the control block is set up is shown here:

The OSWORD Control Block

Control Block Entry	Function
0	Low byte of address to which the characters are to be written
1	High byte of the address to which the characters are to be written
2	Maximum line length
3	Minimum acceptable ASCII code
4	Maximum acceptable ASCII code

During the inputting of characters by this routine, the Delete key has its usual function. The routine can be exited by pressing the Return or the Escape key. For example, the control block that follows has these results when the OSWORD call is made:

Example OSWORD Control Block

Control Block Entry	Value
0	&00
1	&0C
2	&07
3	32
4	96

1. The first character input will be stored at &C00, the second at &C01, and so on.
2. Only seven characters will be accepted; if you try to type in more characters than this then a 'beep' will be generated and the additional characters will be ignored.
3. Only characters with ASCII codes between 32 (the Space character) and 96 (the £ character) will be accepted; others will be ignored.

As you can see, the call enables us to screen out unwanted characters in the input. When the routine is exited, the status of the C flag informs us what caused the termination of the routine. If C=1, then Escape has been pressed. If C=0, then Return terminated the entry of characters and the Y register holds the length of the string entered, including the carriage return ASCII value added to the end of the string by the pressing of the Return key. Remember that you can use this routine on either of the input streams selected by *FX2 or its machine code equivalent.

We've now seen how easy it is to read data into the BBC Micro. Let's proceed to look at means of sending characters to the currently selected output stream. Again, we use an OS call to select the stream. This is *FX3,n — where n specifies the stream to be selected. Each bit of the n parameter controls a different output stream. As an example, *FX3,1 enables the serial, screen and printer output and allows SPOOLed output, provided that a *SPOOL command has been issued.

Output Stream Control Parameter Table

Value \ Bit	0	1	2	3	4	6
1	RS423 ON	Screen OFF	Printer OFF	Printer ON*	Spooled OFF	Printer OFF**
0	OFF	ON	ON	OFF*	ON	ON**

*Printer is on or off independent of whether printer is disabled by other means

**Printer off unless character is preceded by a CHR$ (1)

The main routine that is used for sending characters to the current output stream is named OSWRCH and is called at address &FFEE, vectored through locations &20E and &20F. It is very easy to use; simply load the A register with the ASCII code of the character that you want to write and then call the routine. All three following routines print the character 'A' to the screen:

```
1000 VDU 65
```

```
1000 PRINT CHR$ (65)
```

```
1000 LDA #65
1010 JSR &FFEE
```

The BASIC VDU command has virtually the same effects as using OSWRCH. Characters in the ASCII range 32 to 255 print characters on screen, with the exception of ASCII code 127, which is the Delete character. The characters in the range from 0 to 31, however, have special functions, which we'll now examine. It is these codes that enable us to use OSWRCH to draw graphics to the screen, execute COLOUR and GCOL commands, define characters, and control the 6845 chip — which controls the video display of the BBC Micro.

Writing characters to the screen or elsewhere via the OSWRCH routine is often referred to as writing to the *VDU drivers*. The ASCII Control Codes Table shows the effects of the character codes between 0 and 31 when they are sent to the VDU drivers. As you can see, they enable us to do anything via the OS in our machine code graphics routines that we can do in BASIC.

GRAPHICS VIA OSWRCH

All the usual graphics commands are available to us via the OSWRCH routines. Our second example program, given in the margin, will draw a red line on the screen. Lines 50 to 75 of the program execute a GCOL 0,1 command, setting the colour of the line to red. Lines 90 to 150 then execute a PLOT 5,100,100 command, which is the same as a DRAW 100,100 command. Line 100 sends the PLOT type (5 in this case) to the VDU drivers, followed by a two-byte x co-ordinate, low byte first, and then a two-byte y co-ordinate, low byte first. A MOVE command can be executed by replacing the 5 with a 4 (MOVE is simply a PLOT 4,x,y command). Other graphics operations — such as the PLOT 85 command for drawing triangles — are also accessible by these means. One important point to remember about sending PLOT commands to the