If the prompt is not visible on the bottom line of the screen, type X, otherwise,

Type B 600B

This will set a new Breakpoint after the BC and DE register pair have been incremented, but before the HL pair is incremented.

C – Breakpoint Continue

This command allows you to continue from a Breakpoint, and is executed by typing C followed by ENTER. You can Escape to the monitor by typing X before ENTER. The program being run will continue as if the Breakpoint had never occurred.

The screen is cleared; the program Stack is reset; and the CPU registers are re-loaded from their data block before the Breakpoint address is put into the Program Counter, and execution is resumed.

Type C
Type ENTER

The routine will run on until it reaches the next Breakpoint, and display "Press BREAK for Monitor".

When the prompt appears, after pressing 'BREAK',

Type K	to restore the bytes occupied by the Breakpoint.
Type R	to display the registers.

You can now verify that the Program Counter contains 600B, the BC and DE register pair contain 0001, having been incremented, and the HL pair still contains 0000.

When a routine encounters a Breakpoint, it returns control to the MONITOR with a CALL operation, the return address being stored on the Program Stack, for use by the Breakpoint Continue (C) command. Having encountered a Breakpoint and studied the CPU registers and/or memory locations, one of two situations will occur:

1) Everything will be as you expect, and the program is correct to that point. In this case, you would normally restore the Breakpoint bytes ('K' command) and use the Breakpoint Continue ('C' command) to continue the program to a new Breakpoint.

or

2) An error will become evident, in which case you would track down the error and correct it, and then, leaving the current Breakpoint set, use the 'J' command to re-run the program up to the same Breakpoint, to check that your correction is successful.

The Program Stack operation of the MONITOR allows you to do this providing that, at the Breakpoint, there have been an equal number of PUSHes and POPs, or CALLs and RETs. If the Program Stack is not balanced at the Breakpoint, you will have a cumulative stack imbalance every time you use the 'J' command after a Breakpoint (but not if you use the 'C' command). In this case to restore the Stack to normal once you have traced an error, RETURN to Basic ('Y' command) and re-access the monitor from the beginning, then use the 'J' command to run your program up to the Breakpoint again.

Having set a Breakpoint in a routine, you can either use the 'J' command to run the routine, or you can use the RETURN command to go back to Basic, and run the machine code via the USR function. For example, if you have written some machine code as part of a Basic program, which is accessed by the USR function in the Basic program, you can set a Breakpoint using the MONITOR then return to Basic and run the Basic program. When the Breakpoint is reached in the machine code, the MONITOR will be accessed as shown above, and the Breakpoint Continue command will allow the machine code to resume, and eventually return to the Basic program that called it.

The MONITOR has been carefully designed to allow this free interchange between Basic and machine code, without upsetting the Stack.

P – Printer

This command allows you to produce a Hex dump of any section of memory onto the Sinclair Printer. It takes the form: **'P aaaa bbbb'** where P is the Printer command, aaaa is the Hex address of the first byte to be printed, and bbbb is the Hex address of the last byte you want printed. The Printer produces a display in the format shown below.

```
0000 F3 AF 11 FF FF C3 CB 11
0008 2A 5D 5C 22 5F 5C 18 43
0010 C3 F2 15 FF FF FF FF FF
```