Picturesque

Spectrum MONITOR

Machine Code
Debug &
Disassembler

profisoft

OPERATING INSTRUCTIONS FOR

Spectrum MONITOR

A machine code entry & de-bug/monitor routine

for the Sinclair ZX SPECTRUM 16K & 48K

Published by: Picturesque,
6 Corkscrew Hill,
West Wickham,
Kent,
BR4 9BB.

C O N T E N T S

INTRODUCTION

The SPECTRUM MONITOR is a machine code entry and de-bug monitor utility, with comprehensive facilities including a Disassembler, that has been designed to allow the ZX Spectrum computer to be programmed in machine code, without the need to use any Basic commands.

The MONITOR allows a free interchange with Basic, and has been carefully structured so that machine code routines can be run from either Basic or from the MONITOR, and that the MONITOR can be accessed at any time without upsetting the stack. All keyboard entries are validated, and it is impossible to crash the Spectrum while using the MONITOR commands.

The program cassette that accompanies this book contains two versions of the MONITOR; one for the 16K Spectrum and one for the 48K Spectrum. If you own a 48K machine, you can use either version.

THE SPECTRUM MONITOR occupies just over 4K of memory at the top of memory and is not relocatable. On loading into your Spectrum, Ramtop is automatically reset to below the MONITOR, and should only be altered downwards. In other words, assuming you use the correct version for the memory size your Spectrum contains, you can lower Ramtop further to create space for your own machine code programs if you wish, but the MONITOR uses the 4K of memory immediately below the start of the User Definable Graphics area of RAM. If you load the 16K version into a 48K Spectrum, you can use the memory above the MONITOR but if you move Ramtop above the MONITOR, you may run the risk of corrupting the MONITOR routines with Basic commands.

LOADING THE SPECTRUM MONITOR

Connect your Spectrum to a T.V., cassette recorder, and ZX printer if you own one, as described in the Sinclair manual, and switch on.

The MONITOR loads from cassette in the same way as a Basic program. Set the volume control to about ½ volume and set the tone control to maximum treble.

Type LOAD "Monitor 16" or LOAD "Monitor 48" according to which side of the cassette you are loading, or simply type LOAD "". Start the cassette in play and press ENTER. As with all commercially recorded cassettes, you may find that you have to experiment until you find the optimum replay level for this tape. Make a note of the volume setting you find successful, for future use. The chances of a bad load are reduced if you observe the following:

1) Regularly clean the record/replay head of your cassette recorder, using one of the proprietory cleaning kits.
2) Clean the rubber pinch wheel and the capstan spindle that it makes contact with.
3) Use a cleaning kit supplied with a liquid cleaner that you apply with a cotton wool swab. This type is far more effective than the cleaning cassette type.
4) Disconnect the 'Mic' lead from the cassette recorder when loading.

If you should experience difficulty in loading your copy of the MONITOR, or if you accidentally damage your tape, please return the cassette to Picturesque (the address is on the front page of this book, and on the cassette inlay card). Your cassette will be re-recorded directly from our ZX Spectrum, and sent back to you by return of post, along with postage stamps to cover your postage costs. We believe such a back-up service to be an essential part of our trading standards.

ACCESS TO SPECTRUM MONITOR

Having successfully loaded, the MONITOR automatically relocates itself to the correct part of memory, resets Ramtop to below itself, and displays a message to this effect on the screen. This screen message shows the correct form of address to access the MONITOR, depending on which version is loaded.

```
RANDOMIZE USR 30479   (for 16K)
RANDOMIZE USR 63247   (for 48K)
```

You can access the MONITOR from Basic in this way at any time, which will produce the following message at the bottom of the screen, in addition to whatever was already there:

```
Press BREAK for Monitor
```

On pressing the Break key (shifted or unshifted) the screen is cleared, and the prompt and cursor appear at the bottom of the screen. The MONITOR uses its own internal stack to make its use transparent to your machine code program. These functions will be explained more fully later.

PROMPT & CURSOR

The prompt ($\boxed{>}$) indicates that the MONITOR is waiting to be put into a command mode. It does not appear at the start of every line of the display, but only appears when a command routine has ended, and the MONITOR is waiting for a new command instruction. The flashing cursor is visible for the majority of the time, and indicates a request for a keyboard entry, and shows where the result of that keyboard entry will be displayed on the screen.

COMMAND MODES

The range of commands offered by the SPECTRUM MONITOR are as follows:

M	Display a memory location and its contents, and change its contents.
X	Escape from a command mode to the start of the MONITOR, (Operates on all command modes except R and K.)
A	Move an area of RAM to a new location.
F	Fill a specified area of RAM with a specified byte value.
I	Insert up to 255 bytes into a machine code routine.
D	Delete up to 255 bytes from a machine code routine.

J	Jump to specified address, and start executing the routine there.
B	Set a Breakpoint in a machine code routine, to return control to the MONITOR.
K	Restore codes after a Breakpoint has been passed.
R	Display the contents of the CPU registers.
C	Continue operation of a routine after a Breakpoint.
Y	Return to Basic.
P	Hex dump to Printer.
$	Text entry.
Z	Disassemble any area of memory into Z80 mnemonics either to the screen, or to both the screen and the ZX Printer.
N	Number conversion. Hex to Decimal or Decimal to Hex.

All command codes are accessed by a single keystroke denoted by the letter in the left hand column above. All keyboard entries are checked for validity, and at any given time, only the permitted keyboard entries are accepted by the MONITOR; any other key press is rejected and the keyboard is scanned again.

All numeric inputs and displays are in Hex to facilitate the entry of the Z80 Op. Codes. The number conversion routine simplifies access to Hex addresses.

PLEASE NOTE

All references to addresses and their contents in these instructions will be in Hex, and will be shown as a two or four figure number, without the suffix 'H'.

Each command will now be dealt with in detail, with examples, to clarify its operation.

M - Display contents of memory location

The screen should still only show the prompt and cursor in the bottom left corner.

Type M

An inverse M (blue letter on a white square) will appear immediately to the right of the prompt, and the cursor will move along the bottom line by one character space.

Now type in the Hex value of the address that you wish to inspect, say 6000. (You can use the MONITOR to inspect or change any memory location in RAM or ROM, although you cannot alter the ROM values.) The address appears on the screen as you type it in, and as soon as you have typed the fourth digit, a two character Hex number appears on the screen to the right of the address, showing the Hex value of the contents of that memory location.

```
■■6000 00 -
```

The cursor has now moved on, leaving a space after the two contents digits.
Now type FF.

```
■■6000 00 FF -
```

This has loaded the Hex value FF into memory location 6000. The value is loaded into the location automatically after you type the second digit.
Now type ENTER.

```
■■6000 00 FF
6001 00 -
```

The original line with the prompt has scrolled up one line, and 6001 00 is now displayed on the bottom line. At all times, the MONITOR operates with a scrolling screen, and new information is always displayed on the bottom line.

Typing ENTER displays the next memory location and its contents. To enter a value in this new address, enter the two Hex digits, or type ENTER again for the next memory location.

Now let's check that FF really has been loaded into address 6000.

Type M

```
MM6000 00 FF
6001 00
M-
```

The screen has scrolled up one line, and the inverse M has reappeared on the bottom line.

Typing M after a Hex address, data entry, or ENTER, allows you to re-enter the M command routine at the start. To re-enter the M command in the middle of typing a Hex address, type X, to escape, and M to enter the M command.

Now type 6000

```
MM6000 00 FF
6001 00
M6000 FF -
```

The contents of 6000 are shown as FF.

To summarize the M command so far, you can sequentially step through memory locations using ENTER and change the value of the contents of each address, or, by typing M again, you can define a new starting address. To re-enter the M command with a partly typed address on the screen, type X to escape to the prompt, and then type M to enter the M command.

Now type 00 to clear the contents of address 6000. You will notice that the cursor is still visible on the bottom line of the screen to the right of the 00 just entered. Although address 6000 now contains the value 00, (the value was changed immediately you typed the second 0) you can change it again if you wish.

So, if you enter an incorrect value, and realise what you have done before you press ENTER, you can correct your mistake without having to specify the address again. In fact, you can make only two attempts at entering a value before the routine returns you to the prompt and cursor, when you will have to type M to re-enter the M command.

Remember, that in the M command, ENTER only has the effect of stepping on to the next memory location, whereas in other commands (where applicable) ENTER causes the operation to be executed.

Imagine that you have just entered a machine code routine from, say 6000 to 6200, and you realise that you need to change the value of one byte somewhere near the middle of the routine, but you do not know the precise address. You could spend a long time guessing addresses and looking at their contents, or repeatedly pressing ENTER until you find the right byte. But if you type M, to re-enter the M command, and look at the contents of an address somewhere near the beginning of the routine, then press ENTER, and hold it pressed, after about 1½ seconds the screen will start scrolling quite fast, and will rapidly display successive locations until you release ENTER. In this way you can quickly scan through a routine until you find the byte you are looking for. To effect the alteration, having released ENTER, you will again have to type M xxxx to re-enter the M command at the correct address, and then change the contents of that address.

The 'M' command, (and the Ø command, described on page 21) are the only commands that use the repeating key facility.

X - Escape

The X command allows you to escape from a command mode and returns you to the monitor key scan routine.

Type X

The screen will scroll, and the prompt and cursor will reappear on the bottom line of the screen. You can now enter any of the MONITOR commands. All command routines, with the exception of R (display registers) and K (breakpoint restore), will accept X as an escape command at any time up to the point of execution.

I - Insert

If, having written a machine code routine, you find it necessary to add extra instructions in the middle of that routine, the Insert command (I) allows you to insert up to 255 bytes at any point, and automatically moves up memory a specified area of RAM by the number of bytes you wish to insert.

The Insert command takes the form 'I aaaa bbbb nn' where I is the Insert command mode, aaaa is the Hex address of the first byte of the insertion, bbbb is the Hex address of the highest byte in RAM of the block of memory to be moved, and nn is the number of bytes to be inserted, in Hex.

Example:
Type X to restore the prompt and cursor to the bottom line, and then, using the M command, enter the following consecutive values into memory:

```
6000 00
6001 01
6002 02
6003 03
  etc.
     .
     .
     .
     .
600A 0A
600B 0B
600C 0C
600D 0D
600E 0E
600F 0F
```

(The values entered into these locations are purely for a demonstration of the Insert and Delete commands, and, if run, will cause the ZX SPECTRUM to crash).

In the above example, the start of the imaginary routine is 6000 and the end is 600F. We will now insert 5 bytes, the first new byte to be at 6004.

Type X	to restore the prompt and cursor.
Type I	to get into the Insert mode.
Type 6004	the address of the first byte of the insertion.
Type 600F	the address of the highest byte to be moved.
Type 05	the number of bytes to be inserted (Hex).

At any time up to this point, you can type X to escape from this command mode, as no change to RAM has occurred yet. Indeed, if you make a typing error at any time, you must type X, and start the command again.

If you have entered the Insert example correctly, you can now type ENTER to effect the insertion. The screen will scroll up one line, and the prompt and cursor will return to the bottom line. The insertion has been completed.

Using the M command, check through the 21 locations from 6000. Addresses 6004 to 6008 inclusive will now contain the value 00, and 6009 to 6014 will contain the values 04 through to 0F.

When using the Insert command, any absolute addresses in the remainder of the routine that referred to the area that has been relocated, will have to be changed to maintain correct operation of the routine.

D - Delete

This command has the opposite effect to Insert, and takes the form 'D aaaa bbbb nn', where D is the Delete command mode, aaaa is the address of the first byte to be deleted, bbbb is the address of the highest byte to be moved down RAM, and nn is the number of bytes to be deleted.

Assuming that the result of the Insert example is still in memory, let us now move the area of RAM from 6009 to 6014 back to its original place.

Type X	to restore the prompt and cursor.
Type D	to enter the Delete command.
Type 6004	the start of the area to be deleted.
Type 6014	the end of the area to be moved.
Type 05	the number of bytes to be deleted (Hex)
Type ENTER	to effect the deletion.

The prompt and cursor will reappear on the bottom line, and the deletion will be complete.

Now check through addresses 6000 to 6014, using the M command. The contents of these locations will be as they were before the Insert example, and locations 6010 to 6014 will have been loaded with the value 00.

Again, any absolute addresses relating to the area of RAM that has been moved by Delete, will now need to be changed.

A - Area Relocate

The 'A' command block moves a specified area of RAM, and takes the form: 'A aaaa bbbb cccc' where A is the Area Relocate command, aaaa is the present start address, and bbbb is the present end address of the area to be moved, and cccc is the new starting address.

Assuming that the example used for the I and D commands is still in memory, let us now move the whole area from 6000 to 600F up memory, to start at 6200.

Type X	to restore the prompt and cursor.
Type A	to enter the Area Relocate mode.
Type 6000	the start address of the area to be moved.
Type 600F	the end address of the area to be moved.
Type 6200	the new start address.
Type ENTER	to effect the move.

The screen scrolls up one line, and the prompt and cursor return to the bottom line. The move is complete.

Using the M command, check that addresses 6200 to 620F have been loaded with the same values as those still remaining in addresses 6000 to 600F.

This routine will allow you to move up or down memory, from any original starting address to any new starting address, even if the new area overlaps the original area. The original area (unless over-written by the move) is not changed.

Try moving the area from 6000 to 600F to a new start address of 6008 and then move it back again.

A word of warning: Most MONITOR commands that allow you to alter the values in memory locations, will operate on the area of RAM containing the MONITOR routines. Always check carefully that you are not about to corrupt the MONITOR, which occupies the area from EEAC to FEF9, (48K); or 6EAC to 7EF9 (16K).

F - Fill an area with a given value

The Fill routine allows you to enter the same byte value into a given area of RAM, and takes the form: 'F aaaa bbbb xx', when aaaa and bbbb are the start and end addresses respectively of the specified area, and xx is the value to be entered.

Type X	to restore the prompt and cursor.
Type F	to enter the F command.
Type 6020	the start address.
Type 6100	the end address.
Type AA	the value to be entered. (Hex).
Type ENTER	to effect the Fill.

The screen scrolls, and the prompt and cursor appear on the bottom line of the screen. The fill is complete.

Now use the M command, with ENTER kept pressed, to verify that each byte from 6020 to 6100 inclusive has a value of AA.

Y - Return

Above the 'Y' key is printed the keyword RETURN, and by pressing this key, followed by ENTER, when the prompt and cursor alone are visible on the bottom line of the screen, a return to Basic is effected so that you can use any of the Basic commands. As the MONITOR does not have its own Save and Load commands, you will need the Return command to use the Basic Save and Load.

If, having returned to Basic, you wish to access the MONITOR again the following addresses should be used with the USR function, depending upon which version of the MONITOR you are using:

```
     16K version:  type RANDOMIZE USR 30479
     48K version:  type RANDOMIZE USR 63247
```

followed by ENTER.

Using the MONITOR 'Return' command, will reset the stack, but will not affect the Basic listing or the variables.

To demonstrate the remaining commands of the MONITOR, use the 'M' command to enter the following short program, starting at address 6000 Hex.

```
6000  01 00 00  LD   BC,0000  :  Clear BC
6003  11 00 00  LD   DE,0000  :  Clear DE
6006  21 00 00  LD   HL,0000  :  Clear HL
6009  03        INC  BC       :  BC = BC + 01
600A  13        INC  DE       :  DE = DE + 01
600B  23        INC  HL       :  HL = HL + 01
600C  C9        RET           :  Return

Start 6000
End   600C
```

Having entered Hex codes, go back to 6000 and check that the codes are correct. (Type M 6000 and check the contents of each location).

It is recommended that you would normally Save a machine code program before running it in case it crashes, which it is certainly likely to do unless you are an experienced machine code programmer. In this case, there is no real point in Saving the program, but if you wish to do so, refer to the section on "The Monitor in practice".

The last line of the routine is a return instruction which it is usual to use at the end of a machine code routine, to return you to Basic.

When you are satisfied that you have entered the above program correctly, type 'X' to restore the prompt and cursor.

B - Breakpoint

This command allows you to temporarily interrupt a machine code program at any point, and return control to the MONITOR, so that you can inspect the values in the CPU registers, and in RAM, and make corrections as necessary.

It takes the form 'B aaaa', where B is the Breakpoint command mode, and aaaa is the address of the instruction that the break will replace. (aaaa must be the address of the <u>first</u> byte of a multi-byte instruction).

The Op. codes in the three addresses aaaa, aaaa+1 and aaaa+2 are automatically saved in data bytes within the MONITOR, and these locations are then loaded with CD OF F7 (for the 48K version) or CD OF 77 (for the 16K version), which constitutes a CALL to the entry point of the MONITOR. It must be a CALL to maintain correct operation of the Stack.

On entering the MONITOR at this address, the values in the CPU registers are stored in data bytes within the MONITOR at the addresses shown in Appendix A; the Stack Pointer is set to the MONITOR Stack; the message "Press Break for Monitor" is displayed on the bottom line of the screen, in addition to the screen display your program has created. The MONITOR will now wait until you press 'Break' when it will display the prompt and cursor on the bottom line of the screen.

You will now be able to use any of the MONITOR commands to check or alter the routine, before returning control to the routine at the point at which the break occurred. As the MONITOR uses its own integral Stack, separate from the Program Stack, there is no danger of over-writing the Program Stack during a Breakpoint.

Before running the example just entered, enter a Break-point at address 6009. This will have the effect of stopping the program after the registers BC, DE and HL have been cleared, but before they are incremented.

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

Type B	the breakpoint command mode
Type 6009	the breakpoint address

There is no need to type ENTER, as the Breakpoint is set after typing the fourth digit. The screen will scroll, and the prompt will appear on the bottom line.

Using the 'M' command, check that 6009 to 600B now contain CD OF F7 (for 48K) or CD OF 77 (for 16K) in place of 03 13 23, the latter having been stored for later replacement.

You are now in a position to run the routine.

<u>J - Jump and execute</u>

The Jump command allows you to jump out of the control of the MONITOR to the starting address of any routine that you write, and it takes the form 'J aaaa' where J is the Jump command mode, and aaaa is the start address of your program.

You can run your machine code programs either with the MONITOR 'J' command, or by returning to Basic and using the USR function. Either way, the MONITOR commands are available to you after a Breakpoint.

In this example, we will use the 'J' command.

Type X	to restore the prompt and cursor.
Type J	to enter the Jump command.
Type 6000	the start address
Type ENTER	

The screen is cleared and the routine will run, and then return to the MONITOR, with the "Press Break for Monitor" message.

The sequence of events on executing the J command is:

i) the screen is cleared.
ii) the Stack Pointer is set to the program Stack.
iii) the start address is put into the Program Counter, and the program is executed.

The MONITOR uses its own integral Stack, which is set on entry to the monitor routine, therefore the program Stack, which is set by the Basic initialisation routines when the ZX Spectrum is switched on, must be reset before your program can be run. Item (ii) above does this for you. The use of two stacks helps make the

MONITOR invisible to your machine code program.

Having run, the program will have encountered the Breakpoint at address 6009, and have displayed "Press BREAK for Monitor". Press the BREAK/SPACE key (shifted or unshifted) to access the MONITOR.

The first operation after a Breakpoint should always be to replace the correct byte values to the addresses where the break occurred.

K - Break Restore

This command restores the correct values into the three bytes overwritten by the Breakpoint command.

    Type K

The screen will show K 6009 and will scroll up one line, displaying the prompt on the bottom line. There is no need to type ENTER. Using the M command, verify that the original codes have been replaced in addresses 6009 to 600B.

Only one Breakpoint can be entered at any time, as there is only enough room to store one address and three data bytes, so a Break Restore (K) command must be executed before the next Breakpoint (B) is defined, and it is recommended that a Break Restore (K) command is keyed immediately after a Breakpoint has been encountered.

If you enter an incorrect breakpoint with the B command, type K immediately afterwards to restore the original values to the incorrect breakpoint address, and then re-type the Breakpoint.

The K command can only restore the last entered Breakpoint.

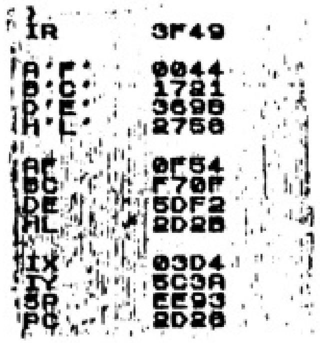
Let us now inspect the CPU registers, to make sure that the program is working as we expect.

R - Display values in CPU registers

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

    Type R

The screen scrolls up, automatically displaying the CPU register contents thus:



```
IR       3F49

A'F'     0044
B'C'     1721
D'E'     3698
H'L'     2758

AF       0F54
BC       F70F
DE       5DF2
HL       2D28

IX       03D4
IY       5C3A
SP       EE93
PC       2D28
```

There is no need to type ENTER.

As you will see, the Program Counter contains 6009, the address at which the Breakpoint occurred. The BC, DE and HL register pairs will all contain 0000. In this example, these are the only registers that we are interested in.

The CPU registers are displayed with their contents shown in Hex, and the values in the registers are stored in RAM at the addresses shown in Appendix A.

When the MONITOR is entered from a Breakpoint, the values of the registers immediately prior to the Breakpoint are stored in these memory locations, so that the operation of your routine can be checked, and corrections to the routine, or the register contents can be made before continuing.

Any changes to the CPU register values stored in RAM during a Breakpoint only take effect after a Jump (J) or Breakpoint Continue (C) command has been executed.

Having a) encountered one Breakpoint, b) restored the correct values after the break, and c) verified that the CPU registers have their correct values, we will now enter another Breakpoint, and continue the routine.

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 return 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
```

Each line shows the Hex contents of eight successive locations, with the Hex address of the first byte shown in the left hand column. The routine will only print complete lines, and if the end address that you specify is part of the way along a line, it will print up to the end of that line.

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

Type P	to enter the Printer command.
Type aaaa	the start address.
Type bbbb	the end address.

Typing X at any point will return you to the prompt and cursor.

Type ENTER.

IMPORTANT NOTE

It is possible to stop the Printer before the routine has been completed, by pressing the Break key (shifted only) but this will return you to the Basic monitor, and you will have to re-access the MONITOR as described on Page 2.

$ - String Entry

This command operates in a similar fashion to the 'M' command, and allows you to enter text directly from the keyboard. It is by no means a word processor, but it offers a much simpler method of text entry than by converting letters to their character codes, and entering the codes individually with the 'M' command.

The command takes the form: "$ aaaa' where $ is the command mode, and aaaa is the starting address of the text block.

Let us enter a simple message into a free area of RAM.

Type $ (Symbol Shift and '4') to enter the $ command.

An inverse $ will appear on the bottom line of the display.

Type 6100

The address is displayed as normal.

Up to the point of entering the last digit of the address, the X command will return you to the prompt. But from now on, this command is slightly different from all the others. As you may well want to type 'X' as a string entry, it cannot now be reserved for the escape command. So in this case only, the escape function is accessed by typing STOP (Symbol Shift and 'A'). The word STOP should serve to remind you of which key to press.

As you type in each letter of the message, it is displayed on the screen, to the right of the address and its present contents, and the character code is stored in that address. The screen scrolls automatically, displaying the next address and its present contents. You do not need to press ENTER to access the next address.

If there is a valid character code in an address, it will be displayed between the address itself and the cursor, otherwise a question mark is displayed. All upper and lower case letters can be entered by use of the Caps Shift key; also punctuation marks and spaces by the use of the Symbol Shift key. The only exception is '$' which is reserved as the command mode.

Graphics, user defined characters, and keywords and expressions such as ** cannot be entered directly. Inverse characters must be created by accessing the colour attributes part of the memory relating to the particular screen location. In other words, any single character that appears on a key top and that can be accessed by a single key press or by one level of shift can be entered. Any character normally accessed by the use of the GRAPHICS or EXTENDED modes will have to be entered by using the 'M' command to enter the Hex character code.

Now type in the following message:

This is "Spectrum MONITOR".

(Use Symbol Shift and P for the " marks). Having typed it in, (mistakes included) now review the message:

Type $	(Symbol Shift and 4) to re-enter the $ command.
Type 6100	the start address.
Type ENTER	and hold it pressed until the whole message is on the screen.

If the message is correct, you can now type STOP (Symbol Shift and A) to return to the prompt.

If you have made a typing error, or if you want to put another message at a new starting address, type $, to re-enter the $ command at the beginning, in the same way that the 'M' command is re-entered. You will have to enter the new address before making your correction, or starting your new message.

The repeating keyboard with the fast scrolling screen works as in the 'M' command, to allow you to review a message quickly.

Remember that, having entered the starting address, the escape command is accessed by typing STOP (Symbol Shift and A).

Z - Disassembler

This command will disassemble any part of RAM or ROM, either to the screen alone, or to both the screen and the ZX Printer. It provides a display that includes the Hex address of the first byte of the instruction, the Hex values of the bytes that relate to that instruction and the Z80 mnemonic for that instruction. The full set of Z80 mnemonics can be disassembled.

The command takes the form: 'Z aaaa bbbb' where 'Z' is the command mode, 'aaaa' is the hex starting address and 'bbbb' is the hex end address of the part of memory you wish to disassemble.

Type Z	to access the command.
Type 0000	the address of the start of the ROM
Type 0020	the end address.

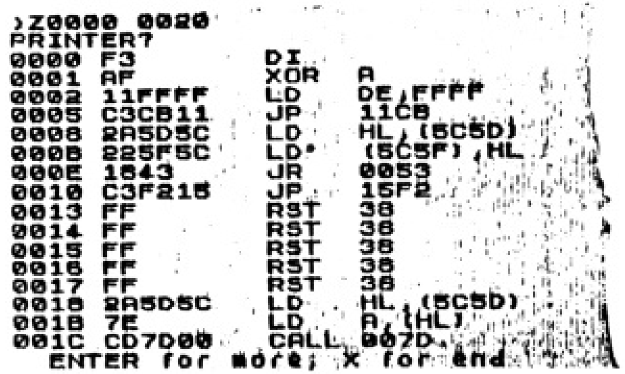
Having typed in the end address, the screen will scroll and display

PRINTER?

Your response to this is similar to the Basic "Scroll?" command. If you wish to use the printer, type ENTER;but if you only require a screen display, type 'N' (for NO).

Type N   for screen display only.

The disassembly will appear on the screen thus:



```
>Z0000 0020
PRINTER?
0000 F3          DI
0001 AF          XOR   A
0002 11FFFF      LD    DE,FFFF
0005 C3CB11      JP    11CB
0008 2A5D5C      LD    HL,(5C5D)
000B 225F5C      LD    (5C5F),HL
000E 1843        JR    0053
0010 C3F215      JP    15F2
0013 FF          RST   38
0014 FF          RST   38
0015 FF          RST   38
0016 FF          RST   38
0017 FF          RST   38
0018 2A5D5C      LD    HL,(5C5D)
001B 7E          LD    A,(HL)
001C CD7D00      CALL  007D
   ENTER for more; X for end
```

16 lines of disassembly will be displayed when using the screen only, followed by the message:

ENTER for more; X for end.

Pressing ENTER will display the next 16 lines, unless the end address is reached, when the prompt and cursor will be returned.

Typing 'X' in response to the above message will also return the prompt and cursor.

If you are disassembling to the Printer, the routine will continue, uninterrupted, until it reaches the end address. The Printer can be stopped by using the Break key in the normal way, which will return you to <u>BASIC</u>. You will then need to access the Monitor as described on Page 2.

If you try to disassemble to the Printer when it is not connected, the screen display will be produced on its own, the routine stopping when it has reached the end address.

All disassembled addresses and values are in Hex. Relative jumps show the address to which the jump will go, with the offset value shown with the hex coding for that instruction.

The only instructions that are displayed in a slightly different form from the published Zilog mnemonics are "JP(HL)" and the IX and IY counterparts "JP(IX)" and "JP(IY)". As these instructions jump to the address actually held in the register, the brackets are not shown in the display, which makes the action of the instruction a little clearer.

<u>N - Number Conversion</u>

This routine will convert Hex numbers to Decimal or vice versa.

Type N   the Number command.

The screen will display:

NUMBER H/D?

Type H (for Hex) or D (for Decimal) to indicate the number system of the number you wish to convert.

Type H   to convert a Hex number to Decimal.

The screen will scroll, and show 'H' (followed by the cursor. Now enter <u>four</u> Hex digits. (You must enter leading zeros when entering a Hex number).

Type 4000
Type ENTER

The display will now show:

H 4000 = 16384

with the prompt and cursor on the bottom line.

To convert from Decimal to Hex, Type D instead of H in response to "NUMBER H/D?", and enter your decimal number, without leading zeros. Again type ENTER to produce an answer.

The MONITOR in Practice

The purpose of this section is to explain the operations of the MONITOR that affect all commands, and which have not been covered so far, and to give some general precautions when using machine code.

1) The MONITOR display produces white characters on a blue background. After lengthy tests with various colour combinations, this gave the most readable display.

2) The loudspeaker will emit a short Beep when a key is pressed. The length of the Beep has been adjusted to give an easily.audible sound, without slowing down the response time of the keyboard. The System Variable PIP does not affect the MONITOR'S keyboard Beep.

3) As has been explained earlier, the MONITOR uses its own internal Stack except when a program is running. The program Stack is reset from "SP" in the data block shown in Appendix A whenever a 'J' (Jump) or 'C' (Break Continue) command is executed. When the 'Y' (Return) command is used, the program Stack is cleared and reset to its normal Basic starting point as defined in the System Variable ERR SP.

4) The CPU register values stored in the addresses shown in Appendix A are only reloaded into the CPU when a Jump (J) or Break Continue (C) command is executed. Returning to Basic (to access your machine code via the USR function) resets the CPU register values as defined by the Basic ROM routines.

5) In addition to the precautions shown on Page 180 of the Sinclair manual advising you not to use the I or IY registers in machine code programs, it is recommended that, if you need to use the alternative BC, DE and HL register pairs and wish to return to Basic after your machine code program, you should save the values held in the alternative registers at the start of your program, and reload them before returning to Basic.

6) The Spectrum MONITOR does not have its own Save and Load routines because the Basic Save and Load routines in the Spectrum allow you to record machine code programs onto cassette. Having written your machine code, you would use the Number Conversion (N) command to convert your Hex start and end addresses into Decimal, calculate the length of your program, and use the Return (Y) command to return to Basic to Save and Verify your machine code.

CONCLUSION

All the commands of the MONITOR have now been demonstrated, and you are ready to start writing and running your own machine code programs.

With the MONITOR to help, you are limited only by your own ingenuity.

Good luck.

APPENDIX A — CPU REGISTERS

The values in the CPU registers are stored in the following locations after a Breakpoint, and can be altered using the M command. The alterations only take effect if the Breakpoint Continue (C) command or a Jump (J) command is used to access the machine code.

16K	48K	REG
7F3D	FF3D	R
7F3E	FF3E	I
7F3F	FF3F	F'
7F40	FF40	A'
7F41	FF41	C'
7F42	FF42	B'
7F43	FF43	E'
7F44	FF44	D'
7F45	FF45	L'
7F46	FF46	H'
7F47	FF47	F
7F48	FF48	A
7F49	FF49	C
7F4A	FF4A	B
7F4B	FF4B	E
7F4C	FF4C	D
7F4D	FF4D	L
7F4E	FF4E	H
7F4F	FF4F	IX (Low)
7F50	FF50	IX (High)
7F51	FF51	IY (Low)
7F52	FF52	IY (High)
7F53	FF53	SP (Low)
7F54	FF54	SP (High)
7F55	FF55	PC (Low)
7F56	FF56	PC (High)

Any changes to the I and IY registers may cause the ZX SPECTRUM to crash.

APPENDIX B — SUMMARY OF COMMANDS

M aaaa nn ENTER	<u>Memory Location</u> & contents in Hex.
	aaaa = address.
	nn = new contents value
	ENTER = next location (repeating)
	'M' re-enters command.

X	<u>Escape</u> to prompt, and wait for new command.
[illegible] aaaa bbbb nn	<u>Insert</u>. aaaa = address 1st. byte insertion bbbb = address highest byte to be moved nn = no. bytes to be inserted Type ENTER to execute.
D aaaa bbbb nn	<u>Delete</u>. aaaa = address 1st byte deletion bbbb = address highest byte to be moved nn = no. bytes to be deleted Type ENTER to execute

A aaaa bbbb cccc	<u>Area Relocate</u>.
	aaaa = present start address
	bbbb = present end address
	cccc = new start address
	Type ENTER to execute.

F aaaa bbbb xx	<u>Fill</u>. aaaa = start address of area to fill bbbb = end address to area to fill xx = value to be loaded into area Type ENTER to execute.
Z aaaa bbbb ENTER/N	<u>Disassembler</u>. aaaa = addr. start of disassembly bbbb = addr. end of disassembly Type ENTER for printer or N for screen. BREAK stops printer early.

Y	<u>Return</u>. Returns to Basic and resets the Basic stack Pointer. Type ENTER to execute.
J aaaa	<u>Jump</u>. aaaa = start address of program Type ENTER to execute.
B aaaa	<u>Breakpoint</u> aaaa = address of Breakpoint Executes automatically on typing 4th. address digit.
K	<u>Break Restore</u>. Executes automatically, and restores last entered Breakpoint.
R	<u>Register Display</u>. Executes automatically, displaying values in CPU registers.
C	<u>Breakpoint Continue</u>. Type ENTER to execute. Continues program execution after a Breakpoint.
$ aaaa letter/ENTER	<u>String Entry</u>. aaaa = address 1st byte of string Letter = character from keyboard ENTER = next location (repeating) '$' re-enters command. Typing a letter automatically increments address.
P aaaa bbbb	<u>Printer</u>. aaaa = address 1st byte to LPrint bbbb = adddrss last byte required to LPrint Type ENTER to execute Type Break to stop early.
N H/D, Number	<u>Number Conversion</u>. H/D = Hex or Decimal number Type ENTER to execute.

<u>NOTES</u>