongly advised that you use the more readable and more flexible procedure instead of **GOSUB**. It is used when a particular routine is used several times in different parts of the same program. for exatupk, to read a key.

It is most useful with **IF** statements.

Here is a game which requires you to put a set of numbers in sequence. The **GOSUB** routine is called from various parts of the program, and has the effect of swapping the numbers around according to which key you press.

```
10 REM SWAP-ROUND
20 MODE 6
30 VDU23,1,0;0;0;0;
40 answer$="123456789"
50 number$=answer$
60 INPUT TAB(8,16)"Difficulty level",level
70 FOR I=1 TO level
80 position=RND(8)+1
90 GOSUB 220
100 NEXT
110 CLS:PRINT TAB(15,10);number$
120 PRINT TAB(6,16)"Press a key between 2
and 9"
130 REPEAT
140 position=VAL GET$
150 IF position <2 OR position>9 THEN GOTO
140
160 GOSUB 210
170 PRINT TAB(15,10)number$
180 UNTIL number$=answer$
190 PRINT TAB(6,16);SPC(9);"Well done";SPC(
11);"END"
200 END
210 temporary$=""
220 FOR J=position TO 1 STEP-1
230 temporary$=temporary$+MID$(number$,J,1)
240 NEXT
250 number$=temporary$+MID$(number$,positio
n+1)
260 RETURN
```

As you can see, **GOSUB** differs from **GOTO** in that the program flow must always **RETURN** to the position following the subroutine call

Just one point about **GOSUB**.

As with **FOR . . . NEXT**, you should not jump out of a subroutine by using **GOTO**. If the computer keeps being told to **GOSUB**, without ever encountering a **RETURN**, it will soon use up its memory.

ON . . . GOTO, ON . . . GOSUB

An instruction such as

```
ON N GOTO 100,200,70,260
```

means that the computer checks on the value of N, and then 'jumps' to the Nth line number in the list.

So, if N = I, the program 'jumps' to line 180; if N = 2 to fine 200; if N = 3 to line 70; and if N = 4 to fine 260.

ON . . . GOSUB works in exactly the same way.