Graphics Via OSWRCH

```
10 MODE 1
20 FOR I%=0TO2STEP2
30 P%=&C00
40 [OPT I%
50 LDA #18
55 JSR &FFEE
60 LDA #0
65 JSR &FFEE
70 LDA #1
75 JSR &FFEE
80
90 LDA #25
95 JSR &FFEE
100 LDA #5
105 JSR &FFEE
110 LDA #0
115 JSR &FFEE
120 LDA #100
125 JSR &FFEE
130 LDA #0
135 JSR &FFEE
140 LDA #100
145 JSR &FFEE
150 RTS
160 ]:NEXT I%
170 CALL &C00
```

VDU drivers is that they expect five bytes after the value 25 is sent as the first byte; if these bytes are not received, then strange things can happen. This applies to all VDU driver operations that require more than one byte to be sent.

VDU23 is another very versatile VDU driver command. It is used to define user-generated characters. For example, VDU23,224,255,255,255,255,255,255,255,255 will define the character 224 (usually undefined) as a solid block. Characters 224 to 255 in Modes 0 to 6 can be redefined by the user with this command. Indeed, using the VDU23 command in conjunction with one of the OSBYTE calls enables the user to redefine other characters in the character set. Any VDU23 calls that are not recognised by the OS — such as VDU23,0 . . . . — are passed through a special vector at &226 and &227. By changing the address contained in this vector, you can add your own VDU23 command routines.

A more advanced use for the VDU23 command is to enable the programmer to access the 6845 video controller chip. VDU23 commands take the form:

```
VDU23,0,register,value,0,0,0,0,0,0
```

where register is the 6845 register to which you want to write, and value is the value to be written to the 6845. As an example of the use of VDU23 in this way, the following program alters two of the 6845 registers: they tell the chip which area of the computer's memory is to be used as video memory. The program alters the start of video RAM to address &0000. This shows the BBC OS workspace on the screen, and various interesting effects can be seen. Try adding a few lines of code, dimensioning some arrays, etc. The routine is written in BASIC but will easily convert to assembler:

```
10 MODE 0
20 VDU23,0,12,0,0,0,0,0,0,0
30 VDU23,0,13,0,0,0,0,0,0,0
40 VDU28,0,10,30,0:REM set up a text window
50 CLS
```

There are two other OS calls that are related to OSWRCH. These are: OSNEWL and OSASCII. OSNEWL, when called at &FFE7, writes a line feed and a carriage return to the screen. OSASCII, called at &FFE3, is a variant on OSWRCH, and is useful for text handling. When a character 13 is written via this call, a line feed, or character 10, is also written to the output screen. This should *not* be used, therefore, if you are writing graphics commands to the VDU drivers, since an extra CHR$(10) might be generated, thus causing confusion.

Finally, *SPOOL and *EXEC are two commands that enable output and input to the currently selected filing system. *EXEC filename will cause a file with the appropriate name to be opened, if present, and its contents read in, as if from the keyboard. *SPOOL writes characters to the file named in the command, as if the characters were being written to the output stream.

That concludes our introductory discussion of the BBC Micro's operating system. In the next few instalments we will pause to consider the use of machine code routines to improve screen output on the Commodore 64, before returning to our extensive investigation of a range of operating systems.

ASCII Control Codes Table

Code	Extra Bytes needed	Description
0	0	Does nothing
1	1	Sends the next character to the printer only
2	0	Turns the printer on
3	0	Turns the printer off
4	0	Writes text to the text cursor
5	0	Writes text to the graphics cursor
6	0	Allows VDU drivers to write characters to output stream
7	0	Generates a short tone
8	0	Moves cursor one space left
9	0	Moves cursor one space right
10	0	Moves cursor one space down
11	0	Moves cursor one space up
12	0	Clears the text area of screen
13	0	Returns cursor to the beginning of current line
14	0	Paged mode turned on
15	0	Paged mode turned off
16	0	Clears the graphics area of screen
17	1	Sets text colour to colour whose code is the following byte
18	2	Does a GCOL with the next two bytes to be sent to the drivers. For example, sending 18,0,3 would perform a GCOL 0,3 command
19	5	Defines the logical colours. See BBC user guide
20	0	Returns logical colours to default values
21	0	Does not allow characters to be written to output stream by the VDU drivers
22	1	Sets screen mode to mode of following byte. Sending 22 and 7 will enable Mode 7. However, HIMEM is not altered
23	9	Sends commands to the 6845 chip; programs user-generated characters
24	8	Defines a graphics window
25	5	Performs PLOT command
26	0	Sets default text and graphics window
27	0	No effect
28	4	Sets up a text window
29	4	Sets the graphics origin
30	0	Returns the text cursor to top left of screen
31	2	Puts text cursor to the x,y position in the two bytes following. For example, sending 31,10,10 will set the text cursor to position 10,10 on the screen