User-Defined Commands (USERV)

```
10 USERV=&200
20 ?USERV=&00
25 ?(USERV+1)=&0C
30 FOR I%=0TO2STEP2
40 P%=&C00
50 [ OPT I%
60     CMP #0
70     BNE notcode
80     TAY
90   .loop
95     JSR &FFE3
100    DEY
110    CPY #0
120    BNE loop
130    RTS
140  .notcode
145    RTS
150 ]:NEXT I%
160
200 FOR rep=1 TO 10
210 FOR asc=33 TO 48
220 *CODE asc,rep
250 NEXT:NEXT
```

address, &00, in location &200, and the high byte, &0C, in location &201. The routine prints to the screen a given number of characters; the first parameter of the *CODE command holds the ASCII code of the character and the second parameter is the number of times that the character is to be printed to the screen.

*LINE isn't as generally useful as *CODE, but if you want to use it then the principles shown in the following program can be applied, as long as you remember that you will enter your program with one in the A register and the X and Y registers pointing to the text string in memory. This is the main function of *LINE: passing text strings over to machine code programs. For situations where there aren't many parameters to pass to your routine, these two calls are the most elegant way of doing it.

Line 20 of the program sets up the USERV to point to our machine code routine. The loop between line 90 and 120 prints the character whose ASCII code is in the A register to the screen Y times. If the routine is entered by a *LINE command, lines 60 and 70 detect this and quit the routine. Lines 200 to 250 actually issue the *CODE command with variable parameters.

USER INTERACTION

The main methods of interaction with a microcomputer are via the keyboard and the VDU, or television screen. Our detailed investigation of the BBC Micro's operating system continues with a discussion of the ways in which the machine's OS enables us to interact with these two vital areas of the computer.

Let's begin by examining the OS call that enables us to read characters from the currently selected input stream. This routine, named OSRDCH is called at address &FFE0 and is vectored through locations &210 and &211. As it accepts single characters from the currently selected input stream, we should first look at how we select the input stream. There are two major input streams — the keyboard and the RS423 input. We can select one of these by means of an OSBYTE, or *FX, call. The following table shows this command in both machine code and BASIC.

Selecting The Input Stream

n	Keyboard	RS423	Assembler	BASIC
0	✓	✗	LDA #2	*FX2,n
1	✗	✓	LDX #n	
2	✓	✓	JSR &FFF4	

Thus, *FX2,1 disables the keyboard and enables the RS435 as the current input stream. Data received on the RS423 input would be treated as if it were being typed in to the computer. In assembler, to do the same job, n would have a value of one.

Once you've set up the input stream that you wish to use, you can access it with OSRDCH. The first thing to say about OSRDCH is that it's really only useful in assembler programs — BASIC is obviously well endowed with input routines such as GET and INPUT. We call this routine at address &FFE0, and after return from the call, the character read in from the input stream is in the A register; if an error has been detected during the read operation, then the carry flag is set to one, otherwise it is reset to zero. Thus, if C=1 on return from the OSRDCH routine, the character code contained in the A register is probably invalid in some way. When we're reading from the keyboard, this error is often caused by the Escape key being pressed. This situation is indicated by C=1 and the A register holding the value 27 (the ASCII value for Escape). If you detect this situation, then it is *vital* to act upon it; the BBC OS expects such an Escape error to be acknowledged by the program.

We do this by using an OSBYTE call with A=126. This cleans up various parts of the BBC OS workspace in response to the Escape error. The acknowledgement operation is, of course, usually done automatically by the BASIC interpreter during an input operation when Escape is pressed. The simple routine that follows reads the current input stream and acts accordingly if an Escape error is detected.

```
1000  .input  JSR  &FFE0
1010          BCS error
1020          RTS
1030  .error  CMP#27
1040          BNE out
1050          LDA #126
1060          JSR &FFF4
1070  .out    RTS
```

Line 1000 calls the OSRDCH routine, and 1010 checks the carry flag. If it is clear, then an RTS to the calling program is executed, with a legal character in the A register. Otherwise, line 1030 checks to see if the error was caused by an Escape event, and, if it was, lines 1050 and 1060 execute the OSBYTE call that acknowledges the Escape event. You might think that in order to enter strings of data into your machine code programs you have to use a routine of your own devising, but you don't. There exists in the OS a means of reading strings of characters from the currently selected input stream. This routine is accessed via one of the OSWORD calls, which will be covered in more detail later in the course. However, we'll use this particular OSWORD call now to read in strings of characters.

The OSWORD routines are called at address &FFF1. There are several of these, and we specify which we require by the value held in the A register when the call is made. In all OSWORD calls, the X and Y registers of the 6502 point to a block of memory called a *control block*, which holds the parameters that are to be passed over to the routine. The X register holds the low byte of the control block address and the Y register holds the high byte of the address — this follows the 6502 Lo-Hi addressing convention. The way in which