9
Characters

Electron users are able to define their own graphics characters, either to produce special alphabets and symbols or, more often, to use as the basis of graphics displays in games. Designing a single character is easy, and is explained fully in Chapter 20 of the Electron's User Guide. Designing characters is fun, but you need lots of squared paper on which to work out the designs and you may need to make several attempts before you make each design perfect. Even then, the character may not be quite what you intended when you finally see it displayed in its proper size on the screen.

This program provides a large-scale display on the screen on which you draw your design. You use the editing keys, the 'arrow' keys and COPY key of the Electron, for this. The design is easily altered, too. You can change it a little at a time until it isjust right. While you are building it up as a large-scale version in the design area, you see it appearing at the bottom of the screen in its proper size. This is the size it will be when you use it in your own programs. As you work out each design, you will be able to gauge how effectivc it will be in use. When the design is ready, the program calculates all the values that are required for the VDU statement which will define the new character.

The program helps you design more thanjust a single character. To produce larger and more elaborate graphics, we often want to compose a design from two or more characters, placed side by side. As shown in Fig. 9.1, designs made from two or more characters offer much more scope for inventiveness than those made from a single character. The program provides an area on the screen where you can work with up td nine characters at once.

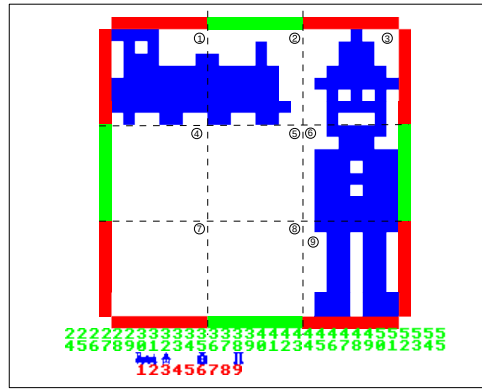

Fig. 9.1. The screen display for CHARACTERS.The dashed lines and encircled numbers are to mark out the individual characters; they do not appear on the screen.

Using the program

As soon as you run the program the display appears as in Fig. 9.1. At this stage the design area, enclosed by the red and green borders is completely blank. The borders are coloured red and green to help you see exactly where the area is sub-divided into three rows ofthree large squares. The dashed lines shown in Fig. 9.1 pick out these squares, but the dashed lines do not appear on the screen.

Below the design area is a row of green numbers. It looks like rmo rows of numbers but, if you read domn each column, you can see that these are the numbers ' 24'',25'',26'and so on, up to ' 55'.These numbers are to be thought ofas ' 224'',225'',226'up to ' 255'.The initial ' 2has been left out ofeach numb::r to save a line on the screen. This range of numbers (224 to 255) covers the 32 ASCII codes allocated for user-defined graphics in the Electron. The program allows you to define characters for any or all of these codes. Eventually, the characters you design will appear in a row below their respective numbers. You can see how this will Iook in Fig. 9.1.

The design area allows nine characters out of the available 32 to be designed at one time. When you first run the program, the message ' Whichnumber? (224-247)' will be showing at the bottom of the

screen. The computer is asking you which character number you want to be used for the top left square of the design area. You can type in any number from 224 to 247. Typing ' 224'will make the design area cover the first nine characters numbers (from 224 to 232). This may not be a good range to choose, as will be explained later. Typing ' 247' will make the design area cover the last nine characters (from 247 to 255). Or you can type a number in between these limits. In Fig. 9.1, the number typed was ' 230' making the design area cover the characters from 230 to 238. As soon as the number has been typed and RETURN has been pressed, the message disappears and is replaced by a row of red figures ' 123456789'These figures indicate which range of character numbers you have elected to work on. They also tell you which is which, for the squares in the drawing area are numbered from 1 to 9 as shown in Fig. 9.1. Note that the numbers shown in the design area of Fig. 9.1 are for reference only and do not appear on the screen.

You will now notice that the cursor has appeared at the top left corner of the design area. It is waiting for you to begin. Draw the characters by using the following keys:

The ' arrow' keys - to move the cursor around in the design area.

The space-bar - to print a white block on the screen. This is how you draw each character.

The COPY key - to delete a white block already drawn. This is how you correct mistakes.

Try moving the cursor around the area. You can wander over the whole area if you wish. There is no need to stop at the boundary between one large square and another. Do not worry about making mistakes for they are easy to correct, using the COPY key. While you are drawing, you will see miniature white designs appearing at the bottom of the screen.

Figure 9.1 shows how to draw a loi:omotive, using two characters, side by side. In this case, a small-scale version ofthe locomotive appears at the bottom of the screen. The quaint person with the pointed hat requires three characters placed one above the other. The characters which make up this person are numbers 3, 6 and 9 in the area, so he appears as 3 separate characters in the strip below.

There is one thing that you must try to avoid when using this program. Unfortunately, the BREAK key ofthe Electron is situated next to the ' arrow'keys. Since you need to use the ' arrow'keys constantly, it is only a matter of time before you press BREAK by accident. You can recover the program by typing OLD, but the display is lost. Actually, the character definitions you have already created are

nor lost. Delete line 60. The same characters will reappear in a row at the bottom ofthe screen when you rerun the program. They will not reappear in the design area. Pressing BREAK is not disastrous but is an annoyance. To avoid it, you could try placing an inverted matchbox tray over the BREAK key when you use this program.

When you have finished designing the nine characters (or as many groups of characters as can be fitted conveniently in to the area) you may wish to design more. If so, press key ' D'This clears the whole screen. The borders of the area then reappear and also the row of small-scale characters you have already designed at the bottom of the screen. You can now *add* to this range. When the ' Whichnumber?' message reappears, key in the first number ofa new range of ASCIl codes . If your first range was from 224 to 232, for example, you might key in ' 233'to cover the next nine codes from 233 to 241 . You can then use the design area, as before. If you want to change one of your earlier designs, press ' D'to, select a range to cover the character you want to change, and then redesign it completely. The strip of characters at the bottom of the screen again shows exactly what each character is like.

When you have finished designing and want to know what VDU 23 statements are to be used to produce these characters in your program, press key ' R'The screen clears and a table of values is displayed. The numbers show you what to put into each VDU 23 statement. Each character you have designed is shown down the left side of the screen, followed by its ASCII code (224-255). This is followed by a row ofeight numbers, the values which are to go into the VDU statement to define the character in the computer. For example, suppose that a row offigures in the table is:

230 240 80 113 127 255 255 255 76

The character for the ' rearhalf' of the locomotive would be to the left of this row of figures, for these are the values for character 230 of Fig. 9.1. To produce this character in a BASIC program the statement required is:

VDU 23,230,240,80,113,127,255,255,255,76

In other words, to get the VDU statement, your program line is ' VDU 23' followed by the numbers in the row of the table, with commas between them.

Depending on your TV set, you may not be able to see the top row of the table. This is the one which holds the figures for character 224. With other programs in this book we have avoided using the top line of the screen, because many users will not be able to see it properly. However, in this program we need to display 32 rows of values and need every line of the screen. This is why some users will find it better not to define character 224 using this program.

Although each character is given a particular number (ASCII code) by this program you are free to choose a different number (within the range) if you wish. When you put the VDU 23 statement in your program, put the chosen code in place of the one from the table. This provides a way of using character 224, if your TV screen does not display the top line of the table. Define a character with any other number and assign it to number 224 afterwards.

Moving characters

If you would rather try out your newly defined characters without having to write your own program, you can make use of the fact that their definitions afe stored safely away in the Electron's memory. They stay there until you switch the computer off, except for any which you might decide to redefine. Here is a way to have fun with the characters with the minimum of effort.

Press ESCAPE to leave the program. Then type '*FX4,0' and press RETURN. This restores the 'arrow' and COPY keys to their normal functions. Now clear the screen for action. You can make the computer display any of the characters by typing the immediate command:

```
PRINT TAB(X,Y)CHR$(N)
```

In the above expression, X and Y are the number of the row and column in which you want the character to appear. In Mode 4, X must be in the range 0 to 39, and Y in the range 0 to 31. N is the ASCII code of the character, in other words, its number in the range 224-255. For example, to make the man's head (Fig. 9. 1) appear at the centre of the screen, type:

```
PRINT TAB (20,15)CHR$(232)
```

Experiment with various values for X, Y and N. You will soon work out the way of placing the whole man anywhere you want him.

A short program can be used to make a character, or group of

characters appear to move across the screen. The two programs in man, defined as in Fig. 9.1 . The programs are readily adaptable, and you could turn them into procedures for use in your own graphics programs.

```
   10 REM MOVE ACROSS
   20 MODE 4:VDU 23,1,0;0;0;0;
   30 FOR J=0 TO 37
   40 PRINT TAB(J,10)"  "TAB(J+1,10)CHR$
(224)CHR$(225)
   50 FOR L=1 TO 200:NEXT
   60 NEXT
   70 VDU 23,1,1;0;0;0;
   80 END
```

Listing 9.1

```
   10 REM MOVE DOWN
   20 MODE 4:VDU 23,1,0;0;0;0;
   30 FOR J=0 TO 28
   40 FOR K=0 TO 2
   50 PRINT TAB(20,J+K)" "
   60 NEXT
   70 FOR J=0 TO 2
   80 PRINT TAB(20,J+K+1)CHR$(232+3*K)
   90 NEXT
  100 FOR L=1 TO 300:NEXT
  120 VDU 23,1,1;0;0;0;
  130 END
```

Listing 9.2.

MOVE ACROSS (Listing 9.1) moves the locomotive across the screen, from left to right. The VDU 23 statement on line 20 is to disablk: the cursor. If you do not do this, a flickering cursor follows the loco across the screen, ruining the effect. The statement at line 70 turns the cursor back on again when the program is over. The main loop of the program (lines 30 to 60) first prints a pair of blank spaces, then prints the two locomotive characters side by side, one place to the right. Each time round the loop, the printing is shifted one place to the right. The blanks clear away the previous image ofthe locomotive and a new image is printed further to the right. The loop at line 50 is a delay, to give you time to see the locomotive before it is moved to its next position.

MOVE DOWN (Listing 9.2) makes the little man descend from the

top of the screen to the bottom. It works in a similar way to the MOVE ACROSS program, except that we now have to print the spaces or characters one below the other, instead of side-by-side. The loop at lines 40 to 60 prints three spaces one below the other, and is followed by a loop at lines 70 to 90 to print the three characters making up the man.

Keying in

Take care with the punctuation in VDU statements. Sometimes there are commas and sometimes there are semicolons. Copy the listing exactly. Also, some VDU statements have a semicolon after the last number, but others have nothing after the last number. Watch out for the brackets in line 590. The question mark in line 60 and some other lines does not appear often in programs. This is the byte indirection operator, explained in Chapter 23 of the User Guide. The symbol after the ' 2ln line 6!0 is the exponentiation symbol. You key it by pressing SHIFT and the ' ^ ~ Ieft-arrow' key.

Program design

20-30 Initialising Mode and a variable.
40 Redefine Mode 1 colours: black as blue, and yellow as green.
50 Disable editing keys.
60 Clear character memory.
70-120 Print borders of design area.
130-200 Print rows of numbers.
210 Print row of characters.
220-260 Getting starting nuinber.
270-280 Clearing message, then printing numbers 1 to 9 below character area.
290-440 Drawing routine, which ends when ' Sbr ' Rls pressed; begins by up-dating all characters (line 310); lines 320-360 accept key-presses; lines 370-420 carry out appropriate actions; PROCbit is used if the COPY key or the space-bar is pressed.
450 Clear screen and repeat drawing routine.
460-550 Display table of VDU statements.
560-570 Wait for space-bar to repeat drawing routine.
580-660 PROCbit to change the appropriate bit in a byte of

memory each time a white ' b!ock' is drawn or deleted.

670-750 PROCchars to display a row of characters at the bottom of the screen.

The program

```
   10 REM ** CHARACTERS **
   20 MODE 1
   30 B$=STRING$(8," ")
   40 VDU 19,0,4,0,0,0:VDU 19,2,2,0,0,0
   50 *FX 4,1
   60 FOR J=0 TO 255:B=J+3072:?B=0:NEXT
   70 COLOUR 129:PRINT TAB(8,1)B$:COLOUR
 130:PRINT TAB(16,1)B$:COLOUR 129:PRINT
TAB(24,1)B$
   80 FOR J=1 TO 24
   90 COLOUR 129:IF J>8 AND J<17 THEN CO
LOUR 130
  100 PRINT TAB(7,J+1)" ":PRINT TAB(32,J
+1)" "
  110 NEXT
  120 COLOUR 129:PRINT TAB(8,26)B$:COLOU
R 130:PRINT TAB(16,26)B$:COLOUR 129:PRIN
T TAB(24,26)B$
  130 COLOUR 128:COLOUR 2:VDU 31,4,27
  140 FOR J=1 TO 32
  150 PRINT; (J+23) DIV 10;
  160 NEXT
  170 VDU 31,4,28
  180 FOR J=1 TO 32
  190 PRINT; (J+23) MOD 10;
  200 NEXT
  210 PROCchars
  220 COLOUR 2
  230 REPEAT
  240 INPUT TAB(6,30)"Which number? (224
-247)  "N$;
  250 N=VAL(N$)
  260 UNTIL N>223 AND N<248
  270 PRINT TAB(6,30)B$;B$;B$;B$
  280 COLOUR 1:PRINT TAB(N-220,30)"12345
6789":COLOUR 2
  290 X=8:Y=2
```

```
  300 REPEAT
  310 PROCchars
  320 REPEAT
  330 VDU 31,X,Y
  340 KEY$=GET$
  350 CODE=ASC(KEY$)
  360 UNTIL INSTR("D R",KEY$) OR CODE>13
4 AND CODE<140
  370 IF CODE=135 THEN COLOUR 128:PRINT"
 ":COLOUR 131:VDU 8:PROCbit(0)
  380 IF CODE=136 THEN X=X-1:IF X=7 THEN
 X=8
  390 IF CODE=137 THEN X=X+1:IF X=32 THE
N X=31
  400 IF CODE=138 THEN Y=Y+1:IF Y=26 THE
N Y=25
  410 IF CODE=139 THEN Y=Y-1:IF Y=1 THEN
 Y=2
  420 IF KEY$=" " THEN COLOUR 131:PRINT"
 ":VDU 8:PROCbit(1)
  430 COLOUR 3:COLOUR 128:VDU 31,4,29
  440 UNTIL INSTR("DR",KEY$)
  450 IF KEY$="D" THEN CLS:GOTO70
  460 CLS
  470 FOR J=0 TO 31
  480 COLOUR(J MOD 2)+2
  490 PRINT TAB(2)CHR$(J+224);TAB(4)J+22
4;
  500 FOR K=0 TO 7
  510 B=3072+J*8+K
  520 PRINT" ";?B;
  530 NEXT
  540 IF J<>31 AND K<>7 THEN PRINT""
  550 NEXT
  560 REPEAT: UNTIL INKEY(-99)
  570 CLS:GOTO 70
  580 DEF PROCbit(Z)
  590 NB=3072+8*(N-224)+(((X-8) DIV 8)+(
(Y-2) DIV 8)*3)*8+((Y-2) MOD 8)
  600 BY=?NB
  610 BV=2^(7-((X-8) MOD 8))
  620 PV=BY AND BV
  630 IF PV=0 AND Z=1 THEN BY=BY+BV
  640 IF PV>0 AND Z=0 THEN BY=BY-BV
```

```
650 ?NB=BY
660 ENDPROC
670 DEF PROCchars
680 VDU 31,4,29
690 VDU 23,1,0;0;0;0;
700 COLOUR 3
710 FOR J=1 TO 32
720 PRINT;CHR$(223+J);
730 NEXT
740 VDU 23,1,1;0;0;0;
750 ENDPROC
```