is to be treated as a * command, and is to be interpreted and executed by the OSCLI routine. X holds the low byte of the address and Y holds the high byte of the address.

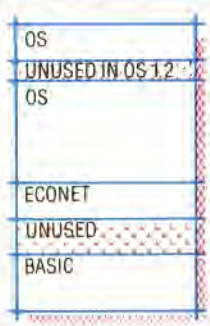
BASIC Version	BASIC+Assembly Language Version
10 DIM C 100 20 OSCLI=&FFF7 30 INPUT"ENTER COMMAND ",A$ 40 $C=A$ 50 X%=C MOD 256 60 Y%=C DIV 256 70 CALL OSCLI 80 GOTO 30	10 DIM C 100 20 OSCLI=&FFF7 30 FOR I%=0 TO 2 STEP 2 40 P%=&C00 50 [OPT I% 60 .code LDX #C MOD 256 70 LDY #C DIV 256 80 JSR OSCLI 90 RTS 100];NEXT I% 110 INPUT"ENTER COMMAND ",A$ 120 $C=A$ 130 CALL code 140 GOTO 110

Running this program produces a prompt: the user types in a * command, presses Return and the command will be executed. The rather odd-looking DIM statement in line 10 of both these programs forces the computer to set aside 100 bytes of memory in the space set aside for BASIC variables and initialises the variable C with the address of the start of this block of memory. This 100 bytes can then be used for storing machine code programs, or, as in this case, data for machine code programs. The $C=A$ statement puts the bytes that make up the command string held in A$ into the block of 100 bytes, starting at the first byte reserved by the DIM command. In both programs, the X and Y registers (or the X% and Y% variables) are set up and the call is made to OSCLI. The command string is then executed.

This program is the basis of a routine for use in menu-driven programs, where it might be useful to allow the user to do things like catalogue disks or tapes without leaving the program. The command required is simply put in to the string variables and passed to OSCLI for execution. Typing in *A$ will not work. The OS will attempt to execute a command called *A$, which simply doesn't exist!

Using this technique, it is also possible to pass numeric variables to a * command by using the STR$ function to convert them into strings. Normally, the CLI will not accept any variable names passed to it; it gives the Bad Command error message instead of trying to evaluate the variable concerned.

Any * command that is not recognised by the OS is passed over to any paged ROMs. Each one is asked if it recognises the command; if it does then it executes it. Commands not passed for execution in this fashion are then treated as bad commands *only if* a fast filing system, such as a disk drive, is not in use. If it is, then the disk will be inspected to see if it contains a file with the same name as the command (without the *). If it does, then the file is loaded into the machine and treated as a machine code program. This can cause *big* problems if the file isn't a machine code program! The computer usually 'hangs' until you put it out of its misery. If such a file isn't found then the Bad Command error message is printed.



BBC Micro Zero Page Usage
The 6502's limited indexing facilities are complemented by its view of the Zero Page (&0000 to &00FF) as an array of auxiliary index registers. This naturally leads to heavy use of these locations by the OS, but a few bytes have been left unused for programmers' purposes

The * Commands

Command	Description and Comments
*HELP	This will give the version number of the BASIC. It can also be used to gain information about other paged ROMs fitted — e.g. *HELP DFS
*BASIC	This command will enter the BASIC language. A variation of this command, such as *WORD, will enter a paged ROM, in this case View
*CODE *LINE	In OS 1.2 only. This enables the user to add new commands to the OS
*KEY	Used to program the red function keys
*MOTOR n	Used to control the tape motor relay: n=0 turns the relay off and n=1 turns the relay on
*ROM, *TAPE, *DISK, *NET	Used to initialise the appropriate filing system — i.e. *TAPE will enter the 1,200 baud tape filing system, and *DISK will enter the disk filing system
*FX	This enables the programmer to control the values of various OS variables, thus controlling the behaviour of the OS
*RUN, *OPT *LOAD, *. *CAT, *SAVE	These are all filing system commands and will be examined later in the series
*SPOOL *EXEC	*SPOOL sends the screen output to the screen and to a file. *EXEC reads in data from a file as if it were being read from the keyboard
*TV x,y	Affects the vertical position of the screen and the screen interface: x=0 means no change in vertical position, x=1 moves screen down one line, x=255 moves screen up one line; y=0 interlace on, y=1 interlace off. The effects of this command come into operation at the next mode change and stay in effect until a CTRL-BREAK or another *TV command is issued

For further explanation of the use of these commands, consult the BBC Micro's user guide