13
Twister

Magazines, such as *Electron User*, often publish drawings of user-defined graphics characters. To help you, they publish the VDU 23 statement required to define these characters in the Electron's memory. Unfortunately, it may happen that a character is the 'wrong way round'. We found an excellent outline of a flying duck, which we wanted to use in a program. We wanted to make it fly from left to right across the screen. But the outline showed a duck heading to the left! We could not bear to see the poor duck flying backward across the screen, so we had to work out the VDU statements for a reversed, mirror-image ofthe duck.

Unfortunately, you cannot reverse the design by just writing out the numbers of the VDU 23 statement in the reverse order. Each value has to be worked out again. You can use the CHARACTERS program (Chapter 9), copying a reversed picture on to the screen. This is fairly quick to do, but an even quicker way is to use TWISTER.

TWISTER not only produces a mirror-image of any given user defined graphics character, but it will turn it upside down, or on its side as well. It produces all of the eight possible versions of the character (including the original), displaying them all on the screen with the VDU 23 values beside each character.

The other useful feature of TWISTER is that as well as operating on user-defined characters, it can perform the same rotating and mirror-imaging action on any of the Electron's text characters. These include all the letters, numerals and symbols that can be keyed in from the Electron's keyboard. If you want an Я or an ᴚ, TWISTER can provide it. Upside-down or mirror-image text characters may rarely be of use, but sideways text characters can be extremely handy. Many displays, particularly ofcharts, diagrams and tables are improved by having some of the text written sideways-on. Sideways text is particularly useful for labelling the vertical axis of a graph, or for putting a heading to a column that is too narrow to take horizontal text. Anyone can define their own sideways characters on paper or by

using CHARACTER, but TWISTER has the advantage that the sideways characters match the style of the normal characters of the Electron exactly. This is because they are copied from the normal characters as displayed on the screen.

Using the program

If you want the program to operate on a character which you have designed yourself, or have copied from a magazine, begin by typing in the VDU 23 statement. Do this *before* you run TWISTER. The character must be defined with character code 224. The statement required is:

```
VDU 23, 224, a, b, c, d, e, f, g, h
```

in which a to h are the values needed to produce the required design. Then press RETURN. This stores the design in the Electron's memory, where TWISTER will be able to find it.

If you are planning to use TWISTER to give you the values for the Electron's own set of characters, there is no need to type anything. Just run the program.

When the program is run, the message 'CHARACTER?' appears near the top of the screen. If you want TWISTER to operate on a character of your own that you have just keyed in, simply press RETURN. Make sure that you do not type anything else, not even a space, before pressing RETURN.

If you want TWISTER to operate on one of the Electron's characters, key in the required character, then press RETURN. Two characters that this program does not accept are the double quotes and the comma, but you are hardly likely to want to work with these two.

As soon as you have pressed RETURN, the eight possible versions of the character are displayed, together with their VDU 23 values. You then have only to copy down the values of the required version of the characters, ready for use in your own programs.

The message 'SPACEBAR FOR NEXT CHARACTER' invites you to let TWISTER operate on another of the Electron's characters. If you want it to work with another character of your own, press ESCAPE, type in the VDU 23 statement for the next character, and then RUN the program again.

Keying in

There is no space between the double-quotes in line 100. The question marks (lines 120, 150 and 500) are the Electron's indirection operator, the use of which is explained in Chapter 23 of the User Guide. Take particular care with typing the lines which contain this operator. A mistake could possibly cause the program to crash when it is first run. You might even lose the program. It is a good idea to save the program to tape *before* you run it for the first time. Then, if it does crash, you will not have to type it all in again. Just reload it and look for the typing error you have made.

The upward pointing arrow in lines 170 and 650 is the exponentiation symbol. Obtain this by pressing SHIFT with the ' ^ ~ left-arrow ' key.

Program design

20-30 Initialising.

40-80 Resetting a variable and clearing an array.

90-100 Which character?

110-130 Copying the bytes of the character on the screen from video memory to character memory (for code 224).

140-180 Reading the bits from character memory (for code 224) into array B%()

190 Display the original character, with its VDU 23 values.

200-250 Rotate character three times, store its rotated form in memory, and display it with its VDU 23 values.

260-270 Rotate it once more to make it upright again. Store this form in B%(), which is now as it was to begin with.

280-310 Produce in R%() a mirror-image of the pattern stored in B%().

320-330 Store this back into B%() and in character memory.

340 If this is the first time round, display the mirror-image, then return to 200-250 to rotate it and display it.

350-370 Inviting repeat of the program.

390-450 PROCrotate to place in R%( ) a rotated (quarter-turn) version of a pattern stored in B%( ).

460-520 PROCdisplay to display the character and its VDU 23 values.

530-590 PROCtrans to transfer a pattern from R%() to B%() without altering it.

600-690 PROCmem to transfer pattern in R%() to character memory as a pattern of bits in eight bytes.

The program

```
   10 REM ** TWISTER **
   20 MODE 4
   30 DIM B%(7,7),R%(7,7)
   40 set=0
   50 FOR J=0 TO 7
   60 FOR K=0 TO 7
   70 B%(J,K)=0
   80 NEXT:NEXT
   90 CLS:INPUT''"CHARACTER? "A$
  100 IF A$="" THEN 140
  110 FOR J=0 TO 7
  120 J?3072=?(HIMEM+728+J)
  130 NEXT
  140 FOR J=0 TO 7
  150 byte=J?3072
  160 FORK=0 TO 7
  170 IF (byte AND 2^(7-K)) THEN B%(J,K)
=1
  180 NEXT:NEXT
  190 PROCdisplay(0)
  200 FOR L=1+set TO 3+set
  210 PROCrotate
  220 PROCmem
  230 PROCdisplay(L)
  240 PROCtrans
  250 NEXT
  260 PROCrotate
  270 PROCtrans
  280 FOR J=0 TO 7
  290 FOR K=0 TO 7
  300 R%(J,K)=B%(J,7-K)
  310 NEXT:NEXT
  320 PROCtrans
  330 PROCmem
  340 IF set=0 THEN set=4:PROCdisplay(4)
:GOTO200
```

```
  350 PRINT'''TAB(3)"< SPACE BAR FOR NEX
T CHARACTER>";
  360 REPEAT KEY$=GET$:UNTIL KEY$=" "
  370 GOTO 40
  380 END
  390 DEF PROCrotate
  400 LOCAL J,K
  410 FOR J=0 TO 7
  420 FOR K=0 TO 7
  430 R%(J,K)=B%(7-K,J)
  440 NEXT:NEXT
  450 ENDPROC
  460 DEF PROCdisplay(L)
  470 LOCAL J
  480 PRINT TAB(1,8+2*L)CHR$(224+L);
  490 FOR J=0 TO 7
  500 PRINT TAB(5+4*J,8+2*L)STR$(?(3072+
L*8+J))
  510 NEXT
  520 ENDPROC
  530 DEF PROCtrans
  540 LOCAL J,K
  550 FOR J=0 TO 7
  560 FOR K=0 TO 7
  570 B%(J,K)=R%(J,K)
  580 NEXT:NEXT
  590 ENDPROC
  600 DEF PROCmem
  610 LOCAL J,K
  620 FOR J=0 TO 7
  630 byte=0
  640 FOR K=0 TO 7
  650 IF R%(J,K)=1 THEN byte=byte+2^(7-K
)
  660 NEXT
  670 ?(3072+L*8+J)=byte
  680 NEXT
  690 ENDPROC
```