INSTRUCTIONS FOR SPECTRUM MONITOR

=================================

The ideal tool to help the Machine Code beginner to get started. Contains all the commands required for the more experienced programmer to run and de-bug Machine Code programs. Commands allow access to memory in Hex or ASCII: Breakpoints to interrupt Machine Code programs: CPU registers display.Commands to help you organise memory or data include Block move, Insert, Delete and Fill. Disassembler to help you understand the ROM and check your own programs in RAM. Hex-Dec-Hex number converter, Printer routines, and full compatibility with Basic.

Spectrum MONITOR is supplied with a 32 page Owner's Manual designed to help the beginner towards a better understanding of Machine Code.The MACHINE CODE SYSTEM comprises the Spectrum MONITOR and the EDITOR ASSEMBLER (available only as separate programs) which together form the key to Machine Code success.

For the beginner or the experienced programmer, the MACHINE CODE SYSTEM is the most valuable software purchase you will make.

**(c) PICTURESQUE 1982**

6 Corkscrew Hill, West Wickham, Kent, BR4 9BB, England.

Internal booklet "Spectrum MONITOR OWNER'S MANUAL VERSION 2.1":

Spectrum 16k & 48k MACHINE CODE SYSTEM

CONTENTS

--------

INTRODUCTION

LOADING THE SPECTRUM MONITOR

ACCESS TO SPECTRUM MONITOR

PROMPT AND CURSOR

COMMAND MODES

M - Memory Location

X - Escape

I - Insert

D - Delete

A - Area Relocate

F - Fill

Y - Return

B - Breakpoint

J - Jump & Execute

K - Break Restore

R - Register Display

C - Breakpoint Continue

P - Print Hex Dump

$ - String Entry

Z - Disassembler

N - Number Conversion

E - Access Editor Assembler

T - Trace (Single Step)

THE MONITOR IN PRACTICE

APPENDICES:

A - CPU Register Data Bytes

B - Summary of Commands & Forms

Copyright 1984 by Picturesque. All rights reserved. This book and the accompanying computer program are copyright. No part of either this book or the accompanying computer program may be reproduced, copied, lent, hired, or transmitted by any means without the prior written consent of the publisher.

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. Used in conjunction with one of the many books available that teach the principles and practice of Machine code programming, the Spectrum MONITOR is an ideal learning tool for the beginner. For the more experienced user, the Spectrum MONITOR offers the commands necessary for successful Machine code programming and de-bugging.

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 5K 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 occupies about 5K 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 1/2 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 proprietary 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 to 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. 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.

The program loads in two parts and auto runs. Once loaded, the Monitor can be customised to work with the following range of Centronics Printer Interfaces:

1) Kempston Interface

2) Kempston 'E' Interface

3) Euroelectronics Lprint Interface

4) Hilderbay Interface, Type H

5) Hilderbay Interface, Type T

6) Hilderbay Interface, Type K

7) Morex Centronics Interface

Options are also given to allow use of the normal ZX Printer, as well as RS232 output via the Sinclair Interface 1. Enter the number of the option you require (there is no need to press ENTER). If you are using a Centronics Interface option you will then be asked to state whether your printer gives an automatic Line Feed after a Carriage Return. If it does, select option 1, otherwise select option 2, and the Monitor's print routines will be set to send a separate Line Feed. (If in doubt about this, you should refer to the operating instructions supplied with your Printer. It is usually possible to select this feature by means of switches inside your printer).

Having customised your program for your printer, you will then be given the option of making a copy of the customised program on to a cassette or on to a Microdrive Cartridge. (Microdrive Cartridges are NOT supplied). If you intend Saving on to Microdrive, you should make sure that you have a Formatted Cartridge with at least 8K of spare capacity available BEFORE you start customising the program. If you Save on to cassette, you should only connect the MIC lead when Saving, and the EAR lead when Verifying.

WARNING - THIS PROGRAM IS COPYRIGHT and ONE COPY ONLY may be made for your OWN PERSONAL USE. It is illegal to sell copies or to give copies to friends.

The customised program is saved in two parts: a Basic loader, and the machine code of the Monitor.

The Basic loader program of the customised Monitor is saved with the name "Mon 16" or "Mon 48" and should be loaded from cassette by

LOAD "Mon 16" or

LOAD "Mon 48"

To load from Microdrive Cartridge, use

LOAD \*"m";n;"Mon 16" or

LOAD \*"m";n;"Mon 48"

where 'n' is the drive number. In all cases, the customised Monitor auto runs on loading.

ACCESS TO SPECTRUM MONITOR

Having successfully loaded, the MONITOR automatically resets Ramtop to below itself, and displays the message:

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. If you return to Basic while using the Monitor, you can re-enter the Monitor by means of

RANDOMISE USR 30479 (16K version)

RANDOMISE USR 63247 (48K version)

PROMPT & CURSOR

---------------

The prompt ( > ) 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 (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.

T: Trace (or Single Step) with front panel display.

E: Access to Picturesque Editor Assembler.

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 any memory location in RAM or RAM, 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.

>M6000 00 -

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

>M6000 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.

>M6000 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

>M6000 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

>M6000 00 FF

6001 00

M6000 FF -

The contents of 6000 are shown as FF.

To summarise 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 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 the 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 1 1/2 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 $ command, described later) 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 of '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.

.

.

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 and 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 address 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 of 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 address relating to the area of RAM that has been moved by Delete, will 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 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 E8EC to FF10, (48K); or 68EC to 7F10 (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', where 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

----------

Printed on the 'Y' key is the keyword RETURN, and by pressing this key, followed by ENTER, when the prompt and cursor alone are visible on the bottom 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, and enable the Z80 Interrupts, 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 first 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 0F F7 (for the 48K version) or CD 0F 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 this routine, before returning control to the routine at the point at which the break occurred. As the MONITOR uses its own illegal 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 Breakpoint 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 0F F7 (for 48K) or CD 0F 77 (for 16K) in place of 03 13 23, the latter having stored for later replacement. You are now in a position to run the routine up to the breakpoint.

J - Jump and execute

--------------------

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 address 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 address 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 3F24

SZ H PNC

A'F' 00 01000100

B'C' 1721

D'E' 369B

H'L' 2758

SZ H PNC

AF 0F 01010100

BC 0000

DE 0000

HL 0000

IX 03D4

IY 5C3A

SP FF42

PC 6009

>

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 FLAGS registers are shown in BIT form, with the purpose of each flag indicated above. If a flag is SET, a 1 is indicated, and if it is reset, a 0 is shown. 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 Breakpoint address is put into the Program Counter, and execution is resumed.

Type C

Type ENTER

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

When the prompt appears, after pressing 'BREAK',

Type K to restore the bytes occupied by the Breakpoint.

Type R to display the registers.

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

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

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

or

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

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

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

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

P - Print Hex Dump

------------------

This command allows you to produce a Hex Dump of any section of memory on to the screen or your Printer. It takes the form 'P aaaa bbbb' where P is the command name, 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 display is produced thus:

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 access the command by typing P followed by the address from which you wish to start the display. If you want to define the address at which the

display will stop, enter that Hex address following the start address, otherwise press ENTER. You will then be asked to select Print options for this command. Respond with:

N if you require a screen output.

/ (Symbol Shift + V) if you require a Printer output. This will direct the printer output to the ZX Printer or via a Centronics interface, depending upon the way the program was initially customised.

\* (Symbol Shift + B) if you require the output via the RS232 facility of Interface 1. (If Interface 1 is not connected, this will produce a Basic error 'i', and leave you in Basic). To use a Printer with the Interface 1 RS232, use the Basic FORMAT command to set the Baud Rate of your Printer. The Monitor uses the "t" type of RS232 output, sending a Carriage Return and a Line Feed. (See Sinclair's Microdrive Manual).

For screen output, 16 lines of screen information are displayed, whereupon you can press ENTER to display the next 16 lines, or press X to escape. When using a Printer option, it is recommended that you define an end address, as the Printer option will continue operating until the end address is reached, or until the BREAK key (with Caps Shift) is pressed. The end address defaults to FFFF Hex if it is not specified. Pressing the BREAK key (with Caps Shift) will give a Basic error message and leave you in Basic ONLY if you are using the ZX Printer or RS232. Breaking out of a Centronics Printer routine returns you to the main Monitor prompt.

$ - String Entry

----------------

This command operates in a similar fashion to the 'M' command, but 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, and single character that appears on a key top and that can be accessed by a single key press or 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 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 or to the 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 start address (e.g. start of ROM)

You can now either type in an end Hex address where you want the disassembly to stop, or you can press ENTER to achieve an open-ended disassembly in the same way that the P command operates. You will be given the same Screen/Printer options as described in the P command above. Again the end address defaults to FFFF hex if it is not defined. If you attempt to disassemble any part of Spectrum Monitor, an error message 'INVALID ADDRESS' is displayed.

The disassembly will appear as follows:

>Z0000 0020

PRINTER? (N / \*)

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 a Printer, the routine continues uninterrupted until it reaches the end address. The printer can be stopped early by pressing CAPS SHIFT and BREAK. If a Centronics Printer option was selected on customising the MONITOR, BREAK will return you to the Monitor's prompt and cursor. If you are using the ZX Printer or an RS232 printer, the BREAK key will return you to Basic. You will then need to re-access the Monitor as described earlier.

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.

N - Number Conversion

---------------------

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 four 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.

E - Entry to 48K Assembler from 16K Monitor

-------------------------------------------

Type X to restore the prompt and cursor to the bottom line of the screen and then type E followed by ENTER. If the 48K Assembler is resident, it will be accessed, otherwise the Monitor's prompt is displayed. A full check of the Assembler is impossible, and it is up to the user to ensure that the Assembler and its associated tables and Source Code have not been overwritten by Monitor commands.

T - Trace (or Single Step)

--------------------------

The Trace command allows you to execute Machine Code in ROM or RAM one instruction at a time, or in certain specified blocks. But at ALL times, the execution of the Machine Code is strictly under the control of the Trace command, and a crash is almost impossible. Commands such as LDIR (a self repeating block move) can cause a crash by overwriting the Monitor, as can any instruction that writes into memory. Altering the values of the Stack Pointer can also overwrite the Monitor, and reference to the memory map of the new Monitor should help to avoid mistakes.

STEP

2D28 4F LD C,A

IR 3F4C STACK

SZ H PNC 0009

A'F' 00 01000100 000D

B'C' 1721 10ED

D'E' 369B 6C5D

H'L' 2758 \*3365

SZ H PNC

AF 0F 01010100

BC 770F (BC)C3 CB 71 2A A4

DE 5D15 (DE)00 00 6A 7C 00

HL 2D2B (HL)FD 21 3A 5C AF

IX 03D4

IY 5C3A

SP FDD3

PC 2D28

M0000 F3 AF 11 FF FF C3 CB 11 2A

The comprehensive screen display throughout the Trace command gives a permanent display of the CPU register contents, a disassembly of the current and the next instruction, the contents of the last five stack locations and the contents of specified memory locations. The display is formed as shown above.

The Trace command is accessed from the Main Monitor by typing T (when the prompt and cursor are visible on the bottom line of the screen) followed by the 4-digit Hex address of the instruction from which single stepping is to start. If you do not specify a Hex address, but press ENTER immediately after typing T, the Trace command is accessed, and the address shown against PC in the register display is used as the starting address. To return to the Main Monitor from the Trace command, type X. The screen will clear, and the Main Monitor's prompt and cursor will appear on the bottom screen line. All register values are passed between the Main Monitor and the Trace command, and you can therefore use all the available commands of the Monitor to

debug your machine code.

The Trace command contains its own set of commands which are separate from the main Monitor commands, and allow control over

a) executing instructions singly;

b) running on to a Breakpoint (not the same as a Breakpoint set by the main Monitor);

and

c) skipping to the end of a subroutine.

Any selection of code that will only operate correctly when run at full speed (e.g. timing loops, sound output or interfaces to external equipment) will not operate correctly in the Trace command as each instruction is decoded and executed separately. The main Monitor commands for setting a Breakpoint and running the code should be used instead.

To demonstrate the trace command, enter the following short routine using the M command in the main Monitor to enter the Hex code shown in bold type.

(If you have already accessed the Trace command, type X to exit to the main Monitor before entering the example program).

6400 0005 ORG 6400H

6400 212C64 0010 LD Hl,642CH

6403 113412 0015 LD DE,1234H

6406 7A 0020 LD A,D

6407 CD1064 0025 CALL 6410H

640A 7B 0030 LD A,E

640B CD1064 0035 CALL 6410H

640E 00 0040 NOP

640F 00 0045 NOP

6410 4F 0050 LD C,A

6411 E6F0 0055 AND 0F0H

6413 1F 0060 RRA

6414 1F 0065 RRA

6415 1F 0070 RRA

6416 1F 0075 RRA

6417 CD2164 0080 CALL 6421H

641A 79 0085 LD A,C

641B E60F 0090 AND 0FH

641D CD2164 0095 CALL 6421H

6420 C9 0100 RET

6421 C630 0105 ADD 30H

6423 FE3A 0110 CP 3AH

6425 3802 0115 JR C,+2

6427 C607 0120 ADD 7

6429 77 0125 LD (HL),A

642A 23 0130 INC HL

642B C9 0135 RET

642C 00 0140 DEFB 0

642D 00 0145 DEFB 0

642E 00 0150 DEFB 0

642F 00 0155 DEFB 0

0160 END

The code from 6400 to 640D converts the value held in the DE register pair into four ASCII character codes representing that value, and puts the four character codes into memory at 642C (this location held in HL). The code starting at 6410 and at 6421 are subroutines called during this process.

Having entered the code, type X to restore the main Monitor Prompt and then type T followed by 6400. The Trace display will be formed in a similar fashion to that shown above. The STEP mode is automatically selected, and is shown at the top of the screen. Below that is a disassembly of the next instruction to be executed. The Register display shows the current register contents (which could by anything as no instructions have been executed yet).

The Stack display shows the last five pairs of bytes on the Stack, with the last value placed on the Stack indicated by the \*.

The M display line at the bottom of the screen gives a window on to memory locations, and the starting address can be changed at any time by typing M followed by a four digit Hex address. In the example above, it would be useful to see a display of the memory that will eventually contain the four ASCII characters, so type M642C. The M display is updated as soon as the fourth digit is entered and you do not need to press ENTER.

While in the Trace command, the address of the next instruction can be changed at any time by typing S followed by the new Hex address followed by ENTER. This restores STEP mode and updates the display ready for execution of the next instruction at the new address.

To execute the first instruction (LD HL,642CH) simply press ENTER. The display will now update and show 642C in HL. Alongside that value are shown the contents of the bytes at and immediately following that address. You will see that the disassembly of the first instruction has scrolled up one line, and that the next instruction (LD DE,1234H) is displayed on the second disassembly line. The upper of the two Disassembly lines is always the instruction just executed, and the lower is always the instruction about to be executed.

Continue pressing ENTER and STEP through the whole routine, stopping when the NOP instruction at address 640E is displayed on the second disassembly line, and observe the effect on the display of each instruction. The register display of the PC value will ALWAYS show the address of the NEXT instruction to be executed, and the Stack display will only change when the Stack is used (in this case the CALL and RET instructions).

At any time you can exit from the Trace command back to the main Monitor by typing X. The screen will clear and the normal Monitor Prompt will be displayed at the bottom of the screen. You may find that you have to press X more than once to do this. For example, if you are in the process of typing in a command within the Trace facility, the first X will clear that command and the second X will access the main Monitor.

You may have found that single stepping through the two subroutines became tedious after you has stepped through them once, and proved that they did work. To speed up this process the Trace command contains two functions that can be used to automatically step through sections of your program that have already been proved. They are:

(i) BREAKPOINT

Do not confuse this Breakpoint function with the BREAKPOINT command in the main Monitor. Setting a Breakpoint while in the Trace command does not alter your program, as it does in the Main Monitor. The Breakpoint address is stored, and after each instruction is executed by the Trace command, the address of the next instruction in your program is compared with the Breakpoint address. If the program address is not equal to the Breakpoint, the next instruction is decoded and automatically executed, until the program address is equal to the Breakpoint address, whereupon the whole display is updated and the STEP mode is accessed.

To use the above example to demonstrate the use of the Breakpoint, type X to access the main Monitor and use the M command to write 00 into the four memory locations from 642C Hex. (You do not need to do this to use the Breakpoint, but in this example it will help to clarify what is happening by clearing the ASCII codes already written there). Re-enter the Trace command by typing T6400 and then type B640E, followed by ENTER. This has put the Trace command into BREAKPOINT mode and this is displayed at the top of the screen.

Pressing ENTER again will cause the program to be executed under the control of the Trace command until the Breakpoint is reached. The display is then fully updated and STEP mode accessed. While the program was being executed, the only part of the display to be updated was the value of the PC in the Register display. Should you have set a Breakpoint at an address that for some reason is never reached (perhaps a conditional instruction that has jumped elsewhere) the constantly changing PC value will indicate that the command is still running. In this case you can escape by pressing CAPS shift and BREAK together, which will access the STEP

mode again and update the display showing the address at which the Break occurred.

You may find that this leaves unwanted address or register contents on the program stack, and an incorrect value for SP in the register display. This will not affect the operation of the Monitor as is uses its own stack. The

simplest was to restore the Stack Pointer to its normal value (usually as set by Basic) is to return to Basic (Key Y) and then to re-access the Monitor as described above.

(ii) SKIP to RET

This function can be accessed at any time while in the STEP mode. Type R and press ENTER. The top line of the display will now show STEP SKIP to RET. Pressing ENTER again will cause the Trace command the execute your program automatically, and stop (in STEP mode) when it has executed the RET instruction associated with the next CALL instruction. While the subroutine is being executed, the display of the PC register value is the only part of the display to be updated. On completion of the subroutine the whole display is updated and the normal STEP mode is accessed.

To demonstrate, type S6400 (assuming you are still in the TRACE command) followed by ENTER. The display will update and you will be ready to single step from address 6400. Press ENTER three times to execute the first three instructions of the example program. The next instruction will now be CALL 6410. Press R followed by ENTER, to access the SKIP to RET mode.

Press ENTER again, and the subroutine at 6410 will be executed in its entirety, including the two CALL instructions within it, and the display will update showing the LD A,E instruction at 640A as the next instruction on the second disassembly line, and the original CALL on the line above. The normal STEP mode will also be indicated at the top of the screen.

You can now continue stepping through the program using the ENTER key. So press ENTER to execute the LD A,E, and then press ENTER again to execute the CALL 6410. The next instruction will be shown as LD C,A at address 6410 and is the first instruction in the subroutine. If you now press R and ENTER to access the SKIP to RET mode, a second press on ENTER will execute the program up to the RET associated with the next CALL. The next CALL is at address 6417 (CALL 6421), and execution will stop showing LD A,C at address 641A as the next instruction. Follow this through by referring to the program listing to clarify what has happened.

If you use the SKIP to RET function, and a RET instruction is found before a CALL instruction, that RET instruction will cause automatic execution to stop and the STEP mode to be accessed. This can be demonstrated by single stepping through the example program using the ENTER key in normal STEP mode, until an instruction in the subroutine at 6421 is shown at the next instruction. Using the SKIP to RET mode at this point will automatically execute the program until the RET at 642B. The routine will stop in the STEP mode, showing the next instruction as being the one following the CALL that called the subroutine.

To summarise SKIP to RET, automatic execution continues up to and including the RET instruction associated with the next CALL, OR to the next RET if a CALL is not encountered in the meantime. In the same way as the Z (Disassemble) command in the main Monitor gives an 'INVALID ADDRESS' error if you try to disassemble the Monitor, you cannot single step through the Monitor program, the same error message being displayed. If this error occurs while you are in the Trace command, an automatic exit to the main Monitor is made, and the main Monitor's Prompt is shown as the bottom of the screen.

REGISTER POINTER

You will notice that an inverse video ( > ) is displayed immediately to the left of the register contents column. This is the Register Pointer and is used to enable the contents of a register pair to be easily changed from the TRACE command. The pointer can be moved down by using the cursor control keys (6 for down and 7 for up). To alter the contents of a register, move the pointer until it indicates the required register pair, and then press the 1 key. The current value is shown on the bottom line of the Trace display. You can now either press ENTER to restore the same values or type in a 4 digit Hex number to change the register value.

This facility only operates in the STEP mode of the Trace command and can easily be used while debugging software using the main Monitor. Access the Trace command by typing T followed by ENTER. All register values are passed to the Trace command, and after making your register alterations, exit to the main Monitor by typing X. Again, all register values are pressed back to the main Monitor.

If your program uses routines that must be run at full Machine Code speed to operate (e.g. timing loops or sound output), you will need to escape from the Trace command by typing X and use the main Monitor commands to run those routines properly. All the register values are pressed back to the main Monitor, and you can then set a normal Breakpoint (using the Main Monitor B command) and use the C (Continue) command to continue executing your program in real time from the point at which Trace stopped.

Having encountered the Breakpoint, use K to clear the Breakpoint, and access the Trace command by typing T followed by ENTER. All the Register values and the address of the next instruction are passed back to the Trace command reply for you to continue Single Stepping from the address shown against PC in the register display.

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 is created using the screen colours defined by the Spectrum's System Variable ATTR-P and can be changed by returning to Basic (Key Y) and using INK, PAPER and BORDER commands.

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 address 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) All the commands of the MONITOR operate even if the Z80 Interrupts have been disabled by the machine code you are testing. (Normally, disabling the Interrupts will stop the Spectrum keyboard from being read). Using the 'J' or 'C' commands to execute machine code program does not alter the Interrupt status, but the Return (Y) to Basic command automatically enables the Interrupts to ensure that the keyboard is scanned in Basic.

6) 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.

7) 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 on to cassette. Having written your machine code, you would use the Number Conversion (N) command to convert your Hex start and end address 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. The Spectrum MONITOR can be used on its own to enter Machine code programs, but to simplify the process of Machine code programming, PICTURESQUE also market an EDITOR ASSEMBLER that is fully compatible with the MONITOR, and which allows you to enter Z80 mnemonics into a listing, with line numbers and Labels. The listing is totally independent from Basic and uses a unique 40 column Screen display.

If you own a 48K Spectrum, the ASSEMBLER and the MONITOR can both reside in memory together, creating the most versatile Machine code system available for the ZX Spectrum. The ASSEMBLER is also designed to make the best use of available memory in a 16K Spectrum.

APPENDIX A - CPU REGISTERS

The value in the CPU Registers are stored in the following locations after a Breakpoint, and throughout the Trace command. The values can be altered by using the M command in the Main Monitor or by using the Register Pointer in the Trace command. Any alterations only take effect if the Breakpoint Continue (C) command or the Jump (J) command is used from the Main Monitor to access the machine code, or if the Trace (T) command is used.

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)

7456 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 Memory location & contents in Hex.

aaaa = address.

nn = new contents value

ENTER = next location (repeating)

'M' re-enters command.

X Escape to prompt, and wait for new command.

I aaaa bbbb nn Insert.

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 Delete.

aaaa = address 1st byte deletion

bbbb = address highest by to be moved

nn = no. bytes to be deleted

Type ENTER to execute

A aaaa bbbb cccc Area Relocate.

aaaa = present start address

bbbb = present end address

cccc = new start address

Type ENTER to execute.

F aaaa bbbb xx Fill.

aaaa = start address of area to fill

bbbb = end address of area to fill

xx = value to be loaded into area

Type ENTER to execute.

Z aaaa bbbb/ENTER Disassembler.

aaaa = addr. start of disassembly

bbbb = addr. end of disassembly (or ENTER for open-ended option)

Print option: N = Screen

/ = ZX Printer or Centronics Printer

\* = RS232 via Interface 1

BREAK stops printer early

Y Return. Returns to Basic and resets the Basic stack Pointer.

Type ENTER to execute.

J aaaa Jump.

aaaa = start address of program

Type ENTER to execute.

B aaaa Breakpoint.

aaaa = address of Breakpoint

Execute automatically on typing 4th address digit.

K Break Restore.

Executes automatically, and restores last entered Breakpoint.

R Register Display.

Executes automatically, displaying values in CPU registers.

C Breakpoint Continue. Continue program execution after a Breakpoint.

Type ENTER to execute.

$ aaaa letter/ENTER String Entry.

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/ENTER Print Hex Dump

aaaa = addr. first byte to print

bbbb = addr. last byte to print (or ENTER for open-ended option)

Print options: N = Screen

/ = ZX Printer or Centronics Printer

\* = RS232 via Interface 1

BREAK stops printer early

N H/D, Number Number Conversion. H/D = Hex or Decimal number

Type ENTER to execute.

E Editor Assembler. Access to 48K Editor Assembler if resident in memory.

Type ENTER to execute.

T aaaa/ENTER Trace (Single Step).

aaaa = addr. from which Trace starts.

Typing ENTER in place of aaaa starts Trace from addr. in PC in Register Display.