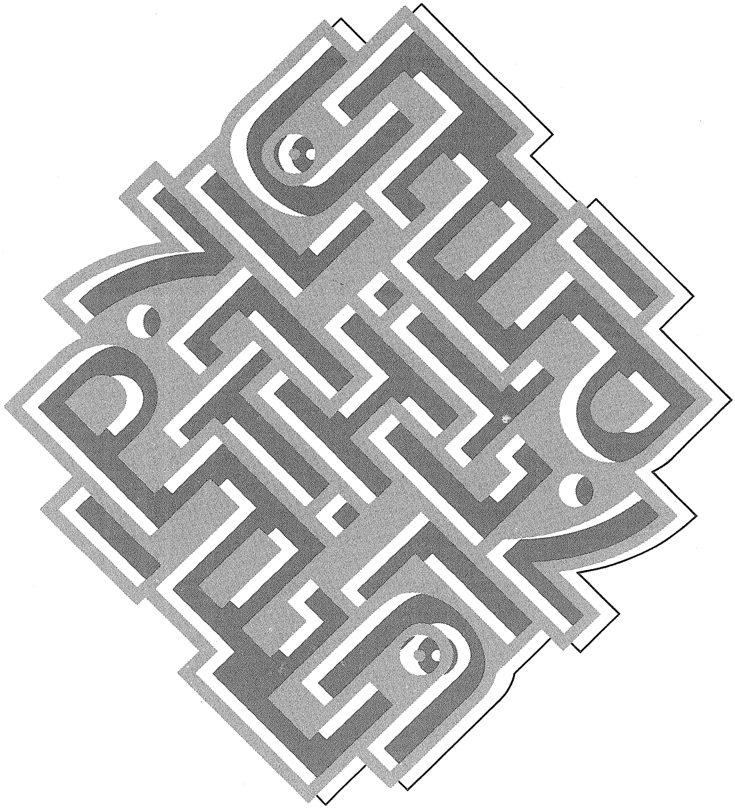
MONITOR



I.S.M.E.C

Graham Bartram

with Ben Keeping and Philip Rogers

BRITISH BROADCASTING CORPORATION

Within this manual and associated software, the letters BBC refer to the British Broadcasting Corporation. The term 'BBC Micro' refers to the computer manufactured by Acorn Computer PLC under licence from the BBC. 'Tube' and 'Econet' are registered trademarks of Acorn Computer PLC. The details from the R65CO2 data sheet are reproduced by kind permission of Rockwell International Ltd.

Disclaimer: Because neither the BBC nor the authors have any control over the circumstances of use of the programs, no warranty is given or should be implied as to the suitability of the programs described in this Manual for any particular application. No liability can be accepted for any consequential loss or damage, however caused, arising as a result of the operation of the program.

First published 1985
Published by the British Broadcasting Corporation
35 Marylebone High Street, London W1M 4AA
Printed in England by Popper & Co., Welwyn Garden City, Herts.

CONTENTS

INTRODUCTION

This User Guide contains all the information you need to use the BBC Monitor. Much of the information in the Guide, such as which key does what, is also available, when using Monitor, by pressing the **RETURN** key or in the form of help pages which come up when setting registers, finding bytes etc.

Chapter 1 explains how to insert the Monitor ROM and then how to start the Monitor. Please read all of this section before you start to fit Monitor.

Chapters 2 to 5 describe the Monitor's facilities in more detail.

Chapters 6 to 8 describe three of the programs that come on the Monitor tape.

1 GETTING STARTED

The components

You should have:

one ROM and utility cassette tape,
a function key strip,
this manual;
your BBC Microcomputer.

Monitor will operate with a BBC Microcomputer/Model A, B, or B+ which uses the version 1.2 Operating System onward. If you are not sure which operating system your computer uses, switch it on, then type:

*FX0 and press **RETURN**

You will see the letters OS plus a number. If this number is 1.20 or higher, you can go straight on to the next section. If you see:

OS0.10 or OS1.00

your Operating System is unsuitable and to use Monitor you must fit a version 1.2 Operating System ROM (or later version) to your machine.

Unless you are very experienced in such matters, you should have Monitor fitted to your machine by an authorised dealer. Remember that fitting it yourself may void the guarantee on your machine.

Installing Monitor

1 Switch off your computer and unplug it from the mains.

2 Remove the top cover. To do this:
 a remove the two fixing screws from the underside. they are the large cross-head screws marked 'FIX'.
 b Loosen the two fixing screws at the back of the machine, but do not completely remove them.
 c Lift the top cover upwards.

Models A and B

3 Loosen the keyboard. To do this:
 a Undo and remove the two fixing nuts and bolts at the sides of the keyboard. (Don't forget to remove the washers.)
 b Move the keyboard *carefully* to reveal the ROM sockets at the front right of the printed circuit board. (The keyboard is connected to the circuit board by a red and black twisted wire and a flat grey cable. If you wish you can unplug these from the

circuit board so as to remove the keyboard completely. Before removing them notice carefully how they plug into the circuit board so that you can replace them correctly.)

4 Plug the Monitor ROM into one of the sockets. The ROM should be inserted with the dimple at the top facing the rear of the machine. Notice that the Operating System scans the sockets from right to left. If you wish Monitor to be entered automa.:cally when you switch on the power or press **BREAK,** plug it into the socket nearest to the right-hand edge of the board. Otherwise, plug it into a socket to the left of the BASIC ROM.

Model B+

3 The ROM sockets on the Model B+ are located in the top left hand area of the circuit board (near the large copper-coloured power supply, toward the back of the computer). There are six sockets arranged in two rows of three. Nearer to the keyboard are two very similar sockets for the speech system. **Under no circumstances should Monitor or any other ROM software be placed in these speech sockets.** Some of the six ROM sockets will already contain ROMs.

4 Choose a free socket from one of these two rows, and plug Monitor in, with the dimple on the chip facing the rear of the machine i.e. like every other north-south facing chip on the board.

CAUTION: Handle the ROM with care and do not touch the pins.

Make sure that the ROM is correctly aligned with the socket.

Each pin should fit into a hole, and there should be none hanging over the edge of the socket. Make sure that all the pins are correctly seated before pushing the ROM home.

DO NOT USE EXCESSIVE FORCE. If a pin is not correctly seated it is easy to bend it up under the ROM if you push too hard.

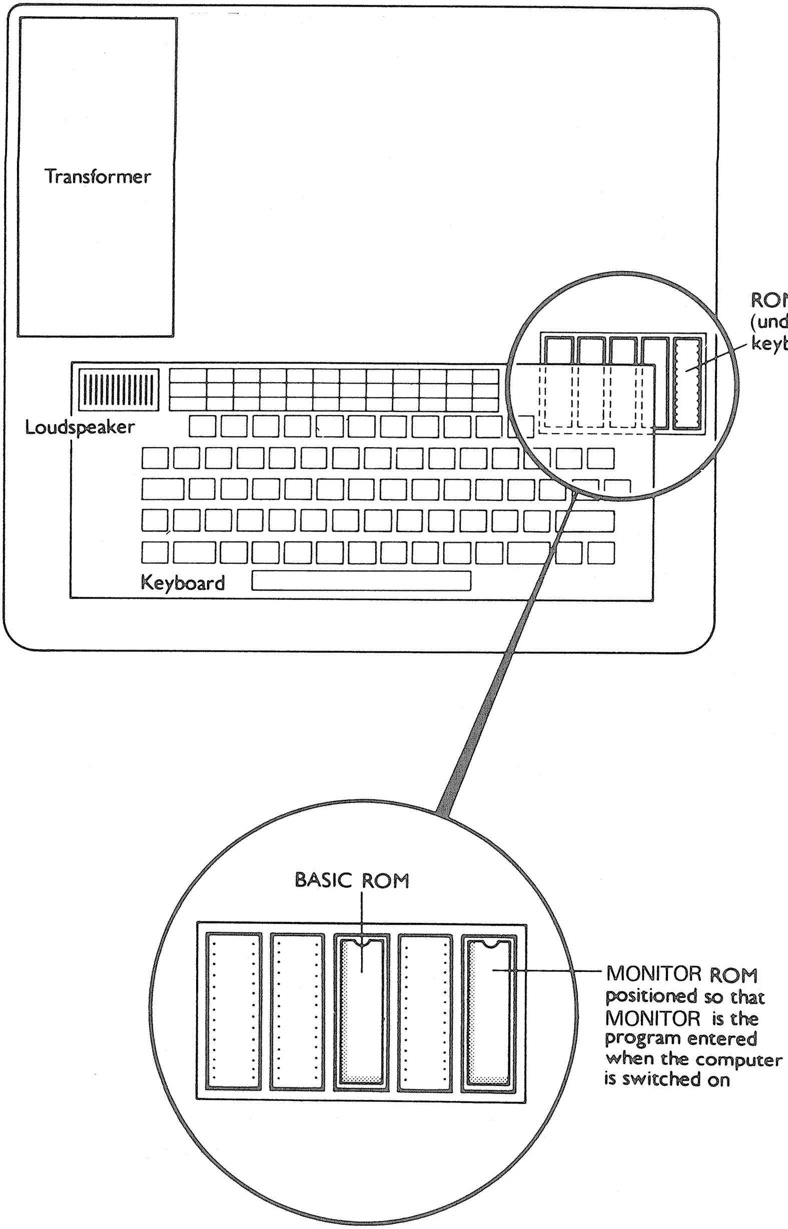
5 (Models A and B only) Replace the keyboard.

a If you have unplugged the wire and cable, carefully plug them back into the circuit board.

b Replace the nuts and bolts securing the keyboard. Do not forget the washers, which go between the nuts and the keyboard base.

6 Replace the top cover.

a Place the cover in position with the back lugs fitted into the loose fixings at the rear of the machine.


Transformer
ROM sockets (underneath keyboard)
Loudspeaker
Keyboard
BASIC ROM
MONITOR ROM positioned so that MONITOR is the program entered when the computer is switched on

This layout applies only to the BBC Model A & B. For details of the Model B+ socket layout, refer to the text.

b Replace the two fixing screws in the base.
c Tighten the two fixing screws at the rear of the machine.

7 Replace the mains power plug and switch on the computer.

Entering Monitor

If you plugged your Monitor ROM into the socket at the right of the circuit board, you will have automatically entered Monitor when you switched on the power.

Otherwise to enter Monitor:

Type * Monitor and then press **RETURN**

Leaving Monitor

If Monitor is NOT in the rightmost socket then pressing **CTRL-BREAK** will take you to your normal power-up language.

Otherwise use the exit **(f9)** option to enter BASIC. Note that if BASIC is not present in your machine, this will not work.

The Monitor utility tape

Enclosed you will find a tape of programs to assist you when using Monitor. These are:

ToDisc: A program to transfer the whole tape to disc.

Menu: A menu to select these programs from disc.

Labels: Converts labels to ASCII and vice versa.
See Section 3. To use CHAIN ''Labels''

RS423: Allows trace output to run on a second BBC Microcomputer.
See Section 4. To use CHAIN ''RS423''

Reloc: Machine code relocator. See Section 7. To use type
* RUN Reloc

MonEdit: Text Editor. See Section 6. To use type * RUN MonEdit

Assem: Disc Assembler. See Section 3. To use type
* RUN Assem

ExSrce: Example source code for the disc assembler. See Section 3. Load using MonEdit or Assem.

!Boot: A !Boot file for disc users.

DFSEd: The DFS disc sector editor. See Section 8. To use type
* RUN DFSEd

ADFSEd: As above for the Advanced Disc Filing System. Type
* RUN ADFSEd.

HIMON: A 'high' version of Monitor for the 6502 Second Processor. To use type * RUN HIMON with the Second Processor active.

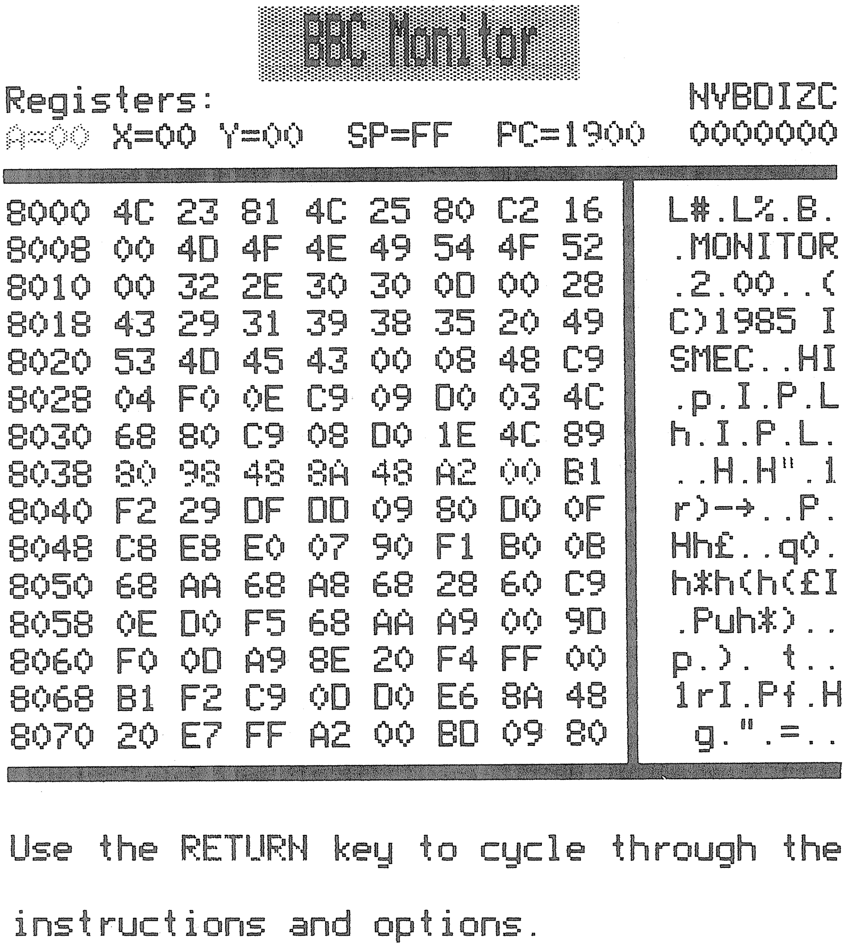
Mouse: A file to allow the AMX Mouse to be used with Monitor. To use type * EXEC Mouse

No Mouse: A file to disable to AMX Mouse, after using the above. To use type * EXEC NoMouse

It is expected that most users will have discs, and if so, the program 'To Disc' will transfer the Monitor tape and prepare a utility disc for you. We did not actually supply the software on disc because this would prevent tape users from using the package properly.

2 LOOKING AT AND ALTERING MEMORY

Memory Display Mode (TAB)
In Memory Display mode, Monitor displays the contents of memory as hexadecimal values in the main display area, and gives the ASCII equivalents in the right-hand section where it is possible to do so. The cursor may be moved around the display with the cursor control keys (see MOVE CURSOR). The cursor may be positioned on either the hex display area or the ASCII display area (see INPUT IN HEX OR ASCII).



```
                    BBC Monitor

Registers:                                  NVBDIZC
A=00  X=00  Y=00   SP=FF   PC=1900         0000000

8000 4C 23 81 4C 25 80 C2 16   L#.L%.B.
8008 00 4D 4F 4E 49 54 4F 52   .MONITOR
8010 00 32 2E 30 30 0D 00 28   .2.00..(
8018 43 29 31 39 38 35 20 49   C)1985 I
8020 53 4D 45 43 00 08 48 C9   SMEC..HI
8028 04 F0 0E C9 09 D0 03 4C   .p.I.P.L
8030 68 80 C9 08 D0 1E 4C 89   h.I.P.L.
8038 80 98 48 8A 48 A2 00 B1   ..H.H".1
8040 F2 29 DF DD 09 80 D0 0F   r)→..P.
8048 C8 E8 E0 07 90 F1 B0 0B   Hh£..q0.
8050 68 AA 68 A8 68 28 60 C9   h*h(h(£I
8058 0E D0 F5 68 AA A9 00 9D   .Puh*)..
8060 F0 0D A9 8E 20 F4 FF 00   p.). t..
8068 B1 F2 C9 0D D0 E6 8A 48   1rI.Pf.H
8070 20 E7 FF A2 00 BD 09 80    g.".=..

Use the RETURN key to cycle through the

instructions and options.
```

Alter memory

In Memory Display mode, values may be entered into the memory directly from the keyboard. The cursor will be shown at either the hex values or at the ASCII values according to which mode of input is currently selected.

Note: When the first digit of a hex value has been entered, Monitor will then wait for either **DELETE** or the second digit to be entered, before it will perform any other action.

Input in hex or ASCII (ESCAPE)

In Memory Display mode, input from the keyboard may be in hex or ASCII. The cursor will indicate where input is going. When the cursor is in the main display which contains hex values, input will be in hex. When the cursor is in the right-hand section which contains ASCII equivalents, input will be in ASCII. **ESCAPE** toggles between the two.

Move cursor (Cursor keys)

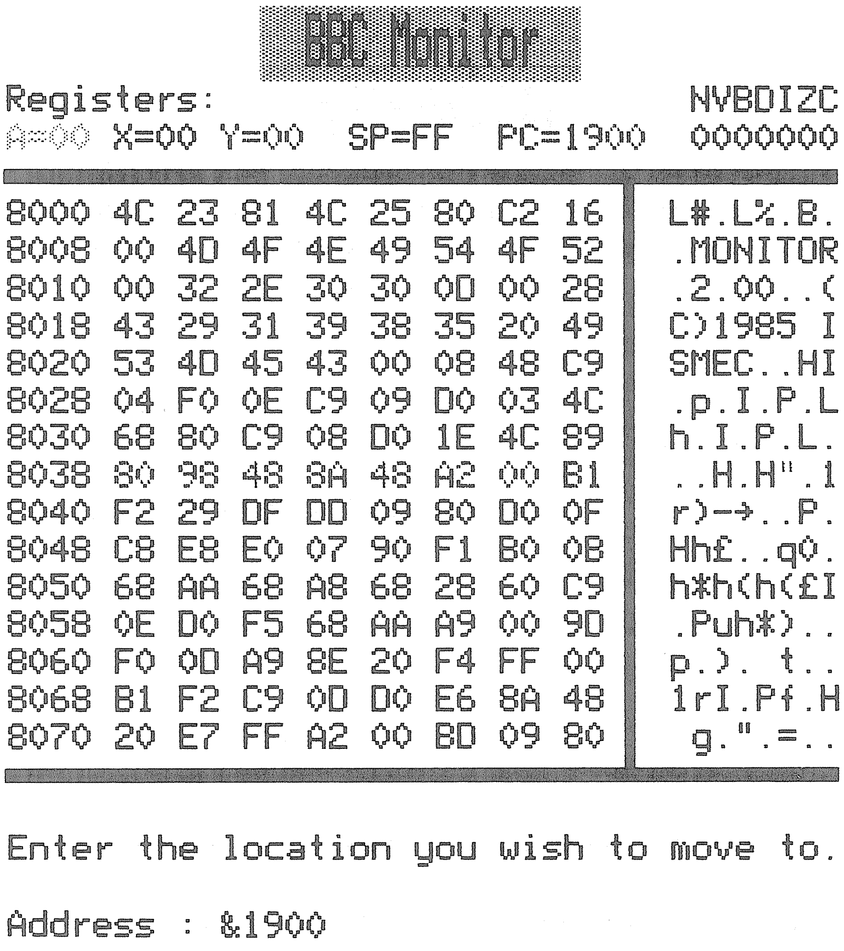
The cursor may be moved around the display area when in memory display mode, using the cursor control keys. The cursor may be moved a screen at a time, either up or down, by using **SHIFT CURSOR UP** and **DOWN.** The cursor may be positioned in either the Hex display area or the ASCII display area (see INPUT IN HEX OR ASCII).

Binary (COPY)

When the **COPY** key is pressed in Memory Display mode the value of the byte at the cursor position is displayed in binary. It will be displayed for as long as the **COPY** key is pressed.

Move to a location (f0)

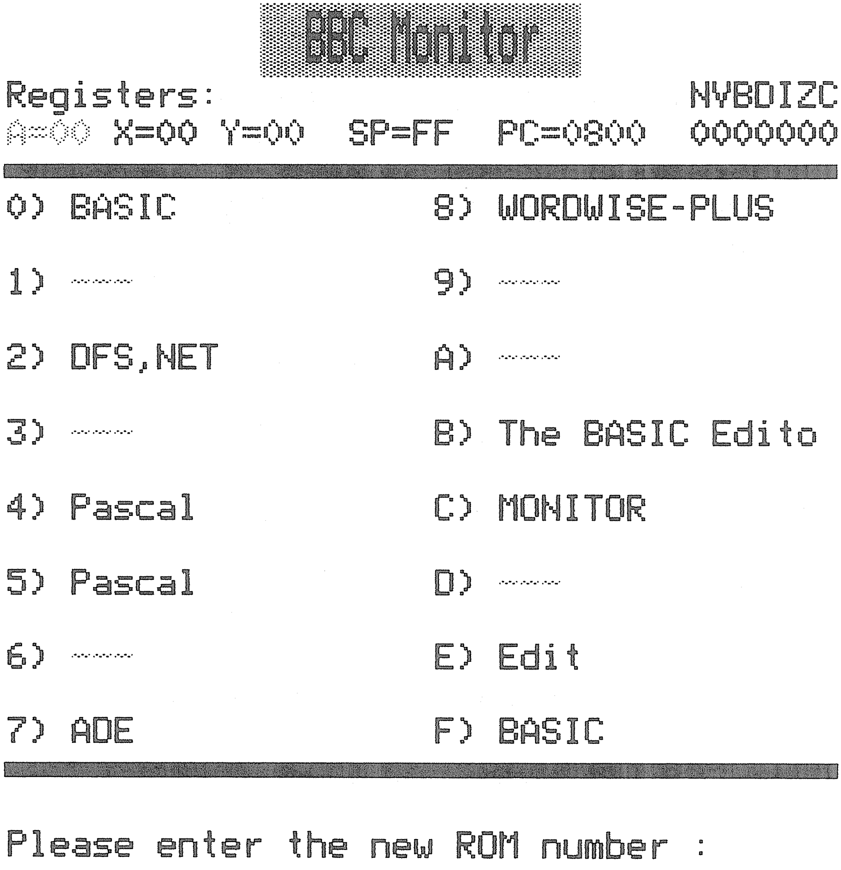
This allows you to move to a particular location in one command rather than moving there using the cursor keys. Monitor will ask for an address and then move the cursor to that location. The address may be entered as a hex number, decimal number, label or arithmetic expression including any of these. This feature applies to all numeric input and is explained in more detail in Section 5, (CALCULATE).



```
                   BBC Monitor
Registers:                                  NVBDIZC
A=00  X=00  Y=00   SP=FF   PC=1900          0000000

8000  4C  23  81  4C  25  80  C2  16     L#.L%.B.
8008  00  4D  4F  4E  49  54  4F  52     .MONITOR
8010  00  32  2E  30  30  0D  00  28     .2.00..(
8018  43  29  31  39  38  35  20  49     C)1985 I
8020  53  4D  45  43  00  08  48  C9     SMEC..HI
8028  04  F0  0E  C9  09  D0  03  4C     .p.I.P.L
8030  68  80  C9  08  D0  1E  4C  89     h.I.P.L.
8038  80  98  48  8A  48  A2  00  B1     ..H.H".1
8040  F2  29  DF  DD  09  80  D0  0F     r)—→..P.
8048  C8  E8  E0  07  90  F1  B0  0B     Hh£..q0.
8050  68  AA  68  A8  68  28  60  C9     h*h(h(£I
8058  0E  D0  F5  68  AA  A9  00  9D     .Puh*)..
8060  F0  0D  A9  8E  20  F4  FF  00     p.). t..
8068  B1  F2  C9  0D  D0  E6  8A  48     1rI.Pf.H
8070  20  E7  FF  A2  00  BD  09  80      g.".=..

Enter the location you wish to move to.

Address : &1900
```

Paged ROM selection (SHIFT f0)
Monitor assumes that each of the Paged ROMs occupies memory from &8000 to &BFFF. This option allows any of the Paged ROMs present in the computer to be the one that appears in this region of memory. To select a ROM, press the hex digit (0—F) that corresponds to its name on screen.

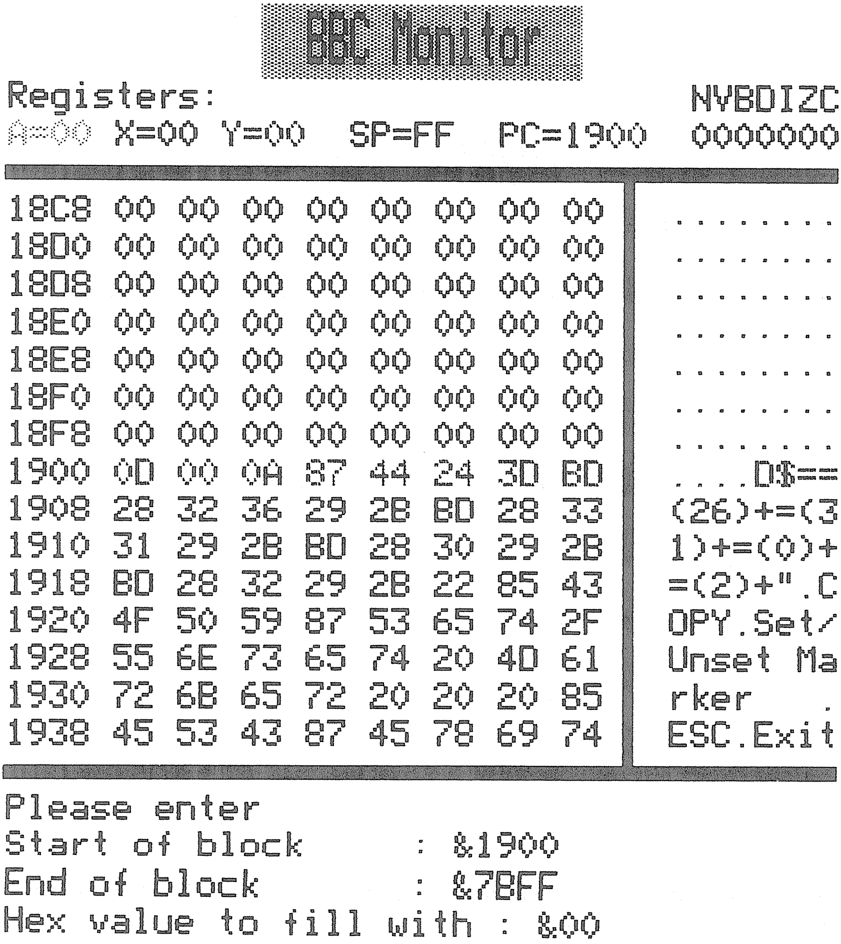
Note: This facility is not available when Monitor is in the Second Processor, unless the I/O Processor memory is selected (see SET OPTIONS).



```
                     BBC Monitor
Registers:                                   NVBDIZC
A=00 X=00 Y=00  SP=FF  PC=0800   0000000

0) BASIC                 8) WORDWISE-PLUS

1) ----                  9) ----

2) DFS,NET               A) ----

3) ----                  B) The BASIC Edito

4) Pascal                C) MONITOR

5) Pascal                D) ----

6) ----                  E) Edit

7) ADE                   F) BASIC

Please enter the new ROM number :
```

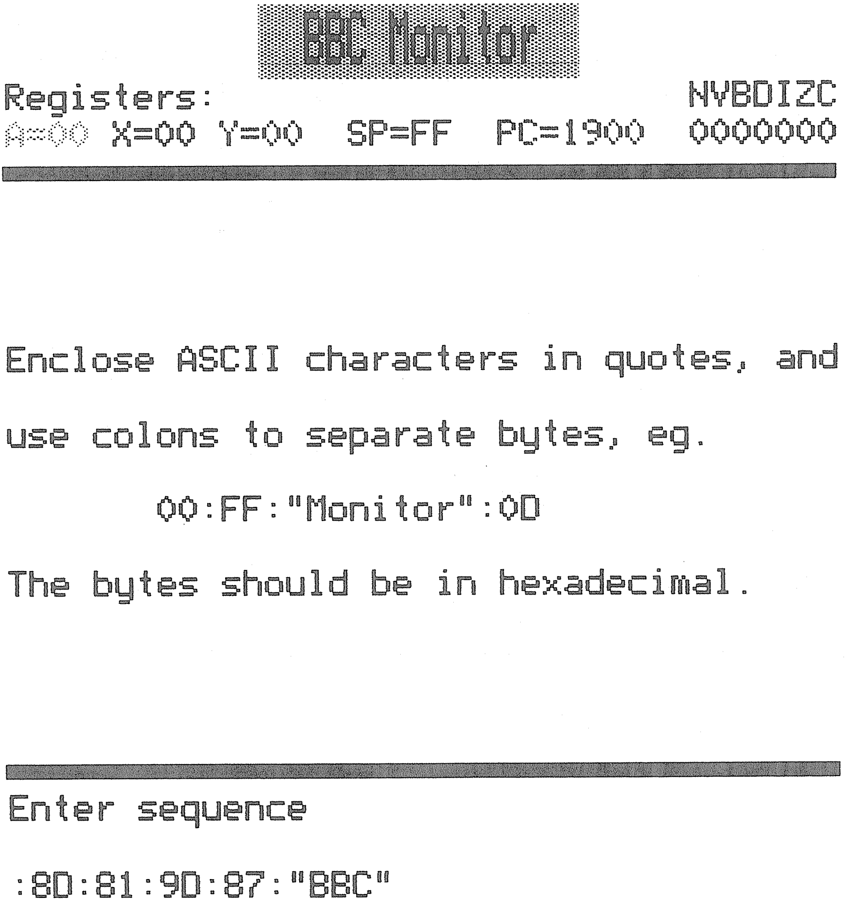
Fill an area (SHIFT f1)

This is a routine to fill an area of memory with a specified byte. Monitor will ask for the start and end locations of the block you wish to fill and then asks for the value needed to fill that block. Note that when using this facility on the Second Processor, any attempt to fill or otherwise alter the memory areas where Monitor or HIMON and the Operating System reside will have unpredictable results.



```
                    BBC Monitor
Registers:                                  NVBDIZC
A=00  X=00  Y=00   SP=FF   PC=1900          0000000

18C8 00 00 00 00 00 00 00 00    ........
18D0 00 00 00 00 00 00 00 00    ........
18D8 00 00 00 00 00 00 00 00    ........
18E0 00 00 00 00 00 00 00 00    ........
18E8 00 00 00 00 00 00 00 00    ........
18F0 00 00 00 00 00 00 00 00    ........
18F8 00 00 00 00 00 00 00 00    ........
1900 0D 00 0A 87 44 24 3D BD    ....D$==
1908 28 32 36 29 2B BD 28 33    (26)+=(3
1910 31 29 2B BD 28 30 29 2B    1)+=(0)+
1918 BD 28 32 29 2B 22 85 43    =(2)+".C
1920 4F 50 59 87 53 65 74 2F    OPY.Set/
1928 55 6E 73 65 74 20 4D 61    Unset Ma
1930 72 6B 65 72 20 20 20 85    rker   .
1938 45 53 43 87 45 78 69 74    ESC.Exit

Please enter
Start of block       : &1900
End of block         : &7BFF
Hex value to fill with : &00
```

Find a byte sequence (f3)
A search may be made in memory for any sequence of bytes. The sequence may be entered from a keyboard as either hex values or as ASCII characters, or as a combination of both. Quote marks are used to enclose a string of ASCII characters. Strings and hex bytes are separated by colons. Quote marks themselves may be searched for by entering their hex value (&22). The ' #' character is used as a single character wildcard, i.e. ' #' will match any single character. The search is carried out starting at the current cursor position, continuing up to &FFFF, followed by &0000 up to the starting location, until the sequence is found, when the cursor position is set to the first byte of



```
                  BBC Monitor
Registers:                             NVBDIZC
A=00 X=00 Y=00  SP=FF  PC=1900  0000000



Enclose ASCII characters in quotes, and

use colons to separate bytes, eg.

        00:FF:"Monitor":0D

The bytes should be in hexadecimal.



Enter sequence

:8D:81:9D:87:"BBC"
```

the sequence and Memory Display/Disassembly mode is entered. The memory is assumed to consist of a continuous 64K bytes starting at the cursor position, and including the currently selected Paged ROM.

Once a sequence of bytes has been entered, **SHIFT f3** may be pressed to repeat the search from the current cursor position. The sequence of bytes may be found in the buffer which this routine uses to match with memory. It may also find the sequence in its input buffer or in screen memory.

Note: If the search is being carried out on the Second Processor examining I/O memory, it may take some time to find the chosen sequence. If it cannot find it, Monitor will loop round indefinitely. To regain control if this happens press **BREAK.**

Printer (f6)

The contents of memory as shown in Memory Display or Disassembly mode may be sent to the printer. Monitor will ask for the start and end locations of the memory block to be printed.

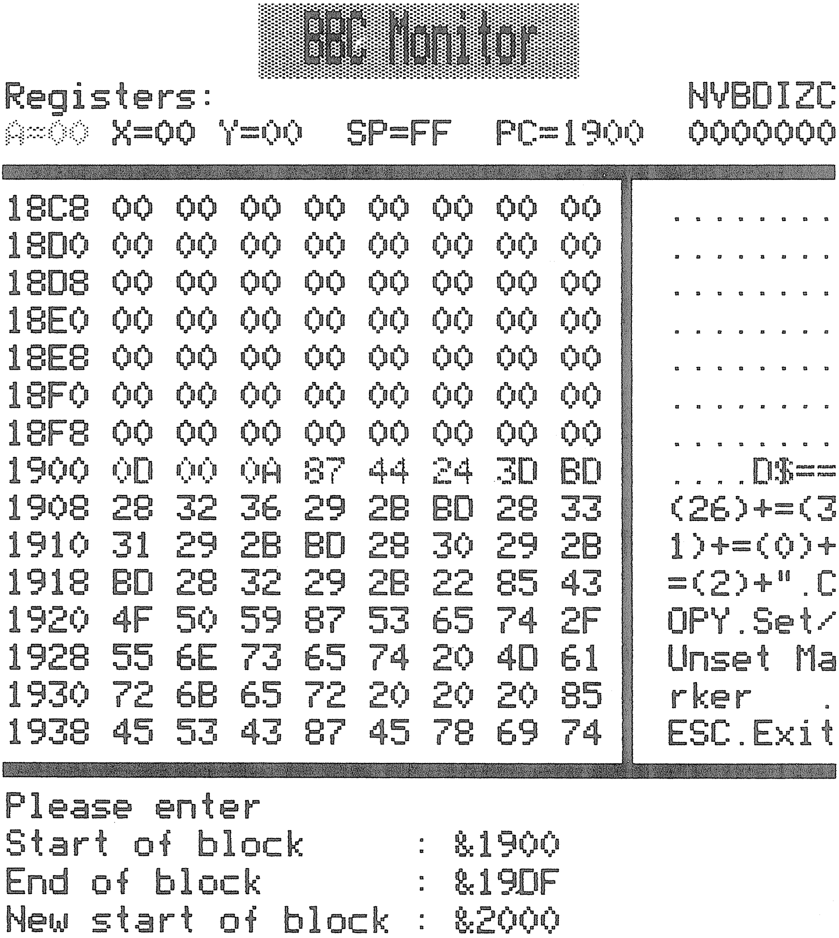
```
1900 A9 0C 20 EE FF A9 20 20   )∘ n∘)
1908 EE FF 18 69 01 C9 7F D0   n∘∘i∘I∘P
1910 F6 60 00 00 00 00 00 00   v`∘∘∘∘∘∘
1918 00 00 00 00 00 00 00 00   ∘∘∘∘∘∘∘∘
1920 00 00 00 00 00 00 00 00   ∘∘∘∘∘∘∘∘
```

```
1900 Start
1900 A9 0C     LDA #&0C         )∘
1902 20 EE FF  JSR OSWRCH       n∘
1905 A9 20     LDA #&20         )
1907 Loop
1907 20 EE FF  JSR OSWRCH       n∘
190A 18        CLC              ∘
190B 69 01     ADC #&01         i∘
190D C9 7F     CMP #&7F         I∘
190F D0 F6     BNE Loop         Pv
1911 60        RTS              £
```

Move a section of memory (f7)

A section of memory may be copied to another position. The new copy may overlap in any way with the original copy and in that case, will correctly overwrite the original with the copy. Monitor will ask for the start and end location of the memory block to be copied and then the location to which the block is to be moved.

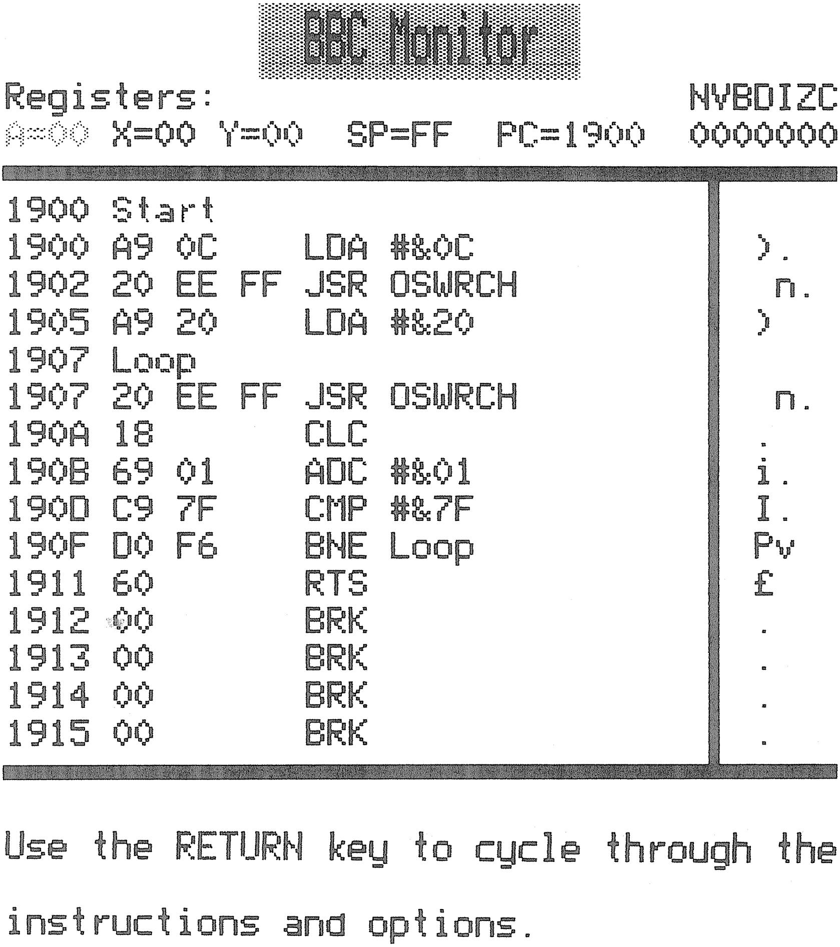
Note: This routine may take some time if the code being copied resides in a sideways ROM.



```
                 BBC Monitor
Registers:                          NVBDIZC
A=00  X=00  Y=00   SP=FF   PC=1900  0000000

18C8 00 00 00 00 00 00 00 00   ........
18D0 00 00 00 00 00 00 00 00   ........
18D8 00 00 00 00 00 00 00 00   ........
18E0 00 00 00 00 00 00 00 00   ........
18E8 00 00 00 00 00 00 00 00   ........
18F0 00 00 00 00 00 00 00 00   ........
18F8 00 00 00 00 00 00 00 00   ........
1900 0D 00 0A 87 44 24 3D BD   ....D$==
1908 28 32 36 29 2B BD 28 33   (26)+=(3
1910 31 29 2B BD 28 30 29 2B   1)+=(0)+
1918 BD 28 32 29 2B 22 85 43   =(2)+".C
1920 4F 50 59 87 53 65 74 2F   OPY.Set/
1928 55 6E 73 65 74 20 4D 61   Unset Ma
1930 72 6B 65 72 20 20 20 85   rker   .
1938 45 53 43 87 45 78 69 74   ESC.Exit

Please enter
Start of block      : &1900
End of block        : &190F
New start of block  : &2000
```

3 DISASSEMBLING & ASSEMBLING MACHINE CODE

Disassembly mode (TAB)
In Disassembly mode, Monitor interprets the contents of memory as assembly language mnemonics starting at the current cursor position. The **CURSOR-DOWN** key may be used to disassemble the next instruction. The **CURSOR-UP** key will start disassembly 8 bytes before



```
                         BBC Monitor
Registers:                                    NVBDIZC
A=00  X=00  Y=00    SP=FF    PC=1900          0000000

1900 Start
1900 A9 0C       LDA #&0C                 >.
1902 20 EE FF    JSR OSWRCH                n.
1905 A9 20       LDA #&20                 >
1907 Loop
1907 20 EE FF    JSR OSWRCH                n.
190A 18          CLC                      .
190B 69 01       ADC #&01                 i.
190D C9 7F       CMP #&7F                 I.
190F D0 F6       BNE Loop                 Pv
1911 60          RTS                      £
1912 00          BRK                      .
1913 00          BRK                      .
1914 00          BRK                      .
1915 00          BRK                      .

Use the RETURN key to cycle through the

instructions and options.
```

the instruction at the top of the screen, and adjust the display. **SHIFT CURSOR DOWN** will start disassembly from the instruction at the bottom of the screen. If a location is labelled the label will be displayed with the disassembly underneath. Breakpoints will be displayed as an extra red label, e.g. BRKpoint 3. The **TAB** key toggles between Disassembly and Memory Display modes.

Labels System (SHIFT f2)

Monitor has a facility which allows memory locations to be labelled. By default Monitor will handle up to 50 labels but this can be extended if necessary. These can be loaded from and saved to disc or tape using options in the LABEL menu. The labels can be used in the assembler as well as in the other numerical input routines. The Operating System labels designated by Acorn are always available (OSBYTE, OSWRCH etc.). Labels can be up to 8 characters long and must start with a letter. Upper and lower case are considered to be different, therefore 'TEST' is not the same as 'Test'.

A separate BASIC program is provided on the utility tape which allows label files to be converted to ASCII files and vice versa, as well as allowing them to be printed out.

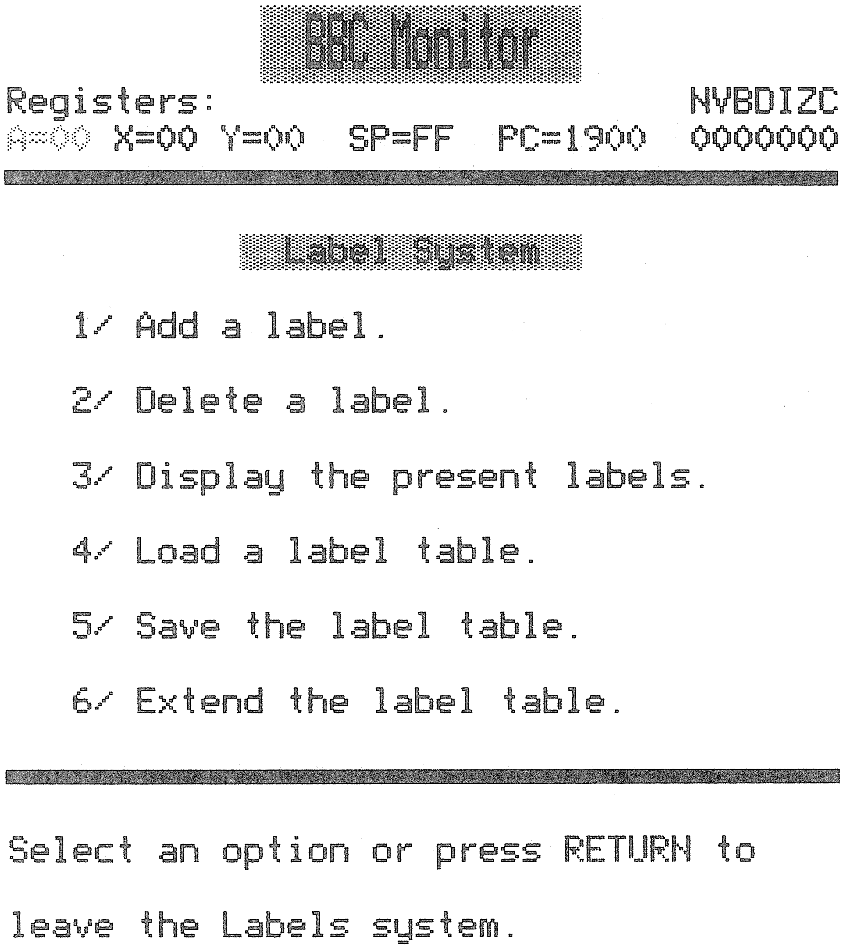
Extend Labels **(Label Option 6)** This facility allows you to move and or extend the label table. You are first asked for the start and end locations of the memory area to be used for labels. You should choose an address in the user area of memory (see Appendices A and C). Monitor will then create a blank label table between these two locations with room for as many labels as will fit. If you wish to preserve any labels previously defined, you should save them before extending or moving the table. The labels can then be loaded into the new table using the load label table option.

Note: Loading in a table which is larger than the area set aside for labels may cause Monitor to crash.

Label (Program on the Monitor tape)

This program enables you to:

Print out label files
Convert label files into ASCII files
Convert ASCII files into label files



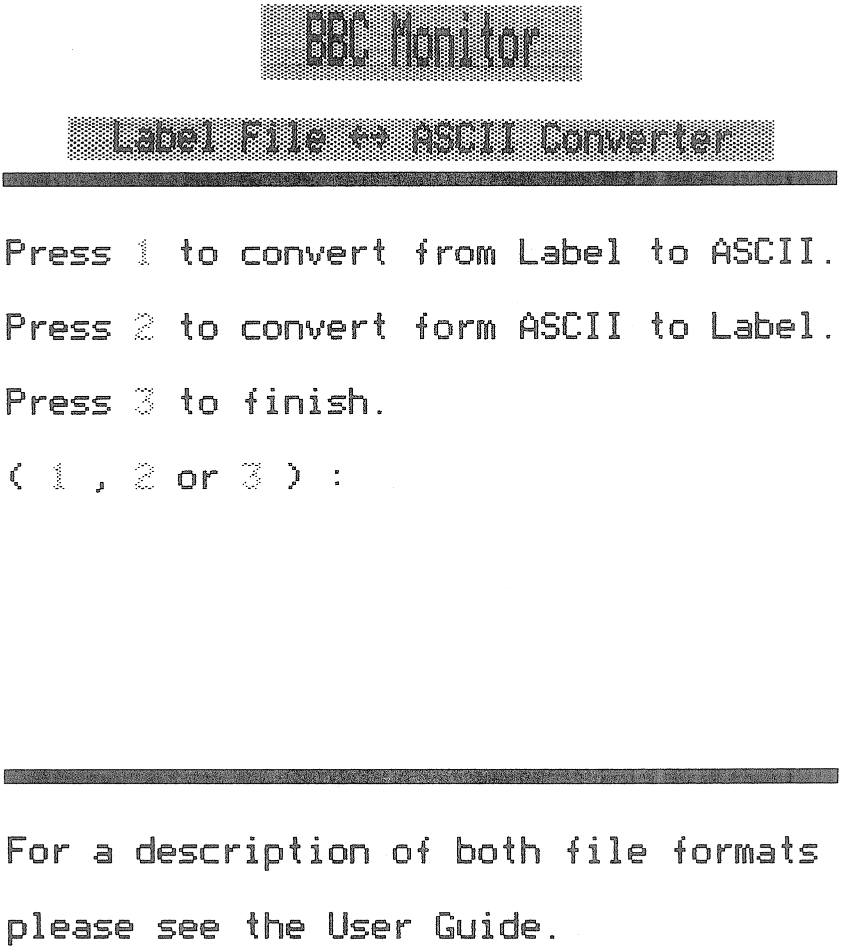
```
                 BBC Monitor
Registers:                               NVBDIZC
A=00 X=00 Y=00  SP=FF  PC=1900  0000000

                Label System

   1/ Add a label.

   2/ Delete a label.

   3/ Display the present labels.

   4/ Load a label table.

   5/ Save the label table.

   6/ Extend the label table.

Select an option or press RETURN to

leave the Labels system.
```

Monitor stores its labels in a compressed format and this is what is saved when the ''SAVE THE LABEL TABLE'' option is selected. If the file needs to be edited or printed the program LABEL is used.

Option one converts a label file (i.e. compressed format) into an ASCII format file which can then be edited using a text editor such as MonEdit. Alternatively, it can display the labels on the screen or on a printer.

Option two is the opposite to Option one in that it converts an ASCII file into a compressed format label file suitable for use with Monitor.

See Appendix C for details of the two file formats.



BBC Monitor

Label File ↔ ASCII Converter

Press 1 to convert from Label to ASCII.

Press 2 to convert form ASCII to Label.

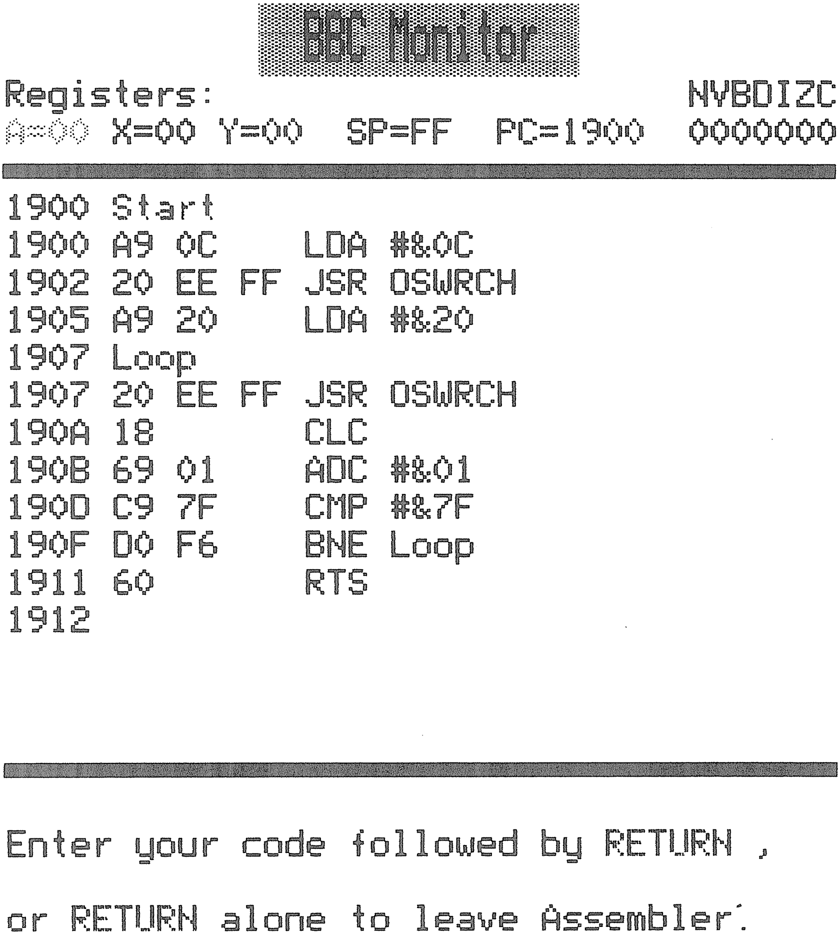
Press 3 to finish.

(1 , 2 or 3) :

For a description of both file formats please see the User Guide.

Assembler (f2)

The single pass assembler is not intended to replace BBC BASIC's full assembler, but to allow you to enter small sections of code without leaving Monitor. The assembler supports 6502 and 65C02 instructions.



```
                  BBC Monitor
Registers:                                NVBDIZC
A=00 X=00 Y=00   SP=FF   PC=1900          0000000

1900 Start
1900 A9 0C      LDA #&0C
1902 20 EE FF   JSR OSWRCH
1905 A9 20      LDA #&20
1907 Loop
1907 20 EE FF   JSR OSWRCH
190A 18         CLC
190B 69 01      ADC #&01
190D C9 7F      CMP #&7F
190F D0 F6      BNE Loop
1911 60         RTS
1912

Enter your code followed by RETURN ,

or RETURN alone to leave Assembler.
```

Source code entered at the cursor position is immediately assembled and displayed. The assembler uses the expression evaluator (see CALCULATE) for all numerical formats, so that numbers may be entered in decimal or hexadecimal, or as labels, or any combination of these. In addition, numerical expressions that evaluate to 16 bits (such as memory addresses), may be preceded by "<" or ">", in which case the high or low byte of the expression, respectively, is returned.

A separate format allows the equivalent of the basic ASC function: the double quote ("), followed by a character, yields the ASCII value of that character.

Labels may be defined at the current cursor position, using the 'full-stop' notation of the BBC Basic Assembler.

An example of the above is:

```
LDA #>BUFFER+10
STA &70
LDA #<BUFFER+10
STA &71
LDA #"!"
LDY #30
.Label STA (&70),Y
DEY
BPL Label
RTS
```

This code will fill 31 bytes from BUFFER + 10 with the ASCII value of '!' (where BUFFER is a previously defined label).

Assembly will start from the current memory pointer value (not the PC value).

Note: HIMON also implements three pseudo-instructions. These are EQUB, EQUW and EQUS. These work in the same way as BASIC 2's instructions. Note that the string argument for EQUS should be enclosed in double quotes (").

Assembler errors

The assembler reports all errors encountered during assembly by giving one of the following error codes:

Error number	Meaning
&1	'')'' expected
&2	spurious meaning
&3	label begins with non-letter
&4	unknown mnemonic
&5	'','' expected
&6	''X'' expected
&7	''Y'' expected
&8	invalid mnemonic
&9	addressing mode does not match mnemonic
&A	relative jump out of range
&B	defined label too long
&C	illegal pseudo-op

The assembler also produces errors when labels are defined wrongly: ''Label already used'' or ''Address already labelled''. (If the assembler encounters a labelled address, the label is printed before code is input.)

If any of the above errors are encountered, the assembler first reports the error, and then allows you to re-enter the line.

Correct code is disassembled after assembly, to standardise screen format. This means that if you enter an operand as an expression, it will be converted on screen to the result of that expression.

In order to exit from the assembler, you should enter **RETURN** on its own when a line of text is expected.

Disc assembler – (SHIFT-f7) in HIMON and Assem – (Program on the Monitor tape)

Assem will only work if Monitor is the currently selected language.

The disc assembler allows you to create your assembly code using a wordprocessor or text editor and then have it entered into Monitor automatically. This means that the 'source code' (i.e. the file from the text editor) can be amended or added to and then re-entered quickly, saving a lot of typing. Source code can be prepared using MonEdit (see Section 6) or a wordprocessor such as VIEW or WORDWISE.

The disc assembler is run by typing * ASSEM **RETURN** (* RUN ASSEM if you are using tape) or by pressing **SHIFT-f7** (HIMON only). You are then asked for the name of the file containing the source code and the address at which the code is to be assembled. Note that ASSEM resides at &7A00 to &7C00, so assembling code to that area will cause a crash which can be rectified by pressing **BREAK**, and reloading ASSEM.

The disc assembler will perform a two-pass assembly which allows labels to be referenced before they are defined e.g.:

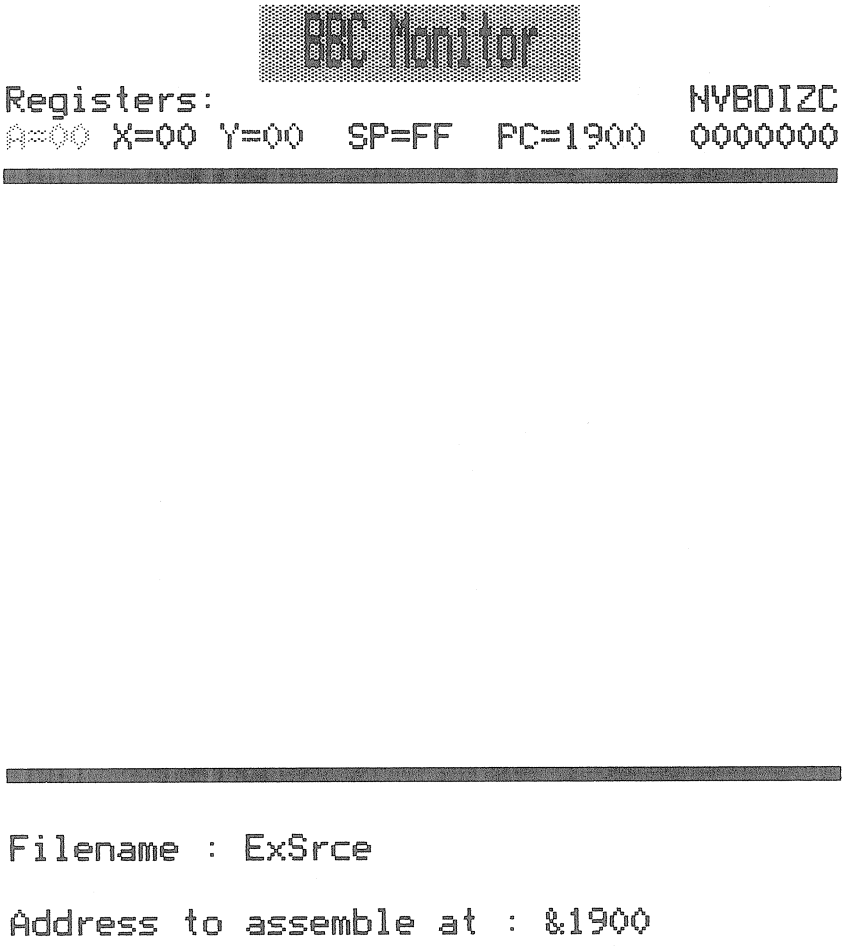
```
        CMP #12
        BNE label       \ this label not yet defined
        LDA #7
        JSR OSWRCH
.label RTS              \ defines label
```

The disc assembler also allows comments using a \ (the half symbol in teletext) in the same way as the BASIC assembler. If you wish to have a blank line separating sections of your programs you must start it with a comment symbol. Otherwise Monitor thinks you have finished your program. Any leading spaces will be ignored when assembling.

Note: Only the first 24 characters of a line (not including leading spaces) will be read by the assembler, so lines such as:
. longline LDA memory + start − end + 12
will be truncated and generate problems. Therefore keep your lines short. Alternatively, put label definitions on a line of their own, e.g:
. longline
LDA memory + start − end + 12
A typical source file is included on the utility tape called ''ExSrce''.

Should there be an error in your source code, the assembler will stop and report the error. Pressing **SPACE** as instructed may cause the assembler to try a second time. This will do no harm but may be stopped by pressing **BREAK** instead.

Note that to get useful results, the source code that you prepare must conform to the requirements of the assembler i.e. as per the BBC BASIC assembler without line numbers. If you prepare the source code in a wordprocessor, do not embed any commands or margins.

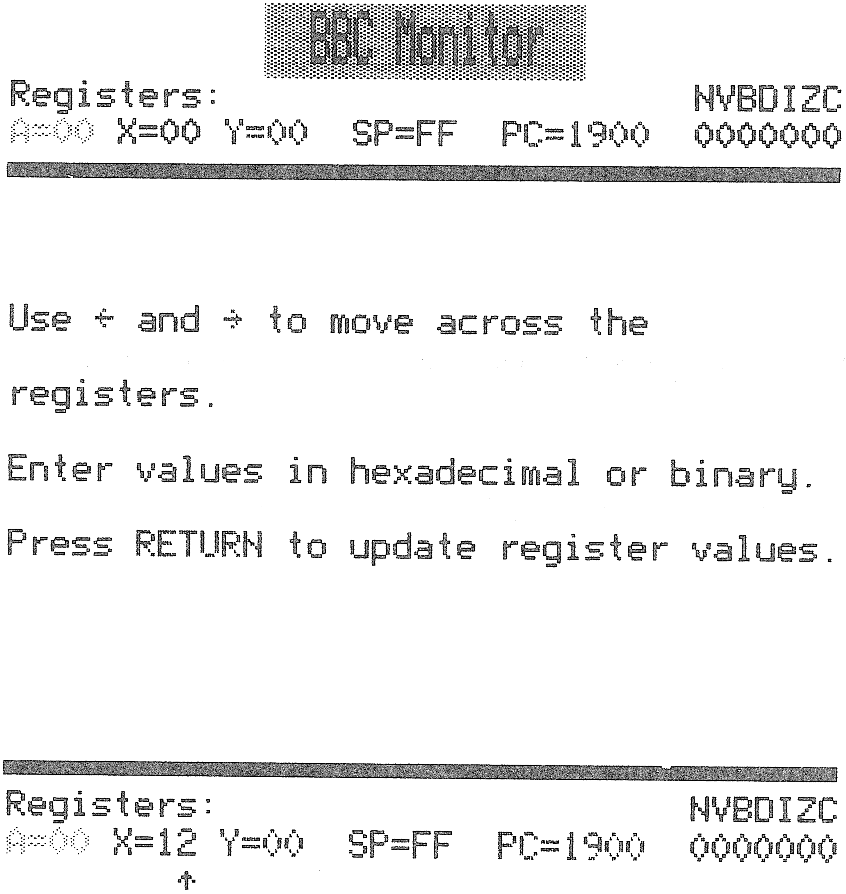


```
                 BBC Monitor
Registers:                               NVBDIZC
A=00 X=00 Y=00  SP=FF  PC=1900          0000000



Filename : ExSrce

Address to assemble at : &1900
```

4 TRACING AND EXECUTING MACHINE CODE

Alter registers and flags (f1)

Monitor allows you to execute and control machine code programs for test and debugging purposes – indeed, this is its prime function. Before a machine code routine can be executed, the program counter (PC) must be set, and where required entry values of the registers and flags. The program counter, as set by this option defines where tracing **(f4)** or execution **(f5)** will begin, and *not* the memory pointer which



```
                    BBC Monitor
Registers:                                   NVBDIZC
A=00 X=00 Y=00  SP=FF  PC=1900   0000000

Use ← and → to move across the
registers.
Enter values in hexadecimal or binary.
Press RETURN to update register values.

Registers:                                   NVBDIZC
A=00 X=12 Y=00  SP=FF  PC=1900   0000000
        ↑
```

determines which section of memory is displayed. These separate pointers allow code to be traced at one location while memory elsewhere is displayed.

The cursor keys are used to move between the 6502 registers. New values are assumed to be in hex and are entered directly from the keyboard, using the same rotating hex keying as in the memory editor. Registers available are A, X, Y, SP and PC. The Processor Status (Flags) register may also be altered by entering values in binary. The new values are confirmed by pressing **RETURN.**

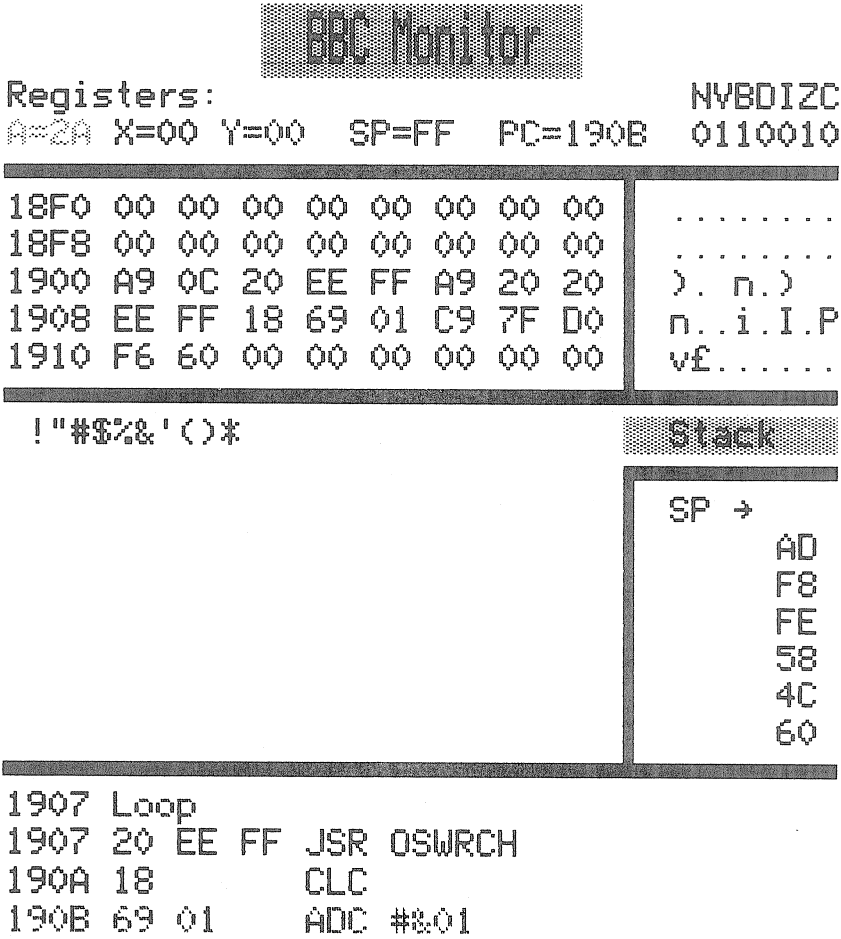
Note: The value of the stack pointer (SP) should be greater than &7F as the stack below this value is used by Monitor. Under normal circumstances, it is very unlikely that you will need to worry about this.

Trace mode (f4)

Monitor attempts to interpret the contents of memory at the location in program counter (PC) as a 65C02 instruction. If the contents of that location cannot be interpreted, no action is taken by Monitor. If the contents are &00 (BRK), Monitor checks to see if it is a breakpoint and if so informs you that a breakpoint has been encountered. Otherwise it displays an error message based on the bytes following the BRK in the normal BBC manner. If the contents are interpreted as JSR (except for JSR to OS calls), Monitor asks whether the subroutine is to be traced; answering 'Yes' causes PC and stack pointer (SP) to be altered and the stack updated, and Monitor waits for the next command; answering 'No' causes Monitor to transfer control to the subroutine, and returns control to Monitor when the subroutine is completed. In all other cases, the single 6502 instruction is executed, and the registers, flags, memory display and stack etc. are altered accordingly and Monitor waits for the next command. If **f4** is now pressed again the next instruction will be traced. Any other key will cause the trace routine to jump back to the main Monitor program.

The display shows the area of memory either side of the cursor position in the top left-hand corner. The right-hand side shows the last few bytes on the stack. The rest of the central section is used to display the output produced by the code being traced. Alternatively, this output may be sent to another computer via the RS423 port (see SET OPTIONS). The bottom area of the screen shows the last instructions to have been executed. The instruction displayed in purple is the instruction which will be executed next (unless the PC is altered).

There are certain restrictions which are inevitable with the operation of tracing code. Tracing an input or output routine one instruction at a time while expecting Monitor to output simultaneously inevitably leads to trouble. For this reason, Monitor will only execute OS calls as complete subroutines. Monitor traps all VDU control codes and prevents them from corrupting the display, but in the case of the trace-screen (see SET OPTIONS) these codes are indicated as hex values on the trace-screen. Normal VDU codes (32–127) are printed on the trace screen. Monitor itself uses certain parts of memory, and is thus



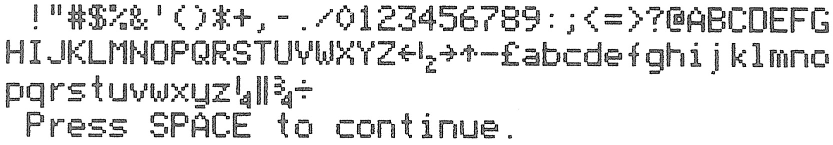
```
                     BBC Monitor
Registers:                                    NVBDIZC
A=2A X=00 Y=00  SP=FF  PC=190B  0110010

18F0 00 00 00 00 00 00 00 00    ........
18F8 00 00 00 00 00 00 00 00    ........
1900 A9 0C 20 EE FF A9 20 20    ). n.)
1908 EE FF 18 69 01 C9 7F D0    n..i.I.P
1910 F6 60 00 00 00 00 00 00    v£......

 !"#$%&'()*                          Stack

                                 SP →
                                          AD
                                          F8
                                          FE
                                          58
                                          4C
                                          60

1907 Loop
1907 20 EE FF JSR OSWRCH
190A 18       CLC
190B 69 01    ADC #&01
```

unable to trace code which attempts to use the same locations. The locations used by Monitor are &50 to &6F, &400 to &7FF, in all cases, and &7800 to &7BFF (in the I/O Processor) or &C000 to &C3FF (in the Second Processor) or &AC00 to &B000 (when using HIMON). Monitor uses the stack from &7F down to &00 and your code should use the stack from &FF down to &80.

Clear trace-screen (CTRL L)
This clears the present trace-screen and can be used whenever a blank screen is required. It should also be used after altering the position of the trace-screen, or after using ASSEM.

JSR to program counter (f5)
A complete subroutine may be executed at normal speed with this option. The return address for Monitor is placed on the stack, and execution begins at the location given in PC. The chosen ROM (if applicable) is also selected.

Note: This facility is blocked if you are examining I/O memory from the Second Processor.



```
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFG
HIJKLMNOPQRSTUVWXYZ←½→↑—£abcdefghijklmno
pqrstuvwxyz¼‖¾÷
 Press SPACE to continue.
```

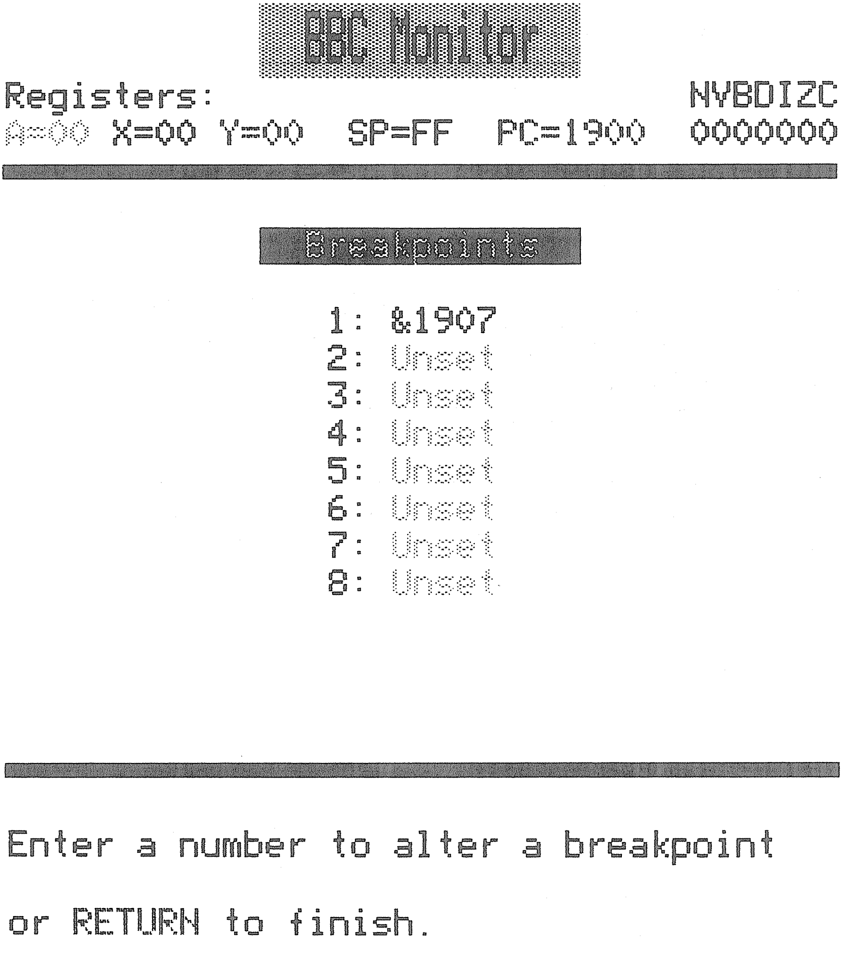
JMP to program counter (SHIFT f5)
This function is much the same as the JSR except that it does not place a return address on the stack. It can therefore be used to continue executing a piece of code after a breakpoint.

Breakpoints (f8)
Breakpoints are a useful device to halt the execution of machine code when it is running at full speed, e.g. when using JSR and non-traced subroutines in TRACE.

Breakpoint locations are entered by first giving the number of the breakpoint to be altered and then entering the memory location at which the breakpoint is to be set. Pressing **RETURN** on its own will unset a breakpoint. They are shown in disassemblies as an additional red label giving the breakpoint number. Breakpoints cannot be set in ROM or in page &FF.

When a breakpoint is encountered in a routine that has been called by JSR **(f5)** or JMP **(SHIFT-F5)**, Monitor will stop executing the code and display the breakpoint number and its location. If a breakpoint is encountered in a subroutine which is being executed, Monitor will display the location at which the breakpoint was encountered. You will then be able to continue tracing.

If you wish to carry on executing the code, using JMP **(SHIFT-f5)**, you will have to clear the breakpoint first. Alternatively you can TRACE the instruction where the breakpoint is set and then use JMP.



```
                    BBC Monitor
Registers:                                NVBDIZC
A=00 X=00 Y=00  SP=FF  PC=1900           0000000

                    Breakpoints

                    1: &1907
                    2: Unset
                    3: Unset
                    4: Unset
                    5: Unset
                    6: Unset
                    7: Unset
                    8: Unset

Enter a number to alter a breakpoint

or RETURN to finish.
```

Note: If a subroutine is being run from either Trace or JSR and it crashes, **BREAK** should be used, *not* **CTRL-BREAK**, to escape from it. This is because pressing **CTRL-BREAK** will leave zeros in any locations where breakpoints were set.

RS423 (Program on Monitor tape)

If you wish to make use of the Trace Output facility (see SET OPTIONS) of Monitor to send the output of the code being traced to another Acorn computer, then the program RS423 should be loaded into the other computer and run before starting to trace the code. It turns the second computer into a 'dumb' terminal which listens to the computer running Monitor. If you wish to use this 'dual screen' trace with a computer that does not use BBC BASIC, you will have to study the listing and make the necessary conversion.

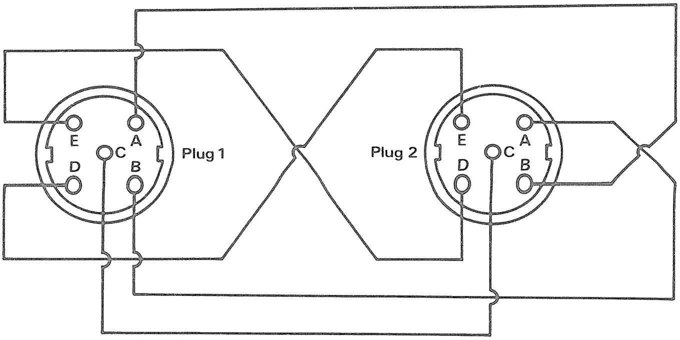
To use the facility, you will need to take up a suitable cable to connect the computers, and details of a cable to connect two Acorn computers are given below.

Details of RS423-RS423 cable for two Acorn microcomputers

The connections are as follows:

Plug 1		**Plug 2**
Ground	↔	Ground
CTS	↔	RTS
RTS	↔	CTS
Data in	↔	Data out
Data out	↔	Data in

The diagram below shows these connections. The plugs are viewed from the rear (i.e. they show the cable side of the plug rather than the pin side)



A	Data in
B	Data out
C	Ground
D	CTS
E	RTS

The cable can also be used outside of Monitor to allow two computers to communicate with each other via the RS423. For example, to send a program from computer A to computer B do as follows:

Each computer should be set up with the following function keys:

```
*KEY0 *FX8,8|M*FX7,8|M*FX3,5|ML.|MP."*FX2,0"|M*FX3|M

*KEY1 *FX8,8|M*FX7,8|MNEW|M*FX2,1|M
```

Press **f1** on computer B. This sets it up ready to receive the program.

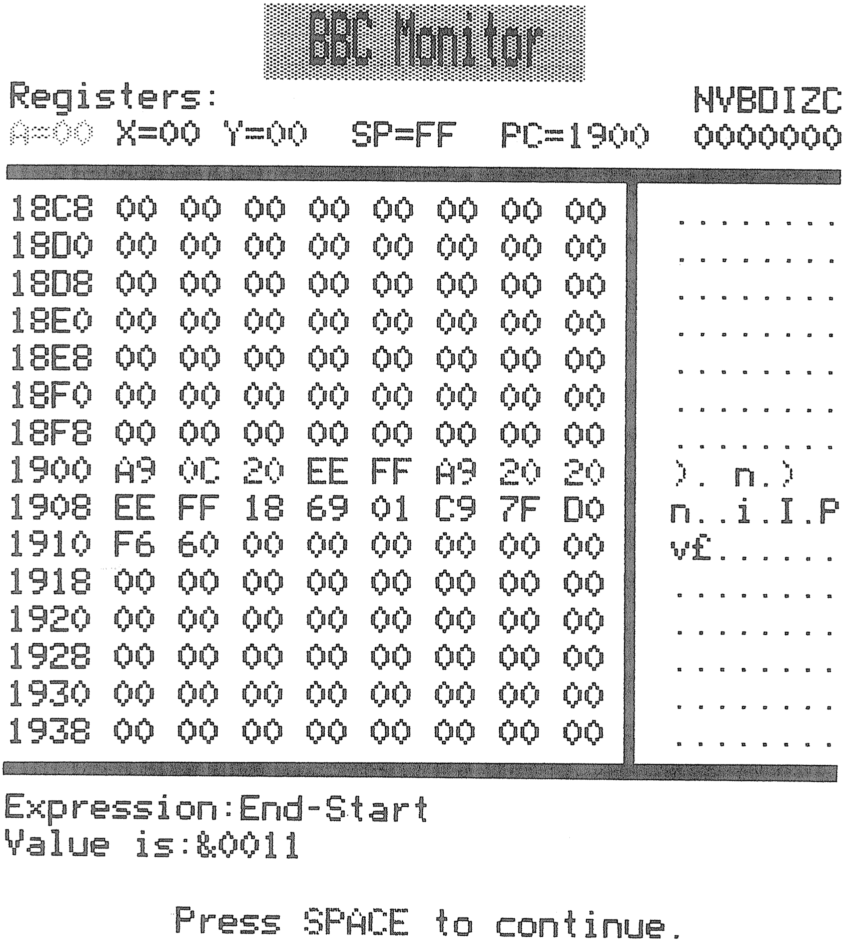
Press **f0** on computer A. This transmits the program to computer B.

The program is now in computer B. If you have difficulty it may help to reduce the baud rate settings (* FX7 and * FX8). See the BBC Micro User Guide for details.

This procedure is useful if the machine on the other end of the RS423 cable has not got a disc drive. You can use the above procedure to transfer the program ''RS423'' to the other computer before starting Monitor.

5 ADDITIONAL COMMANDS

Calculate (SHIFT f6)
This routine allows you to do simple arithmetic using hex, decimal and labels. Only addition and subtraction are allowed. The result is given in hexadecimal. The input is checked for errors as it is typed and illegal characters will be ignored. If an unknown label is encountered an error is produced and you can then re-enter the calculation.



```
                         BBC Monitor
Registers:                                        NVBDIZC
A=00  X=00  Y=00   SP=FF   PC=1900   0000000

18C8 00 00 00 00 00 00 00 00    . . . . . . . .
18D0 00 00 00 00 00 00 00 00    . . . . . . . .
18D8 00 00 00 00 00 00 00 00    . . . . . . . .
18E0 00 00 00 00 00 00 00 00    . . . . . . . .
18E8 00 00 00 00 00 00 00 00    . . . . . . . .
18F0 00 00 00 00 00 00 00 00    . . . . . . . .
18F8 00 00 00 00 00 00 00 00    . . . . . . . .
1900 A9 0C 20 EE FF A9 20 20    ) .   n . )
1908 EE FF 18 69 01 C9 7F D0    n . . i . I . P
1910 F6 60 00 00 00 00 00 00    v £ . . . . . .
1918 00 00 00 00 00 00 00 00    . . . . . . . .
1920 00 00 00 00 00 00 00 00    . . . . . . . .
1928 00 00 00 00 00 00 00 00    . . . . . . . .
1930 00 00 00 00 00 00 00 00    . . . . . . . .
1938 00 00 00 00 00 00 00 00    . . . . . . . .

Expression:End-Start
Value is:&0011

             Press SPACE to continue.
```

This routine is called whenever a numerical input is required (e.g. ''MOVE TO''). It is therefore possible to move to a calculated address, e.g. LABEL + &F2 + 5. In most cases, an ampersand (&) will be displayed, as experience has shown that a hexadecimal value is usually entered, as this is how Monitor displays addresses. If this is *not* so, the ampersand can be deleted using the **DELETE** key.

Pressing **COPY** when a numeric input in requested will enter the cursor's present position in memory. This can be useful for setting breakpoints or defining Labels.

Set options (SHIFT f4)

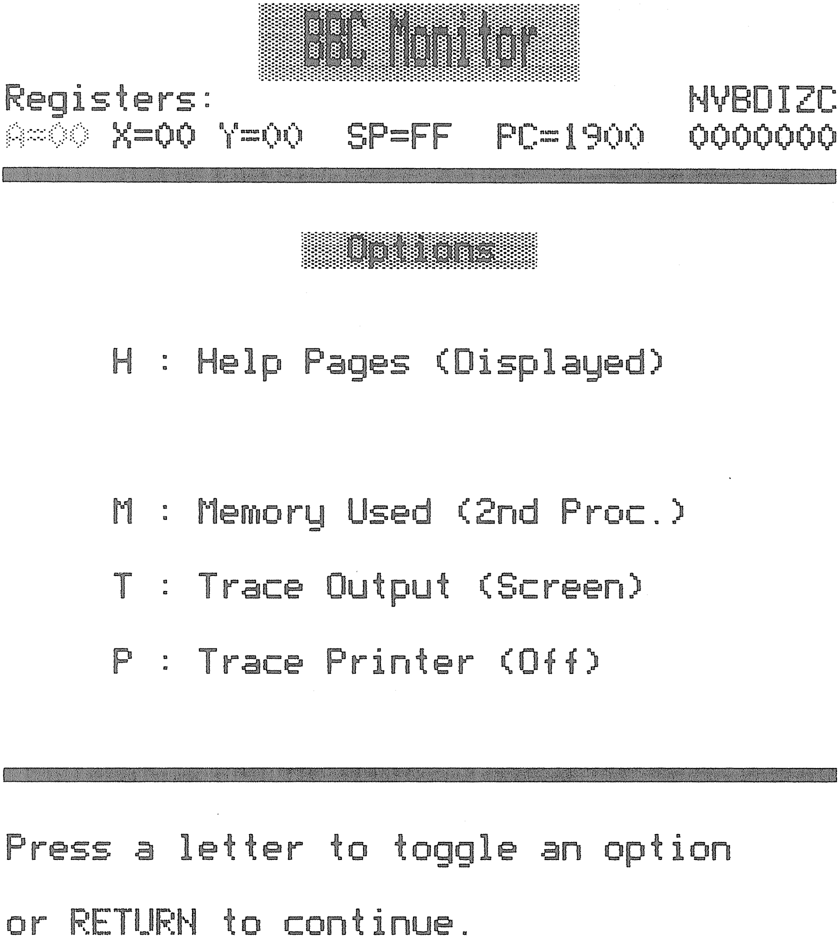
Monitor has a number of options which can be controlled by this command:

Help pages: Certain functions such as Find and Alter Registers clear the main section of the screen in order to display a help message. This can be prevented by setting HELP PAGES to ''not displayed''.

Shadow Memory: BBC Microcomputers with OS2.0 or greater have a system whereby the screen memory is separate from the main memory. By selecting ''In'' on this option Monitor will examine the separate SHADOW memory. If ''Out'' is selected Monitor will examine the normal memory. Note that this option is only displayed if the I/O Processor memory is being examined and the computer has OS2.0 or greater.

Memory Used: If a 6502 Second Processor is attached, this option allows you to select whether the Second Processor or I/O Procesor memory is examined.

Note: When Monitor is running in the Second Processor it cannot execute or trace code in the I/O Processor, but it can assemble code into the I/O Processor's memory. Code can be executed by setting USERV(&200 in the I/O Processor), to point to the code and then using * CODE or OSBYTE &88. This command will be transferred across the tube and cause the I/O Processor to jump to the code pointed to by USERV (&200,&201).



```
                 BBC Monitor
Registers:                              NVBDIZC
A=00 X=00 Y=00  SP=FF  PC=1900  0000000

                   Options

     H : Help Pages (Displayed)


     M : Memory Used (2nd Proc.)

     T : Trace Output (Screen)

     P : Trace Printer (Off)


Press a letter to toggle an option

or RETURN to continue.
```

Trace output In Trace mode, input and output from the code being traced may be directed to the ''trace-screen'', or to another BBC Micro via the RS423.

The trace-screen is a reduced screen area which shows output ASCII characters and gives the ASCII codes of non-printable characters enclosed by angle brackets. Thus changes of mode and graphics are trapped, but the operation of Monitor remains visible.

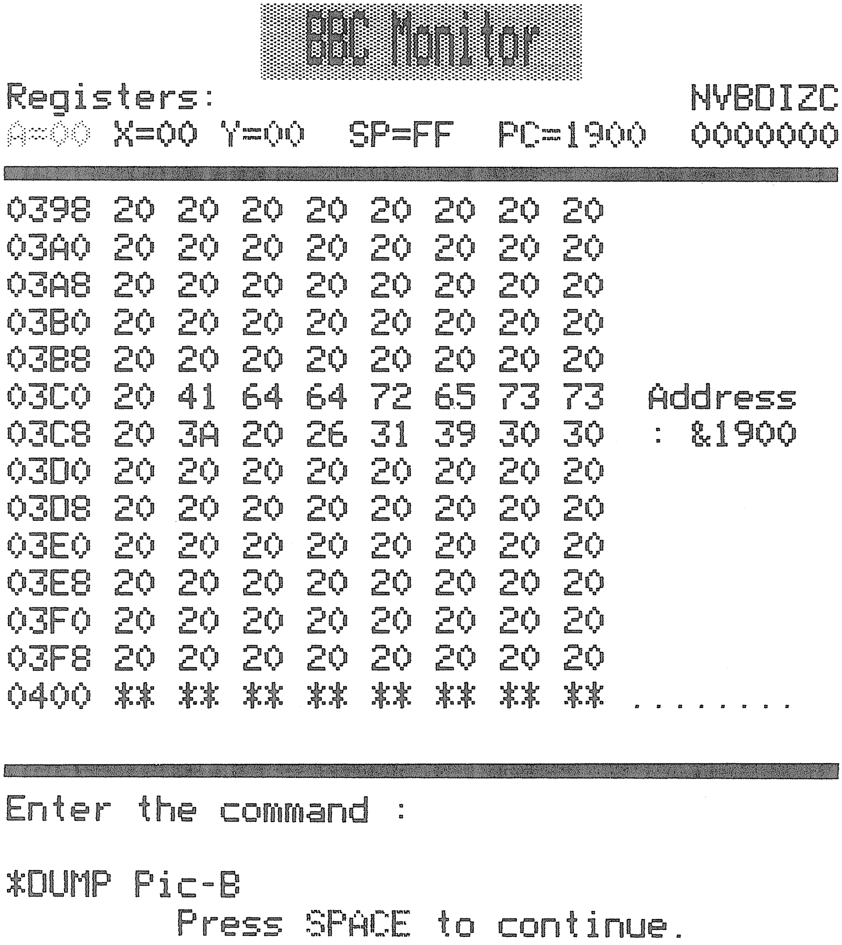
When output is directed to another BBC Micro, the complete display created by the traced code is visible on that computer, while Monitor is visible on its own screen. In this mode, any input requested by the traced code must be supplied by the other computer. Output is sent via the RS423 to the RS423 connection of a second BBC Micro. A short program must be run on this second computer to enable it to receive its input from the computer running Monitor and this is on the utility tape as a BASIC program called "RS423". Details of the cable required to connect the RS423 ports are given under the section on the RS423 program.

The trace-screen uses a section of memory &400 bytes long to store a copy of the trace output. This area is from &7800 upwards on the I/O processor or &C000 on the Second Processor. If these positions are not convenient they can be changed by altering the two byte pointer at &5E0 and &5E1. This points to the start of the &400 byte block. After altering these locations it is advisable to press **CTRL-L** to clear that area of memory otherwise unpredictable events might occur!

```
1900 Start
1900 A9 0C     LDA #&0C              A=0C X=00 Y=00  SP=FF  ..B....
1902 20 EE FF  JSR OSWRCH            A=0C X=00 Y=00  SP=FF  .VB..ZC
1905 A9 20     LDA #&20              A=20 X=00 Y=00  SP=FF  .VB...C
1907 Loop
1907 20 EE FF  JSR OSWRCH            A=20 X=00 Y=00  SP=FF  .VB..ZC
190A 18        CLC                   A=20 X=00 Y=00  SP=FF  .VB..Z.
190B 69 01     ADC #&01              A=21 X=00 Y=00  SP=FF  ..B....
190D C9 7F     CMP #&7F              A=21 X=00 Y=00  SP=FF  N.B....
190F D0 F6     BNE Loop              A=21 X=00 Y=00  SP=FF  N.B....
1907 Loop
1907 20 EE FF  JSR OSWRCH            A=21 X=00 Y=00  SP=FF  .VB..ZC
190A 18        CLC                   A=21 X=00 Y=00  SP=FF  .VB..Z.
190B 69 01     ADC #&01              A=22 X=00 Y=00  SP=FF  ..B....
190D C9 7F     CMP #&7F              A=22 X=00 Y=00  SP=FF  N.B....
190F D0 F6     BNE Loop              A=22 X=00 Y=00  SP=FF  N.B....
1907 Loop
1907 20 EE FF  JSR OSWRCH            A=22 X=00 Y=00  SP=FF  .VB..ZC
190A 18        CLC                   A=22 X=00 Y=00  SP=FF  .VB..Z.
190B 69 01     ADC #&01              A=23 X=00 Y=00  SP=FF  ..B....
190D C9 7F     CMP #&7F              A=23 X=00 Y=00  SP=FF  N.B....
190F D0 F6     BNE Loop              A=23 X=00 Y=00  SP=FF  N.B....
1907 Loop
1907 20 EE FF  JSR OSWRCH            A=23 X=00 Y=00  SP=FF  .VB..ZC
190A 18        CLC                   A=23 X=00 Y=00  SP=FF  .VB..Z.
190B 69 01     ADC #&01              A=24 X=00 Y=00  SP=FF  ..B....
190D C9 7F     CMP #&7F              A=24 X=00 Y=00  SP=FF  N.B....
190F D0 F6     BNE Loop              A=24 X=00 Y=00  SP=FF  N.B....
1907 Loop
1907 20 EE FF  JSR OSWRCH            A=24 X=00 Y=00  SP=FF  .VB..ZC
190A 18        CLC                   A=24 X=00 Y=00  SP=FF  .VB..Z.
190B 69 01     ADC #&01              A=25 X=00 Y=00  SP=FF  ..B....
190D C9 7F     CMP #&7F              A=25 X=00 Y=00  SP=FF  N.B....
190F D0 F6     BNE Loop              A=25 X=00 Y=00  SP=FF  N.B....
```

Trace printer This facility allows you to print out the instructions being traced, and the value of the registers and flags after each instruction has been executed.

Operating System commands (*)
The Operating System commands are always available from Monitor. The initial * is supplied as a prompt. Pressing **RETURN** on its own will return you to Monitor. Operating System commands will be executed, but transfer of control to a different language (such as BASIC) may cause unpredictable effects.



```
                BBC Monitor
Registers:                              NVBDIZC
A=00  X=00  Y=00   SP=FF   PC=1900      0000000

0398 20 20 20 20 20 20 20 20
03A0 20 20 20 20 20 20 20 20
03A8 20 20 20 20 20 20 20 20
03B0 20 20 20 20 20 20 20 20
03B8 20 20 20 20 20 20 20 20
03C0 20 41 64 64 72 65 73 73  Address
03C8 20 3A 20 26 31 39 30 30   : &1900
03D0 20 20 20 20 20 20 20 20
03D8 20 20 20 20 20 20 20 20
03E0 20 20 20 20 20 20 20 20
03E8 20 20 20 20 20 20 20 20
03F0 20 20 20 20 20 20 20 20
03F8 20 20 20 20 20 20 20 20
0400 ** ** ** ** ** ** ** **  ........

Enter the command :

*DUMP Pic-B
        Press SPACE to continue.
```

The central area of the screen is cleared to display any output generated by the * command. if more than one page of output is generated (e.g. *INFO *.*), successive pages may be displayed by pressing **SHIFT**.

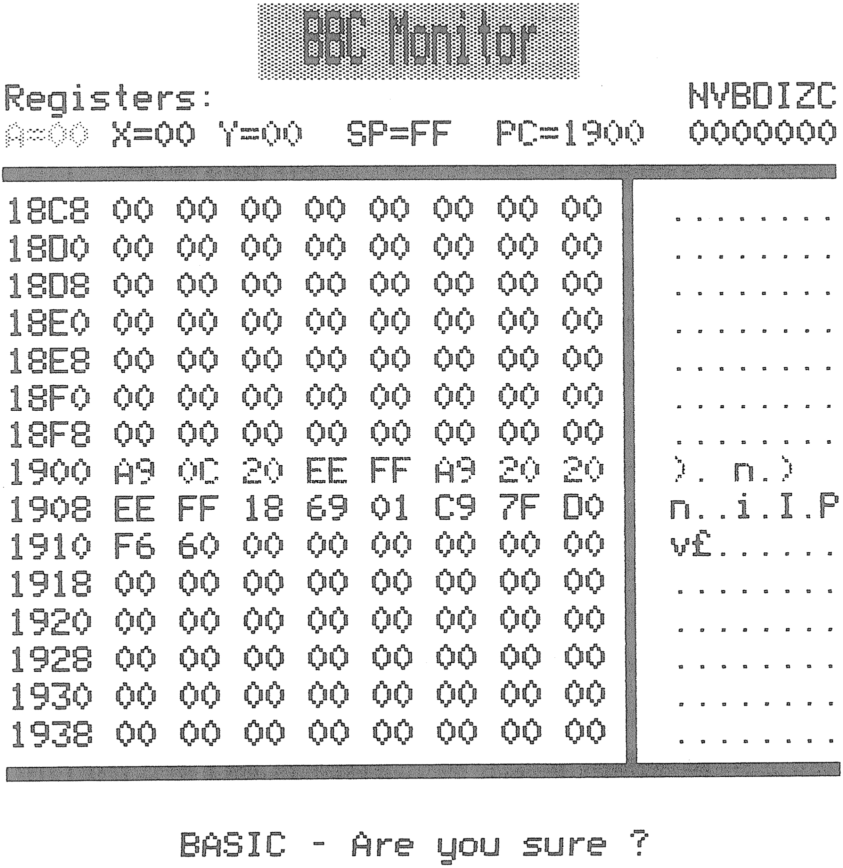
User defined function keys (CTRL-f0 to CTRL-f9)

You can, if you wish, program any or all of the user-function-keys with the normal *KEY command. These can then be used to perform certain repetitive functions. It should be noted however that Monitor clears the input buffer whenever a function is terminated. This includes altering one byte of memory. The sequences for each of the function keys (normal and shifted) and special keys are given below. Remember that to use the strings you have defined with *KEY you must press **CTRL-fkey**.

Key	*KEY SEQUENCE	ASCII CODE
f0	¦!a	225
to	to	to
f9	¦!j	234
SHIFT-f0	¦!n	238
to	to	to
SHIFT-f9	¦!w	247
TAB	¦I	9
RETURN	¦M	13
ESCAPE	¦[	27
CTRL-L	¦L	12
COPY	¦!¦G	135
←	¦!¦H	136
→	¦!¦I	137
↓	¦!¦J	138
↑	¦!¦K	139

Exit (f9)
This command returns you to BASIC. If there is no BASIC chip in the computer this routine will probably cause the computer to crash, as it involves a * BASIC command. You can exit to other languages using a normal ' * ' command i.e. * PASCAL.

If you have created any labels and you wish to keep them, you must first save them before exiting.



```
                     BBC Monitor
Registers:                                   NVBDIZC
A=00  X=00  Y=00    SP=FF    PC=1900         0000000

18C8 00 00 00 00 00 00 00 00     ........
18D0 00 00 00 00 00 00 00 00     ........
18D8 00 00 00 00 00 00 00 00     ........
18E0 00 00 00 00 00 00 00 00     ........
18E8 00 00 00 00 00 00 00 00     ........
18F0 00 00 00 00 00 00 00 00     ........
18F8 00 00 00 00 00 00 00 00     ........
1900 A9 0C 20 EE FF A9 20 20     ). n.)
1908 EE FF 18 69 01 C9 7F D0     n..i.I.P
1910 F6 60 00 00 00 00 00 00     v£......
1918 00 00 00 00 00 00 00 00     ........
1920 00 00 00 00 00 00 00 00     ........
1928 00 00 00 00 00 00 00 00     ........
1930 00 00 00 00 00 00 00 00     ........
1938 00 00 00 00 00 00 00 00     ........

          BASIC - Are you sure ?
```

6 THE EDITOR

MonEdit (A program on the Monitor tape)

This is a simple text editor that will allow you to create text files for use with the disc assembler or the Labels System (via 'Labels').

It creates ASCII format files (the same format as used by VIEW and WORDWISE) up to 450 lines long with a maximum line length of 38 characters. These files can be loaded and saved from the editor or can be displayed using the * TYPE command (disc machines only).

The editor itself allows lines to be inserted or deleted and blocks of text can be deleted or copied to the cursor's position.

The cursor is moved around the text using the cursor keys. Pressing **SHIFT** with a cursor key causes it to move 15 characters or lines instead of 1.

Pressing **RETURN** while in column 0 will cause a blank line to be inserted above the current line.

Pressing **RETURN** anywhere else along a line causes a blank line to be inserted below the current line.

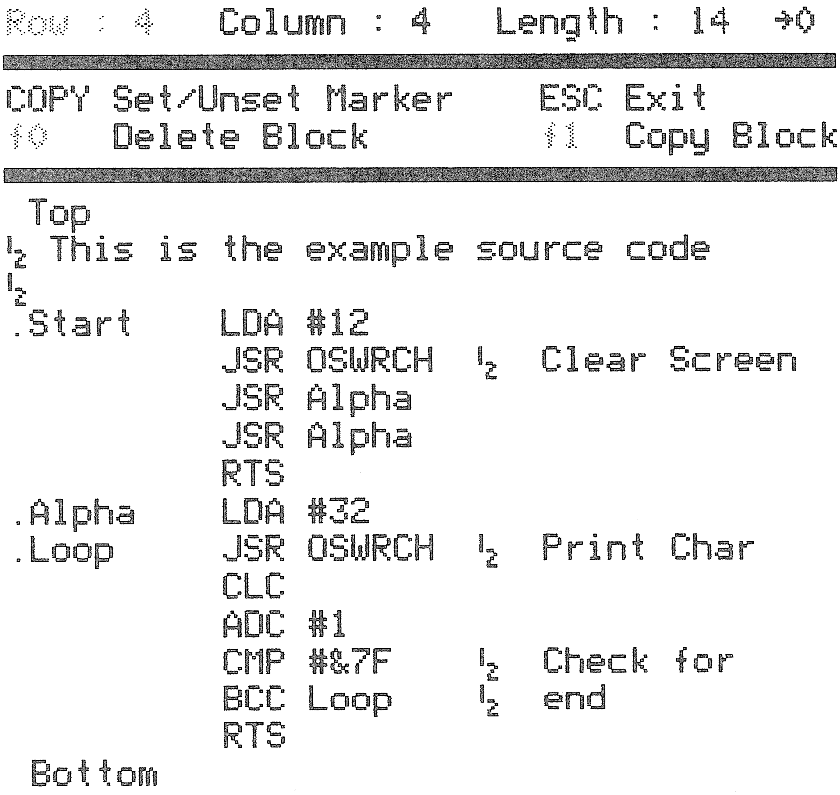
DELETE deletes the character before the cursor. If **DELETE** is pressed with the cursor on the first character of a blank line, then the line is deleted. Note that only blank lines can be deleted in this way so all characters must be deleted from a line before it can be deleted.

TAB moves the cursor to the next tab position. These are at 10, 24 and 37; these values being suitable for the disc assembler source code, which is what the editor was designed for, although simple correspondence and the like should not be difficult to handle.

Two markers can be placed in the text to indicate a block to be copied or deleted. They are set and unset by pressing **COPY** on the appropriate line. The block to be copied/deleted will include the lines with the markers on.

When you exit from the Editor you are given the option of pressing **f0** to re-enter the Editor without losing the text. This can only be guaranteed to work if no BASIC or OS commands or lines have been entered prior to pressing **f0.**

The best way to find out about the editor is simply to try it out on some test pieces of text.



```
Row : 4   Column : 4   Length : 14   →0

COPY Set/Unset Marker     ESC Exit
f0   Delete Block         f1  Copy Block

 Top
½ This is the example source code
½
.Start     LDA #12
           JSR OSWRCH  ½  Clear Screen
           JSR Alpha
           JSR Alpha
           RTS
.Alpha     LDA #32
.Loop      JSR OSWRCH  ½  Print Char
           CLC
           ADC #1
           CMP #&7F    ½  Check for
           BCC Loop    ½  end
           RTS
 Bottom
```

7 THE RELOCATOR

Reloc (Program on Monitor tape)

This utility allows machine code programs to be relocated in memory (i.e. moved). It is activated by typing * RELOC from BASIC or by using the star command (*) function of Monitor and typing RELOC. (Tape users type * RUN RELOC.)

The first thing it does is ask whether you wish to relocate RELOC itself. This allows you to position RELOC itself so that it does not interfere with the code being relocated.

You are then given a series of options:

Relocate Code **(f0)**

This command enters the relocator proper.

Select Paged ROM **(SHIFT-f0)**

This command performs the same function as **SHIFT-f0** in Monitor. It will not work in the Second Processor.

Enter a * command **(*)**

This allows you to load and save sections of memory before or after relocating them, or use other system commands such as * DRIVE.

Exit to Monitor **(f9)**

This command returns you to Monitor. **CTRL-BREAK** will return you to the default language.

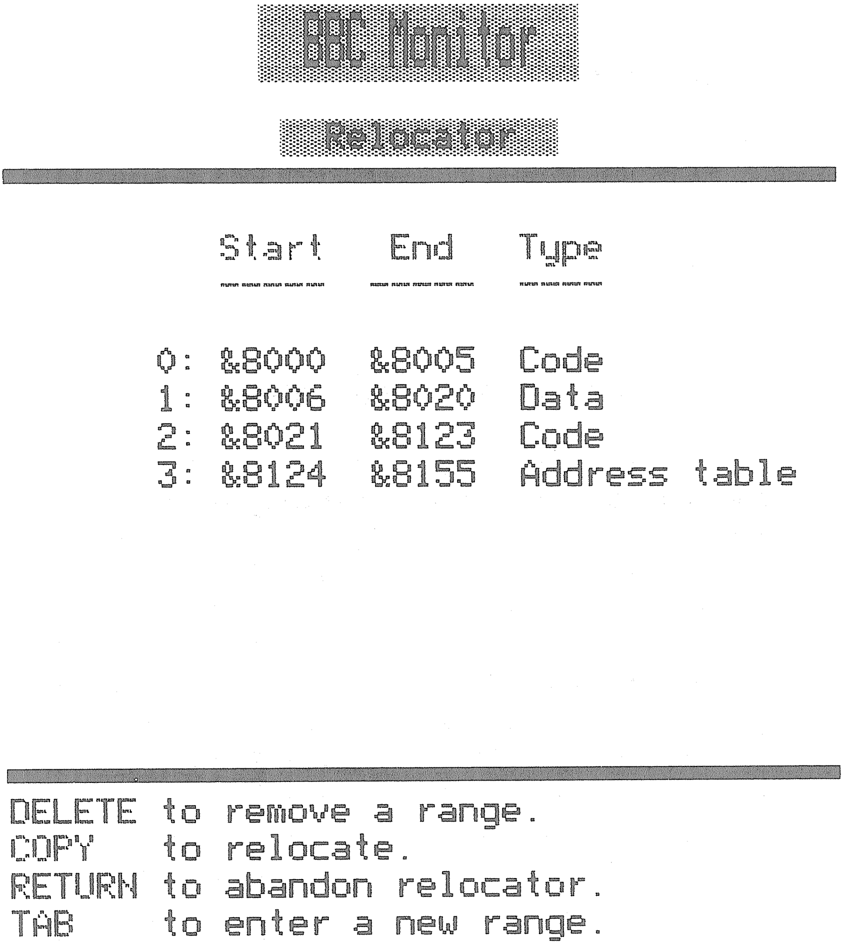
When the relocator is entered, you are asked to set the start and end addresses of the code to be relocated and where you wish it to relocate to.

You are then given the opportunity to define areas as code (i.e. normal machine code), data (character strings etc. which should not be altered), or address table (i.e. two byte addresses in Lo-Hi format which may need to be amended).

When all the desired areas have been defined, press **COPY** and the code will be relocated. You will then be returned to the command stage and from here the code can be saved or you can return to Monitor.

Relocate (SHIFT-f8 – HIMON only)

This option in HIMON functions in the same way as the relocate code **(f0)** option in RELOC.

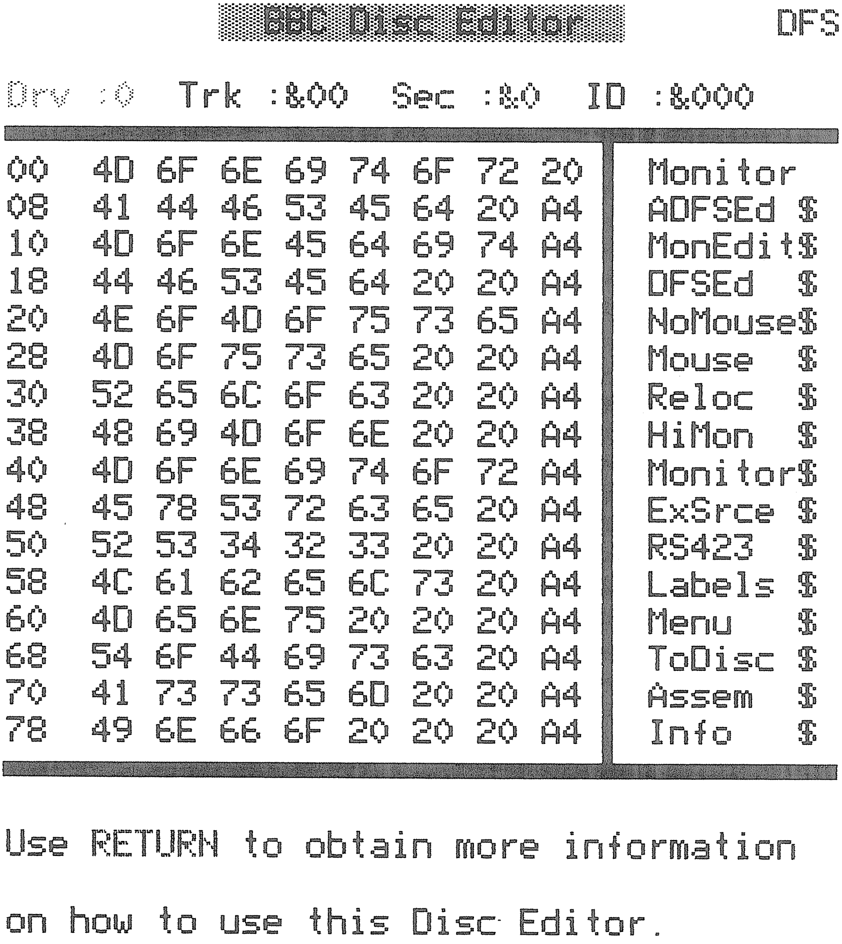


```
                 BBC Monitor

                  Relocator

         Start     End      Type
         -----     ---      ----

     0:  &8000    &8005    Code
     1:  &8006    &8020    Data
     2:  &8021    &8123    Code
     3:  &8124    &8155    Address table


DELETE to remove a range.
COPY   to relocate.
RETURN to abandon relocator.
TAB    to enter a new range.
```

8 THE DISC EDITOR

Introduction

This program is designed to give you access to the surface of the disc. Probably the most important reason for being able to do this is to recover DELETE'd programs or to try and rescue the contents of a damaged disc by transferring the undamaged sectors on to a new disc.



```
              BBC Disc Editor              DFS

Drv :0   Trk :&00   Sec :&0   ID :&000

00   4D 6F 6E 69 74 6F 72 20   Monitor
08   41 44 46 53 45 64 20 A4   ADFSEd $
10   4D 6F 6E 45 64 69 74 A4   MonEdit$
18   44 46 53 45 64 20 20 A4   DFSEd  $
20   4E 6F 4D 6F 75 73 65 A4   NoMouse$
28   4D 6F 75 73 65 20 20 A4   Mouse  $
30   52 65 6C 6F 63 20 20 A4   Reloc  $
38   48 69 4D 6F 6E 20 20 A4   HiMon  $
40   4D 6F 6E 69 74 6F 72 A4   Monitor$
48   45 78 53 72 63 65 20 A4   ExSrce $
50   52 53 34 32 33 20 20 A4   RS423  $
58   4C 61 62 65 6C 73 20 A4   Labels $
60   4D 65 6E 75 20 20 20 A4   Menu   $
68   54 6F 44 69 73 63 20 A4   ToDisc $
70   41 73 73 65 6D 20 20 A4   Assem  $
78   49 6E 66 6F 20 20 20 A4   Info   $

Use RETURN to obtain more information

on how to use this Disc Editor.
```

Both these facilities are provided by the Disc Editor. It also provides the facility to look through a disc, find a sequence of bytes anywhere on the disc, resave a sector, alter the contents of a sector, display the catalogue and display the file information.

The options are controlled by the function keys at the top of the keyboard. A strip is included with this package which can be inserted under, or simply placed on, the plastic strip above the red keys to remind you of their function.

The position on the disc that you wish to look at can be specified in two ways. You can enter the drive number, track number and sector number. Alternatively, you can just enter the sector ID number. This is a combination of the track and sector number (actually, it is the track number times 10 (17 in ADFS) plus the sector number); this is the form the computer uses in the right hand column of the *INFO command display. To change the drive that the editor is looking at, use the long form of entry **(f1)**.

Commands

Commands marked by (*) are not available in ADFSEd.

f0 allows you to enter the sector number of the sector you wish to look at. This number is the one displayed after each file in the INFO display, and is in fact the track number times 10 plus the sector number. (DFS) or track times 17 plus sector number (ADFS).

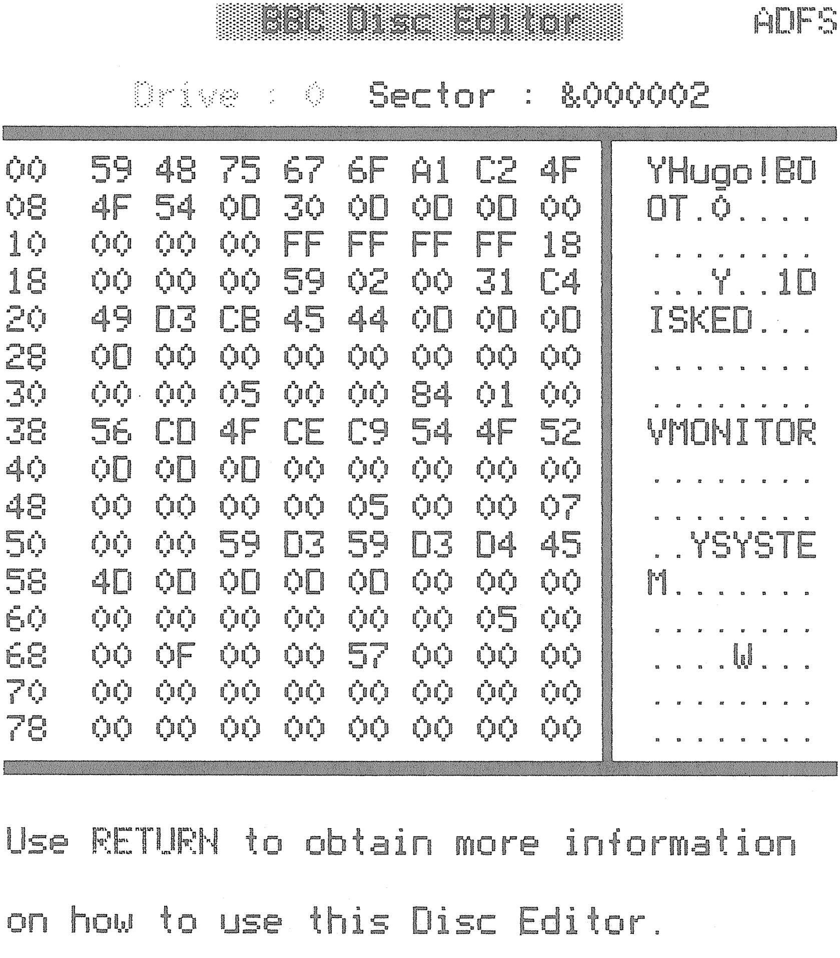
f1 allows you to enter new values for the three parameters Drive number, Track number and Sector number. The drive number can range from &0 to &3 (&0 and &1 in ADFS). The track number can be between &0 and &27 (&4F for an 80 track drive). The sector number is always between &0 and &9 (combined in ADFS).

Pressing **SHIFT f0** or **f1** will allow you to alter the position on the disc without updating the contents of the screen. In this way, sectors can be moved from one area to another, or from one disc to another. If the present screen contents did not originate from the sector and track displayed at the top of the screen, a red asterisk will flash beside the sector ID. This facility should be used with extreme caution!

f2 saves the sector currently displayed on to the disc at the position shown at the top of the display. If you wish, you may insert a blank formatted disc into the drive before saving the sector. This is very useful in recovering the contents of a damaged disc, by copying all the undamaged sectors on to a new disc. Before actually writing to the disc, the editor asks ''Write to disc – are you sure?'' The function will only be carried out if you answer ''Y'' to this question.

SHIFT f2 (*) will display the ID fields of the current track. These give such information as sector size and addresses. For further information about disc ID marks, you should refer to a publication specifically aimed at the subject, such as 'Mastering the Disc Drive' from BBC Publications.

f3 will find a sequence of bytes on the disc (if they exist) and then move the cursor to the beginning of them. The method of entering the FIND sequence may at first seem complicated but is designed to allow you to



```
           BBC Disc Editor            ADFS

        Drive : 0   Sector : &000002

00  59 48 75 67 6F A1 C2 4F   YHugo!BO
08  4F 54 0D 30 0D 0D 0D 00   OT.0....
10  00 00 00 FF FF FF FF 18   ........
18  00 00 00 59 02 00 31 C4   ...Y..1D
20  49 D3 CB 45 44 0D 0D 0D   ISKED...
28  0D 00 00 00 00 00 00 00   ........
30  00 00 05 00 00 84 01 00   ........
38  56 CD 4F CE C9 54 4F 52   VMONITOR
40  0D 0D 0D 00 00 00 00 00   ........
48  00 00 00 00 05 00 00 07   ........
50  00 00 59 D3 59 D3 D4 45   ..YSYSTE
58  4D 0D 0D 0D 0D 00 00 00   M.......
60  00 00 00 00 00 00 05 00   ........
68  00 0F 00 00 57 00 00 00   ....W...
70  00 00 00 00 00 00 00 00   ........
78  00 00 00 00 00 00 00 00   ........

Use RETURN to obtain more information

on how to use this Disc Editor.
```

search for both control codes and ASCII strings without having to work out the ASCII for each character in the sequence. When you have entered the sequence you will be asked for start and end tracks and sectors. This allows you to select a particular section of the disc to be searched. This can speed things up tremendously if you know roughly where the sequence is (e.g. you know that the sequence is in a program stored at Track 5 Sector 3 onwards). This function can take quite a while to find the sequence, so have patience. If it does not find it, the cursor will go to the top left hand corner of the sector after the last sector checked.

Note that '#' is used as a wild card character in the same way as in Monitor.

SHIFT-f3 will find the next occurrence of the sequence specified by FIND **(f3).**

f4 and **f5** move you backwards and forwards one sector respectively. The present drive number, track, sector and sector ID are shown at the top of the display. On the left of the screen a hex representation of the sector is presented in 16 rows of eight columns. On the right is an ASCII dump of the sector (also 16 × 8). Any non-printable characters are shown as a full stop (.) and any character with a value greater than 128 is shown as the ASCII for that value minus 128. The cursor can be moved around the hex section or ASCII sections (depending on what the present input mode is) using the four cursor control keys which are located at the top right of the keyboard. To alter a byte or sequence of bytes, move the cursor to the beginning of the first byte to be altered and start typing in hex (or ASCII if you are in the ASCII input mode). For more information on how to enter bytes, see the section under the heading **ESCAPE.**

f6 will print out the current sector on the printer (if attached).

f7 displays the full information about each file on the disc. As well as the title, directory and access state, it also displays the load address, execution address, length in bytes and the first sector at which the file is stored. This information is helpful in locating a file on the surface of the disc.

f8 (*) will help you recover a DELETE'd file or program. Before using it you will have to do some detective work to find out how long the file is and whereabouts on the disc it is stored. You should use the INFO **(f7)** and FIND **(f3)** options to help you do this. Using INFO you can find out where the spaces not occupied by files are; e.g. if the sector ID jumps by 20 and the file stored there is only 1K long, then there is a good chance that there is another DELETE'd file in between. You may also remember approximately where the DELETE'd file appeared in the * INFO listing and concentrate your search around there. The FIND function can be of great help especially if the DELETE'd file had some title REMs or a distinctive piece of text within it. Also, any BASIC program that starts at line 10 (most do) will always commence on disc with the byte sequence &OD:&OO:&OA. The FIND command should be able to locate these and you have then the beginning of the file. Once you have done this, write down the sector ID.

You will then have to find the length of the file. This involves stepping forward using the **f5** option until you find the end of the file or, more likely, the beginning of another. You can then work out how long it is by subtracting the end sector number from the start number, adding 1 and then multiplying the result by 256. If you are in doubt, always make the length longer as this will do no harm.

When you have both the start sector and length, you can use the **f8** option. First you will be asked for the start sector, the length, the start address and the execution address.

When you have entered the sector number and length the Editor will create an entry in the catalogue called R.RECOVER. If all has gone well, this should be your *delete*'d program, in which case you should * RENAME *it immediately.* If not, simply delete it and try again. This will remove the entry from the catalogue and will not alter any other part of the disc. It may be that you did not get the length long enough or you found an old copy of the same program instead of the most recent copy.

Note: This method of recovery will only work if the disc has not been written to after the accidental deletion. If it has, it will do no harm to try and recover it, but your chances are slim!

f9 allows you to leave the Editor. The message ''Exit to BASIC – are you sure?'' is displayed. Answering ''N'' to this will leave you where you are.

ESCAPE causes the editor to switch input mode between hex input and ASCII input. You can tell which mode you are in by looking at the position of the cursor. If it is in the hex display section of the screen, then the Editor is in hex input mode, otherwise it is in ASCII input mode. The default mode is hex and this is set after disc operations etc.

The hex mode can be used to enter any character (0 to 255) into a location on the display. The character is entered as a 2-digit hex number, e.g. carriage return (ASCII 13) is entered as **0** (zero) followed by **D**. Notice that leading zeros must be entered. The Editor will not allow you to carry on until you have finished typing in a complete 2-digit number.

The ASCII mode can be used to enter characters from ASCII 32 to ASCII 126. When a character is pressed at the keyboard, the Editor checks to see if it is a legal character, and if so, prints it at the present cursor position and alters the hex representation as well.

The DFS disc catalogue

The catalogue is stored on track 0 sectors 0 and 1. The information held for each file consists of:

Filename	7 bytes
Directory	1 byte
Load Address	18 bits (2 high 8 middle 8 low)
Exec Address	18 bits (2 high 8 middle 8 low)
Length in bytes	18 bits (2 high 8 middle 8 low)
Start sector of file on disc	10 bits (2 high 8 low)

The information for the nth file entry is held in bytes 8 * n + &08 to 8 * n + &0F on sectors 0 and 1 as follows:

Byte	Sector 0	Sector 1
8 * n + &08 + &09	Filename Filename	Load Address middle bits Load Address low bits
+ &0A + &0B	Filename Filename	Exec Address middle bits Exec Address low bits
+ &0C + &0D	Filename Filename	Length middle bits Length low bits
+ &0E	Filename	high order bits 76 54 32 10 Exec Length Load Start
+ &0F	Directory	Start sector low bits

The first eight bytes in sector 0 contain the first eight bytes of the disc title.

The first eight bytes in sector 1 contain the following information:

&00 to &03	Last 5 bytes of the disc title
&04	The number of catalogue entries (multiplied by eight)
&05	The number of times that the disc has been written to (in binary coded decimal) since formatting.
&06 (bits 0,1)	Number of sectors on disc (2 high order bits of a 10 bit number)
(bits 4,5)	!BOOTstart-up option
&07	Number of sectors on disc (8 low order bits of a 10 bit number)

Note: The above information only applies to discs created for use with the DFS. Discs created for the Econet large file server or ADFS have a different format.

APPENDIX A

Monitor memory map in I/O processor

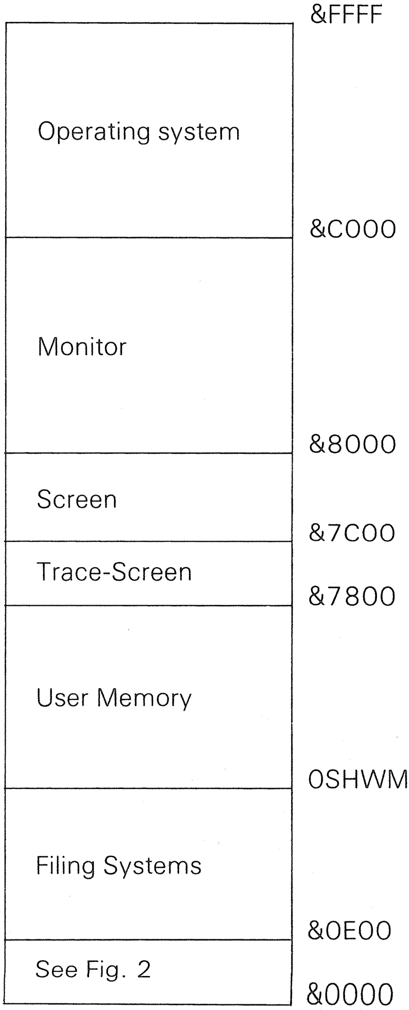


Fig. 1

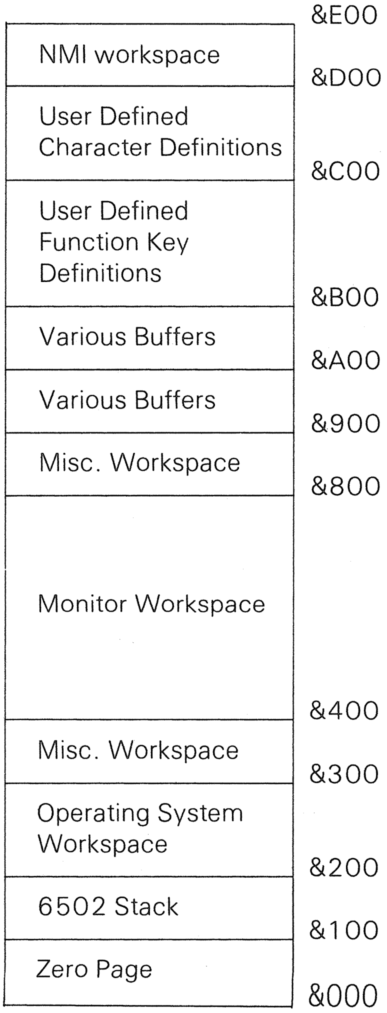


Fig. 2

Note zero page locations &70 – &8F are reserved for the user.

Monitor memory map in 6502 Second Processor

HIMON resides at &B000 to &F800 with Trace-Screen at &AC00 to &B000

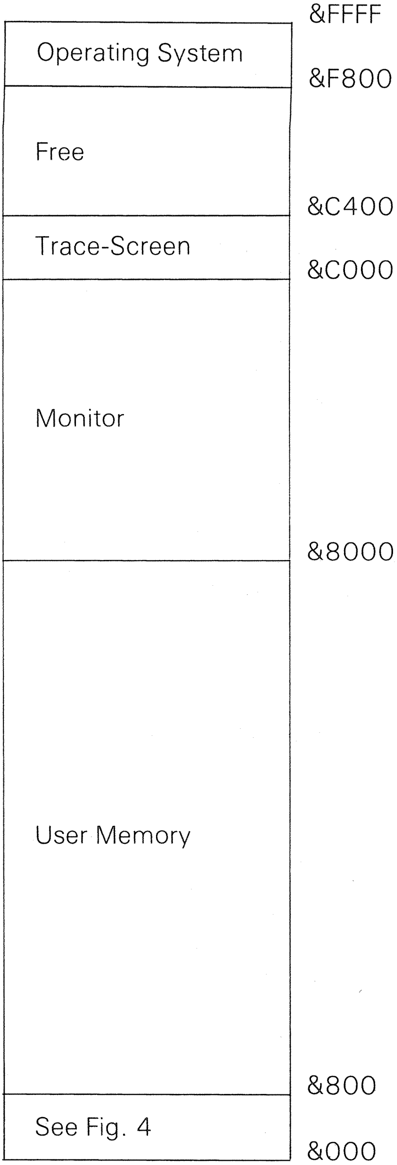


Fig. 3

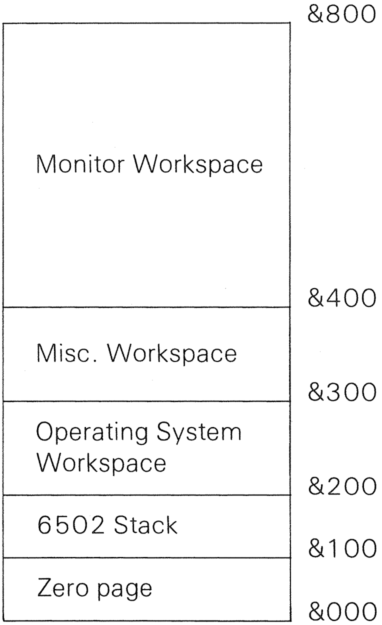


Fig. 4

APPENDIX B

List of Built-in Labels

Vectors

&200	USERV
&202	BRKV
&204	IRQ1V
&206	IRQ2V
&208	CLIV
&20A	BYTEV
&20C	WORDV
&20E	WRCHV
&210	RDCHV
&212	FILEV
&214	ARGSV
&216	BGETV
&218	BPUTV
&21A	GBPBV
&21C	FINDV
&21E	FSCV *
&220	EVNTV
&222	UPTV *
&224	NETV *
&226	VDUV *
&228	KEYV *
&22A	INSV *
&22C	REMV *
&22E	CNPV *
&230	IND1V *
&232	IND2V *
&234	IND3V *

OS Calls

&FFB3	OSWRSC *
&FFB9	OSRDSC *
&FFB9	OSRDRM *
&FFBC	OSVDU *
&FFBF	OSEVEN *
&FFC2	GSINIT *
&FFC5	GSREAD *
&FFC8	NVRDCH
&FFCB	NVWRCH
&FFCE	OSFIND
&FFD1	OSGBPB
&FFD4	OSBPUT
&FFD7	OSBGET
&FFDA	OSARGS
&FFDD	OSFILE
&FFE0	OSRDCH
&FFE3	OSASCI
&FFE7	OSNEWL
&FFEE	OSWRCH
&FFF1	OSWORD
&FFF4	OSBYTE
&FFF7	OSCLI

6502 Vectors

&FFFA	NMI
&FFFC	RESET
&FFEE	INTRPT

Labels marked with a (*) do not appear when the Second Processor memory is being examined.

APPENDIX C

File Formats for the Labels System

The Labels System of Monitor allows label tables to be loaded and saved on disc or tape. The files are stored in the following format:

For each of the labels there are 2 bytes for the address (in Lo-Hi format) and eight bytes for the label (padded with spaces where necessary). Therefore, the label '&203F Loop2' would be stored as:

3F 20 4C 6F 6F 70 32 20 20 20 (in hex)

If the first byte of the label string (i.e. the first byte after the address) is zero then that label is not valid. It has either been deleted or has not yet been used. The built-in labels are not saved when a label table is saved.

The program ''Labels'' allows you to convert a label file (see above) into an ordinary ASCII file which can then be edited using a wordprocessor or editor. It also allows ASCII files to be converted into label files. The format used for the ASCII files is as follows:

Each label is separated from the next by a carriage return (Code 13) and is stored as a sequence of ASCII codes. This format is designed for use with editors etc., and means that the BASIC commands INPUT# and PRINT# cannot be used with an ASCII file. The sequence is

&Address Label <CR>

Note the ampersand (&) before the address (which must be in hex) and the space between the address and the label. Each label must be on a separate line. <CR> stands for carriage return. A typical ASCII file might look like this:

```
&3000  Start
&3005  MainLoop
&3010  Enter
&301F  Checkit
&302C  OK
&3030  End
```

Remember that labels must start with a letter and cannot be more than 8 characters long. You may read or create an ASCII file from your own program, by means of the commands BGET# and BPUT#, or by using MonEdit.

APPENDIX D

Operating System calls

The following call is implemented in Monitor and allows the I/O Processor memory to be read from or written to independently of which processor is being used (i.e. I/O or Second Processor). It differs from OSWORD 5 & 6 in that it uses only 16 bit addressing and also allows any of the 16 possible sideways ROMs to be selected for reading.

Read/Write I/O processor memory

OSWORD with A= &5F
The call address is &FFF1

On entry X (lo-byte) and Y (hi-byte) point to the parameter block:

Offset		
0		Command
1	Lo	Address to read
2	Hi	from / write to
3		—
4		—
5		Data or result
6		ROM to be selected or 128 for shadow memory

For a write operation the command byte should be set to &01 and the byte to be written placed in XY + 5.

For a read operation the command byte should be set to &00. The result will be returned in XY + 5.

APPENDIX E

65C02 Instruction Set

Monitor is designed to be used with either a 6502 processor or the new 65C02 processor which is fitted to the Second Processor.

The 65C02 has several new mnemonics and two new addressing modes. These are explained on the following few pages.

MSD \ LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	BRK Implied 1 7	ORA (IND, X) 2 6			TSB ZP 2 5	ORA ZP 2 3	ASL ZP 2 5	RMB0 ZP 2 5	PHP Implied 1 3	ORA IMM 2 2	ASL Accum 1 2		TSB ABS 3 6	ORA ABS 3 4	ASL ABS 3 6	BBR0 ZP 3 5**	0
1	BPL Relative 2 2**	ORA (IND), Y 2 5*	ORA (IND) 2 5		TRB ZP 2 5	ORA ZP, X 2 4	ASL ZP, X 2 6	RMB1 ZP 2 5	CLC Implied 1 2	ORA ABS, Y 3 4*	INC Accum 1 2		TRB ABS 3 6	ORA ABS, X 3 4*	ASL ABS, X 3 7	BBR1 ZP 3 5**	1
2	JSR ABS 3 6	AND (IND, X) 2 6			BIT ZP 2 3	AND ZP 2 3	ROL ZP 2 5	RMB2 ZP 2 5	PLP Implied 1 4	AND IMM 2 2	ROL Accum 1 2		BIT ABS 3 4	AND ABS 3 4	ROL ABS 3 6	BBR2 ZP 3 5**	2
3	BMI Relative 2 2**	AND (IND), Y 2 5*	AND (IND) 2 5		BIT ZP, X 2 4	AND ZP, X 2 4	ROL ZP, X 2 6	RMB3 ZP 2 5	SEC Implied 1 2	AND ABS, Y 3 4*	DEC Accum 1 2		BIT ABS, X 3 4*	AND ABS, X 3 4*	ROL ABS, X 3 7	BBR3 ZP 3 5**	3
4	RTI Implied 1 6	EOR (IND, X) 2 6				EOR ZP 2 3	LSR ZP 2 5	RMB4 ZP 2 5	PHA Implied 1 3	EOR IMM 2 2	LSR Accum 1 2		JMP ABS 3 3	EOR ABS 3 4	LSR ABS 3 6	BBR4 ZP 3 5**	4
5	BVC Relative 2 2**	EOR (IND), Y 2 5*	EOR (IND) 2 5			EOR ZP, X 2 4	LSR ZP, X 2 6	RMB5 ZP 2 5	CLI Implied 1 2	EOR ABS, Y 3 4*	PHY Implied 1 3			EOR ABS, X 3 4*	LSR ABS, X 3 7	BBR5 ZP 3 5**	5
6	RTS Implied 1 6	ADC (IND, X) 2 6†			STZ ZP 2 3	ADC ZP 2 3†	ROR ZP 2 5	RMB6 ZP 2 5	PLA Implied 1 4	ADC IMM 2 2†	ROR Accum 1 2		JMP (ABS) 3 6	ADC ABS 3 4†	ROR ABS 3 6	BBR6 ZP 3 5**	6
7	BVS Relative 2 2**	ADC (IND), Y 2 5*†	ADC (IND) 2 5†		STZ ZP, X 2 4	ADC ZP, X 2 4†	ROR ZP, X 2 6	RMB7 ZP 2 5	SEI Implied 1 2	ADC ABS, Y 3 4*†	PLY Implied 1 4		JMP (ABS, X) 3 6	ADC ABS, X 3 4*†	ROR ABS, X 3 7	BBR7 ZP 3 5**	7
8	BRA Relative 2 3*	STA (IND, X) 2 6			STY ZP 2 3	STA ZP 2 3	STX ZP 2 3	SMB0 ZP 2 5	DEY Implied 1 2	BIT IMM 2 2	TXA Implied 1 2		STY ABS 3 4	STA ABS 3 4	STX ABS 3 4	BBS0 ZP 3 5**	8
9	BCC Relative 2 2**	STA (IND), Y 2 6	STA (IND) 2 5		STY ZP, X 2 4	STA ZP, X 2 4	STX ZP, Y 2 4	SMB1 ZP 2 5	TYA Implied 1 2	STA ABS, Y 3 5	TXS Implied 1 2		STZ ABS 3 4	STA ABS, X 3 5	STZ ABS, X 3 5	BBS1 ZP 3 5**	9
A	LDY IMM 2 2	LDA (IND, X) 2 6	LDX IMM 2 2		LDY ZP 2 3	LDA ZP 2 3	LDX ZP 2 3	SMB2 ZP 2 5	TAY Implied 1 2	LDA IMM 2 2	TAX Implied 1 2		LDY ABS 3 4	LDA ABS 3 4	LDX ABS 3 4	BBS2 ZP 3 5**	A
B	BCS Relative 2 2**	LDA (IND), Y 2 5*	LDA (IND) 2 5		LDY ZP, X 2 4	LDA ZP, X 2 4	LDX ZP, Y 2 4	SMB3 ZP 2 5	CLV Implied 1 2	LDA ABS, Y 3 4*	TSX Implied 1 2		LDY ABS, X 3 4*	LDA ABS, X 3 4*	LDX ABS, Y 3 4*	BBS3 ZP 3 5**	B
C	CPY IMM 2 2	CMP (IND, X) 2 6			CPY ZP 2 3	CMP ZP 2 3	DEC ZP 2 5	SMB4 ZP 2 5	INY Implied 1 2	CMP IMM 2 2	DEX Implied 1 2		CPY ABS 3 4	CMP ABS 3 4	DEC ABS 3 6	BBS4 ZP 3 5**	C
D	BNE Relative 2 2**	CMP (IND), Y 2 5*	CMP (IND) 2 5			CMP ZP, X 2 4	DEC ZP, X 2 6	SMB5 ZP 2 5	CLD Implied 1 2	CMP ABS, Y 3 4*	PHX Implied 1 3			CMP ABS, X 3 4*	DEC ABS, X 3 7	BBS5 ZP 3 5**	D
E	CPX IMM 2 2	SBC (IND, X) 2 6†			CPX ZP 2 3	SBC ZP 2 3†	INC ZP 2 5	SMB6 ZP 2 5	INX Implied 1 2	SBC IMM 2 2†	NOP Implied 1 2		CPX ABS 3 4	SBC ABS 3 4†	INC ABS 3 6	BBS6 ZP 3 5**	E
F	BEQ Relative 2 2**	SBC (IND), Y 2 5*†	SBC (IND) 2 5†			SBC ZP, X 2 4†	INC ZP, X 2 6	SMB7 ZP 2 5	SED Implied 1 2	SBC ABS, Y 3 4*†	PLX Implied 1 4			SBC ABS, X 3 4*†	INC ABS, X 3 7	BBS7 ZP 3 5**	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

□ — New Opcode

	0	
0	BRK Implied 1 7	—OP Code —Addressing Mode —Instruction Bytes; Machine Cycles

†Add 1 to N if in decimal mode.
*Add 1 to N if page boundary is crossed.
**Add 1 to N if branch occurs to same page;
Add 2 to N if branch occurs to different page.

The columns shaded are instructions which are available on some 65C02 processors, but not that one fitted to the BBC Micro Second Processor.

Instruction Set Alphabetic Sequence

NOTES:

† New Instruction

* Previous Instruction with additional addressing mode(s)

A =Accumulator X,Y=Index registers

Mnemonic	Function
*ADC	Add Memory to A with Carry
*AND	AND Memory with A
ASL	Shift Left One Bit (Memory or A)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Zero
*BIT	Test Bits in Memory with A
BMI	Branch on Result Minus
BNE	Branch on Result not Zero
BPL	Branch on Result Plus
†BRA	Branch Always
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
*CMP	Compare Memory and A
CPX	Compare Memory and X
CPY	Compare Memory and Y
*DEC	Decrement Memory by One
DEX	Decrement X by One
DEY	Decrement Y by One
*EOR	Exclusive-OR Memory with A
*INC	Increment Memory by One
INX	Increment X by One
INY	Increment Y by One
*JMP	Jump to New Location

Mnemonic	Function
JSR	Jump Saving Return Address
*LDA	Load A with Memory
LDX	Load X with Memory
LDY	Load Y with Memory
LSR	Shift One Bit Right (Memory or A)
NOP	No Operation
*ORA	OR Memory with A
PHA	Push A on Stack
PHP	Push Processor Status on Stack
† PHX	Push X on Stack
† PHY	Push Y on Stack
PLA	Pull Accumulator from Stack
PLP	Pull Processor Status from Stack
† PLX	Pull X from Stack
† PLY	Pull Y from Stack
ROL	Rotate One Bit Left (Memory or A)
ROR	Rotate One Bit Right (Memory or A)
RTI	Return from Interrupt
RTS	Return from Subroutine
SBC	Subtract Memory from A with Borrow
SEC	Set Carry Flag
SED	Set Decimal Mode
SEI	Set Interrupt Disable Status
*STA	Store A in Memory
STX	Store X in Memory
STY	Store Y in Memory
† STZ	Store Zero
TAX	Transfer A to X
TAY	Transfer A to Y
† TRB	Test and Reset Bits
† TSB	Test and Set Bits
TSX	Transfer Stack Pointer to X
TXA	Transfer X to A
TXS	Transfer X to Stack Register
TYA	Transfer Y to A

APPENDIX F—A summary of the Monitor commands

RETURN	Cycle through the instructions and options.
f0	Move to a given address.
f1	Alter the 6502 registers.
f2	Enter the 6502 assembler.
f3	Find a sequence of bytes.
f4	Trace a single instruction at the location held in PC.
f5	JSR to the location held in PC.
f6	Dump or disassemble memory to the printer.
f7	Move a section of memory.
f8	Breakpoints.
f9	Exit Monitor and enter BASIC.
SHIFT f0	Select a paged ROM.
SHIFT f1	Fill a section of memory with a specified byte.
SHIFT f2	Enter the Label System.
SHIFT f3	Find the next occurrence of the bytes set by **f3.**
SHIFT f4	Set the options.
SHIFT f5	JMP to the Program Counter.
SHIFT f6	Perform a calculation.
SHIFT f7	Disc Assembler (HIMON only).
SHIFT f8	Relocator (HIMON only).
*	Accept and execute a * command.
TAB	Switch between Memory Display and Disassembly modes.
ESCAPE	Switch between input in hex and input in ASCII (Memory Display mode only).
COPY	Memory Display mode: display the binary value at the cursor position. During numerical input : pressing **COPY** followed by **RETURN** will enter the current cursor position.
CTRL-L	Clear tracescreen.
CURSOR KEYS	Move the cursor. In Memory Display mode: The keys move the cursor around the screen in the normal manner. **SHIFT CURSOR UP** or **DOWN** will move the cursor up or down a screen at a time.

In Disassembly mode:
The **CURSOR DOWN** key will disassemble the next instruction. The other cursor keys have no effect.

CURSOR UP will move the 'cursor' up eight bytes and start disassembly from that point.

SHIFT CURSOR DOWN will start disassembly from the last instruction on the screen.

RETURN When input is requested by one of the Monitor functions, pressing **RETURN** on its own will return you to Memory Display or Disassembly mode.

APPENDIX G—a summary of the Disc Editor commands

f0	Specify a new sector ID.
SHIFT-f0	Specify a new sector ID, but do not load sector.
f1	Specify new values for drive, track and sector.
SHIFT-f1	Specify new values for drive etc. but do not load sector.
f2	Save the current sector on disc.
SHIFT-f2 *	Display the sector IDs of the current track.
f3	Find a sequence of bytes on the disc.
SHIFT-f3	Find the next occurrence.
f4	Move backwards one sector.
f5	Move forward one sector.
f6	Print out the current sector.
f7	Display the disc information.
f8 *	Recover a file.
f9	Exit the Editor.
ESCAPE	Switches between the HEX and ASCII input modes.
RETURN	Cycles the information at the bottom of the screen.
*	Perform a * command.
COPY	Display byte at cursor position in binary.

* – not available in ADFSEd.