

To move PA-man lead to Bcoo

Start of Programs 26004

SHARPSOFT

PPPPPPF		AAAAAA		MM	MM	000000	N	N
P	P	A	A	M	M	0	NN	N
P	P	A	A	M	M	0	N	N
P	P	A	A	M	M	0	N	N
PPPPPPF		AAAAAAA	-----	M	M	0	N	N
P		A	A	M	M	0	N	N
P		A	A	M	M	0	N	NN
P		A	A	M	M	000000	N	N

P A - M O N 2.0A

A Machine Language Monitor for the Sharp MZ-80A

Copyright (c) 1983 by
Peter Andersson

SHARPSOFT Ltd
86-90 Paul St
LONDON EC2A 4NE

INTRODUCTION

PA-MON is a combined machine language monitor and debug utility for machine language programs. It consists of 31 different commands, such as:

- Modify memory using hexadecimal numbers
- Enter machine code using standard Z80 mnemonics
- Fill memory
- Copy
- Relocate machine code programs
- Disassemble
- Tabulate
- Write a cassette file
- Read a cassette file
- Verify a cassette file
- Search memory
- Execute machine code programs using a Z80 simulator
- Execute machine code programs in real time using breakpoints
- Turn the printer on or off

Also a command is included which makes it possible to relocate PA-MON to another memory area. This means that PA-MON is able to be co-resident with any machine code program, provided enough memory remains. PA-MON requires less than 5 KByte of memory.

OUTLINE OF THE MANUAL

The commands have been divided into six different groups, namely:

- Memory Manipulation commands
- Memory Display commands
- File Read and Write commands
- Search and Compare commands
- Debug commands
- Special commands.

Each command will be described separately, using examples when necessary. At the end of this manual a Command Summary is included.

The following conventions will be used:

The syntax definitions show optional items in square brackets ([]). All command parameters are shown separated by spaces for clarity in the syntax definitions, although it is not necessary to separate the command name and the first parameter.

All numerical parameters must be entered as 2- or 4-digit hexadecimal numbers and leading zeros must be added if necessary.

The following definitions apply throughout this manual:

<address>, <address1>, <address2>, ...	4-digit hexadecimal numbers
<type>	2-digit hexadecimal number
<count>	2- or 4-digit hexadecimal number
<number1>, <number2>	4-digit hexadecimal numbers
<port>	2-digit hexadecimal number

<CR> is used to denote the CR (or ENT) key.

If a syntactically illegal command is entered, three question marks (???) will be displayed.

If the user tries to alter a write protected memory location the error message "Protected" will appear. Please refer to the Write Protection Off (@) and Write Protection On (*) commands for further details.

The prompt normally used is an asterix (*).

The "Z80-Assembly Language Programming Manual" referred to in the command descriptions is published by Zilog Inc. The Z80 Instruction Set section of the MZ-80A Owner's Manual can be used instead.

MEMORY MANIPULATION COMMANDS

MODIFY (M) Command

This command is used to modify the contents in memory using two-digit hexadecimal numbers or ASCII characters.

Syntax: M [I] <address> <CR>

<address> is the address of the first memory location to be modified. If M <address> is entered the computer responds with:

aaaa bb

where aaaa is the specified <address> and bb the contents in that memory location, and the cursor is positioned on the first digit of bb.

To alter this value, just rewrite bb with the desired value. When the CR key is pressed, this new value is entered into the memory and a new line will appear showing the contents in the next memory location. The computer now waits for more to be entered.

It is possible to enter more than one two-digit hexadecimal number on the same line, just by continuing to type in new numbers. The numbers can be separated with one or more spaces, but this is not necessary. It is essential that all numbers are entered as two-digit numbers and a leading zero must be added if values less than 10H are entered.

If an ASCII character is entered separated with at least one space on each side, then the ASCII value of that character will be entered into the memory. Note that ASCII characters can be mixed with hexadecimal numbers on the same line.

The following example shows what a legal line can look like:

Suppose that the following line has been entered from the keyboard

2B05 01020396 S H A R P 20 M Z - B O K OD

After pressing the CR key the following hexadecimal values will be entered into the memory starting at address 2B05H:

01 02 03 96 53 48 41 52 50 20 4D 5A 2D 38 30 4B OD

(Note that, for example, 53 = ASCII for 'S')

It is also possible to alter the address: Just move the cursor, using the cursor keys, to the address and rewrite it. The cursor keys can also be used to go back and correct previously entered lines. Just remember to press CR when the cursor is positioned on the corrected line.

If the command is entered as M I <address> the only difference from what has been described above is what will be displayed on the screen each time the CR key is pressed. Instead of just displaying the contents of one memory location, the contents in one to four consecutive locations are displayed. The computer assumes that the contents in the memory is a Z80 machine code instruction and the number of bytes displayed reflects how many bytes are occupied by this instruction.

To make it easier to specify the displacement in the Z80 instructions JR and DJNZ it is possible to enter the absolute address of the destination of the jump instead of the relative equivalent. To do this, enter a period (".") immediately followed by the absolute address. This five character construction must be separated with at least one space on each side.

An example:

```
1523 18 .1516
```

When entering this line a JR instruction will be inserted at memory location 1523H and 1524H, which will cause a jump to address 1516H if executed.

If something illegal is entered a bell will sound, the address where the error occurred will be displayed, and the modification of the memory proceeds from this address.

To return to the command level just press BREAK.

Z80 MNEMONIC (Z) Command

This command is used to enter machine code instructions into memory using standard Z80 mnemonics.

Syntax: Z <address> <CR>

<address> is the starting address where the instructions are to be entered.

After pressing CR, the address <address> will be displayed and the computer now waits for Z80 instructions to be entered.

All numeric arguments in the instructions must be entered as two- or four-digit hexadecimal numbers and leading zeros must be added if necessary.

If something illegal is entered a bell will sound and the same address will be displayed again.

When entering the relative jump instructions, JR and DJNZ, the absolute address of the destination of the jump must be used, not the relative displacement.

The argument in the RST instruction must be entered as 00, 08, 10,, 38, which is the format specified in the "Z80-Assembly Language Programming Manual".

The bit number in the BIT, RES, and SET instructions must be entered as a one-digit number.

To find the correct format for an instruction, the Disassemble (D) command can be used.

As in the Modify (M) command the cursor keys can be used to go back and correct previously entered lines. It is also possible to alter the address displayed on the left of the line.

To return to the command level, just press BREAK.

Example: *Z5000
5000 LD B,05
5002 LD DE,6000
5005 CALL 0003
5008 DJNZ 5002
500A RET
500B

FILL (F) Command

This command is used to fill a section of memory with a constant value or a repeated sequence.

Syntax: F <address1> <address2> <data sequence> <CR>

The arguments <address1> and <address2> are the start and end addresses, respectively, of the memory area to be filled. <data sequence> specifies the sequence with which the memory is to be filled and consists of two-digit hexadecimal numbers and single ASCII characters. If two consecutive characters in this sequence are legal hexadecimal digits these are interpreted as a hexadecimal number. If one character is specified, separated with at least one space on each side, it is interpreted as a single character and the ASCII value of that character is inserted into the 'fill sequence'. The character '→' (CTRL-Z) has a special meaning in the specified <data sequence>. When the memory area is filled, each memory location which corresponds to the right arrow will be unaffected. The '→' must be separated with at least one space on each side.

Example:

```
F 5000 5010 1011 12 Z B 0 → 00 FF
```

After execution of this command the contents in the memory area 5000H to 5010H will be ('XX' means that the memory cell has not been altered):

Address	Contents
5000	10 11 12 5A 3B 30 XX 00
500B	FF 10 11 12 5A 3B 30 XX
5010	00 XX XX XX XX XX XX XX

Note that the memory beyond address <address2> will never be affected by this command.

COPY (C) Command

This command is used to copy a block of memory from one area to another.

Syntax: C <address1> <address2> <address3> <CR>

<address1> and <address2> are the start and end addresses, respectively, of the block to be copied. <address3> is the start address of the destination.

Example:

```
C 561A 5679 621B
```

This command will copy the contents in the memory area 561AH to 5679H to the memory area 621BH to 627AH.

RELOCATE (L) Command

After moving a part of a machine code program from one memory area to another, some of the address references in the destination area must be adjusted. If this is not done the program will probably not execute in a correct way. The reLocate command has been included to make this process easier.

Syntax:

L [I] <address1> <address2> <address3> <address4> <address5> <CR>

This command will be described by means of an example, but first a short description of the five arguments, <address1> - <address5>, is presented:

- <address1> - The start address of the area where the program was copied FROM.
- <address2> - The end address of the area where the program was copied FROM.
- <address3> - The start address of the area where the program was copied TO.
- <address4> - Lower limit of the range of adjusted memory references.
- <address5> - Upper limit of the range of adjusted memory references.

Example:

Suppose we have the following machine code program located at address 5000H and upwards:

Address	Instruction	Comments
5000	LD A,20	ASCII for space
5002	CALL 0012	Write the ASCII char in Acc
5005	INC A	
5006	JP NZ,5002	Continue until Acc=0
5009	LD DE,5010	Pointer to message
500C	CALL 0015	Write message
500F	RET	Exit
5010	52	The message 'READY'
5011	45	in hex terminated
5012	41	with CR.
5013	44	
5014	59	
5015	0D	

Now we want to move this program to address 6000H. To do this we use the Copy (C) command:

C 5000 5015 6000

and now we have a copy of the above program at address 6000H:

Address	Instruction
6000	LD A,20
6002	CALL 0012
6005	INC A
6006	JP NZ,5002
6009	LD DE,5010
600C	CALL 0015

```

600F      RET
6010      "READY"
6015      OD

```

To make the above program work properly we have to alter the JP instruction at address 6006H to JP NZ,6002 and the message pointer at address 6009H to LD DE,6010. The two CALL instructions must not be altered, because they refer to addresses outside the moved program. To make the above adjustments, the command

```
L I 5000 500F 6000 5000 5015
```

can be used.

The first three addresses coincide with the arguments used in the Copy (C) command, except that the data area containing the message has been left out. The syntax has been chosen in this way because it was believed that this command would be used together with the Copy command in most cases. The last two addresses indicate which range of arguments in the machine code program is to be adjusted.

The program has now been adjusted to

```

6000      LD A,20
6002      CALL 0012
6005      INC A
6006      JP NZ,6002
6009      LD DE,6010
600C      CALL 0015
600F      RET
6010      "READY"
6015      OD

```

The optional 'I' used in the above reLocate command indicates that all 'Load a Register Pair Immediate' instructions should also be adjusted. Hence if the 'I' had not been specified in this command, the instruction at address 6009H would not have been adjusted. Please refer to the end of the description of this command for a list of all instructions that are adjusted by this command.

Here follows a short description of how this command works:

The computer starts searching for instructions that contain absolute memory references at the address specified by the <address3> argument. The search will terminate at address <address3>+<address2>-<address1>. When an instruction which contains an absolute memory reference is found, the computer checks if this memory reference is in the range specified by the <address4> and <address5> arguments. If this is the case, the instruction is adjusted. Thereafter the search continues. If an illegal instruction is encountered during the search, it is treated as a two-byte instruction. Legal instructions are those defined in, for example, the "Z80-Assembly Language Programming Manual".

Some hints:

If the area to be adjusted contains data sections, the result will, in most cases, not be correct. Instead, adjust each area containing machine code programs separately, and skip the data areas.

It is important that <address3> points to the start of a machine code instruction.

No memory location at an address higher than <address3>+<address2>-<address1> will be adjusted and therefore it is important that this expression point to the last byte of the last instruction to be adjusted.

Here follows a list of the Z80 instructions that are adjusted by this command (nn is a two-byte number):

LD A,(nn)
LD (nn),A
LD dd,(nn) where dd is HL, BC, DE, SP, IX, or IY
LD (nn),dd
JP nn
CALL nn
JP cc,nn where cc is NZ, Z, NC, C, PO, PE, P, or M
CALL cc,nn

If the optional 'I' is specified in the command the following instructions are also adjusted:

LD dd,nn where dd is HL, BC, DE, SP, IX, or IY

MEMORY DISPLAY COMMANDS

DISASSEMBLE (D) Command

This command is used to display the contents in memory using standard Z80 mnemonics (as defined in "Z80-Assembly Language Programming Manual"). The contents are also displayed as hexadecimal numbers and interpreted as ASCII characters.

Syntax: D <address1>[<address2>] <CR>

If both <address1> and <address2> are specified, the memory area between these addresses will be disassembled. If only <address1> is specified, the contents from this address and upwards will be disassembled.

The format of the disassembled lines are as follows:

<address> <hex dump> <mnemonic> ;<ASCII>

All numeric arguments are displayed as two- or four-digit hexadecimal numbers. When the relative jump instructions JR and DJNZ are disassembled the absolute address of the destination of the jumps are displayed instead of the relative displacement.

It is possible to add a trailing 'H' to the hexadecimal numbers and the ASCII character interpretation can be disabled. See the Flags (%) command, flags 3 and 4 respectively. How the character interpretation is done when the most significant bit in the byte is set can be manipulated with flags 1 and 2.

To temporarily discontinue the listing, press the SPACE bar. To continue, press it once more. Pressing BREAK aborts the command.

TABULATE (T) Command

This command is used to display the contents in memory as hexadecimal numbers. The contents interpreted as ASCII characters are also displayed.

Syntax: T <address1>[<address2>] <CR>

If both <address1> and <address2> are specified, the contents in the memory between these addresses will be displayed. If only <address1> is specified the contents in this memory location and upwards will be displayed.

Each line of the listing consist of an address followed by eight two-digit hexadecimal numbers, representing the contents in memory, and is finished with eight ASCII characters representing the character interpretation.

The character interpretation can be manipulated in the same way as in the Disassemble (D) command; refer to this command for more information.

Pressing the SPACE bar will discontinue and resume the listing. Pressing BREAK aborts the command.

FILE READ AND WRITE COMMANDS

These commands are only capable of writing, reading, and verifying files such as machine language or Basic programs. For example, it is not possible to read text files created with the Text Editor SP-2202 or Basic Data files.

WRITE (W) Command

This command is used to write the contents of a part of the memory to a cassette file.

Syntax:

W [<type>] <address1> <address2> <address3> [<filename>] <CR>

<address1> is the starting address of the memory area to be saved and <address2> the end address. <address3> is the auto start address, i.e. where the execution should start after loading the file from the monitor.

<type> is the filetype and it should be specified as a two-digit hexadecimal number. If it is not present, it is assumed to be 01H.

If the filename is specified and it is longer than 16 characters, only the first 16 characters are used.

When the CR key is pressed, the computer displays 'RECORD.PLAY'. Just press the Record and Play buttons on the tape recorder and the contents of the selected memory area will be saved to tape.

The command is aborted if the BREAK key is pressed.

READ (R) Command

This command is used to load a cassette file or to read a file header. The command can be entered in four different ways.

Syntax: i) R <CR> (Read)
 ii) R G <CR> (Read and Go)
 iii) R H <CR> (Read Header)
 iv) R <address> <CR>

i) R (Read)

When this command is entered, the computer will display the message 'PLAY'. When the Play button on the cassette recorder has been pressed, the computer will begin searching for a file. When one is found the following message appears on the screen:

Loading <filename>
<type> <address1>-<address2> <address3>

The second line contains the following information:

<type> - the file type
<address1> - the starting address of the memory area which the file is loaded to

- <address2> - end address of the memory area which the file is loaded to
<address3> - auto start address of the program, i.e. the address where the execution will start if the program is loaded from the monitor or when the command R G is used (see below)

When the loading is completed the computer returns to the command level. No auto start is done.

ii) R G (Read and Go)

This is identical to the R version of this command except that when the loading is completed, an autostart will be done to the loaded program. If the autostart address is less than 0100H, no autostart is done.

iii) R H (Read Header)

This version is used to read and display the file header of the first file found, without loading the file.
When a file is found the computer displays:

```
Found <filename>  
<type> <address1>-<address2> <address3>
```

and returns to the command level. The second line contains the same information as that displayed when the R version of this command is used.

iv) R <address>

This version is identical to the R version of this command, except that the file will be loaded at address <address> and upwards regardless of the starting address in the file header. Note that the start and end addresses displayed do not reflect where the file is actually loaded. Instead, they are the addresses found in the file header.

The BREAK key can be used to abort this command.

VERIFY (V) Command

This command is used to verify that a file saved with the Write (W) command has been correctly written to tape.

Syntax: V <CR>

When the command is executed the computer displays 'PLAY'. Press the Play button and the computer starts searching for a file. When one is found the following appears on the screen:

```
Verifying <filename>  
<type> <address1>-<address2> <address3>
```

(Please refer to the Read (R) command for an explanation of the second line.)

Now the computer compares the contents in the memory with the data

recorded on the tape. If this is identical, the message 'Ok' will appear. If not, 'Data Error' will be displayed, and this means that the file is defective or that the contents in the memory has been altered since the data were written to tape.
To abort the command, press the BREAK key.

SEARCH AND COMPARE COMMANDS

SEARCH (S) Command

This command searches the memory for a specified data sequence.

Syntax: S <address1> <address2> <data sequence> <CR>

<address1> and <address2> are the start and end addresses, respectively, of the memory area to be searched. <data sequence> is the data sequence to be searched for, and it must be entered in the same manner as in the Fill (F) command. Please refer to that command for further details.

As in the Fill (F) command, a right arrow ('→') in the <data sequence> has a special meaning. In the present case, it is used to match any value.

Example:

Suppose we want to find all the locations in the monitor ROM where a call is made to addresses in the range 0000H - 00FFH. The code for the CALL instruction is CDH, and it is followed by the low and high byte of the called address. Therefore we are looking for sequences of the form CD XX 00, where XX can take any value in the range 00H - FFH. The following command will do the job (note that the monitor resides between addresses 0000H and 00FFH):

```
S 0000 00FF CD → 00
```

All addresses where the specified sequence is found are now displayed.

To discontinue and resume the display, use the SPACE bar. To abort the command, press the BREAK key.

COMPARE MEMORY BLOCK (Y) Command

This command is used to compare the contents of two memory blocks.

Syntax: Y <address1> <address2> <address3> <CR>

<address1> and <address2> specify the start and end addresses, respectively, of one memory block. <address3> is the start address of the other block.

After entering the above command the computer will compare the contents in the memory cell at address <address1> with the contents at address <address3>. Thereafter the contents in address <address1>+1 and <address3>+1 are compared and so forth until the terminating address <address2> is reached. All addresses where the memory contents differ will be displayed. The displayed addresses always are in the range <address1> - <address2>.

Example:

Suppose that the two blocks at addresses 3000H and 4000H contain the following:

Address	3000	3001	3002	3003	3004
Content	01	02	03	04	05

Address	4000	4001	4002	4003	4004
Content	01	9F	03	04	10

After executing the command

Y 3000 3004 4000

the following will be displayed:

3001 3004

The SPACE bar can be used to discontinue and resume the listing. Pressing BREAK aborts the command.

DEBUG COMMANDS

These commands can be used to debug machine code programs. To make this possible, PA-MON always holds the current state of all the Z80 CPU registers for the program being debugged in an internal memory area. This area will be referred to as the INTERNAL REGISTERS in the sequel.

ALTER REGISTER (A) Command

This command is used to alter the contents of PA-MON's internal registers. Three versions of this command exists.

Syntax: i) A <CR>
 ii) A X <CR>
 iii) A S <CR>

The first version (i) is used to alter the contents of the Main Registers B, C, D, E, H, L, A (the accumulator), and F (the flags). When this command is entered, the following will be displayed on the screen:

```
B C  D E  H L  A  SZ H PNC
dddd eeee ffff gg bbbbbbbb
```

(dddd, eeee, and ffff represent four-digit hexadecimal numbers and gg a two-digit hexadecimal number. bbbbbbbb is a sequence of eight zeros and ones.)

The cursor is positioned on the first digit of dddd and it is now possible to alter the contents of the different registers; just rewrite the old values with the desired ones. Note that the cursor keys can be used. When the modifications have been done, press the CR key and the new values will be entered into PA-MON's internal registers.

The flags (F-register) are the bbbbbbbb sequence. A zero means that the flag is reset and a one that it is set.

The different flags are as follows:

```
S = Sign flag
Z = Zero flag
H = Half-carry flag
P = Parity/overflow flag
N = add/subtract flag
C = Carry flag
```

The remaining two bits (the third and fifth from the left) are not used.

The second version (ii) of this command is used to exchange the contents in the Main Registers and the Alternative Registers, B', C', D', E', H', L', A', and F'.

To alter the values of the Alternative Registers, first use the A X command, then modify the contents with the A command and finish with another A X command.

The third version (iii) is used to alter the contents in the Special Purpose Register set, PC (the Program Counter), SP (the Stack Pointer),

IX, IY, I (the Interrupt vector), and R (the Refresh register). When this version of the command is executed, the following will be displayed on the screen:

```
PC   SP   IX   IY   I   R
aaaa bbbb cccc dddd ee ff
```

The modifications of the contents are now made in the same manner as in the A version of the command.

Upon start of PA-MON all registers and flags are set to zero, except the Stack Pointer (SP) which is set to 10F0H.

DISPLAY REGISTER (X) Command

This command is used to display the contents of all the internal registers.

Syntax: X <CR>

The contents are displayed as follows:

```
B C D E H L A SZ H PNC
dddd eeee ffff gg bbbbbbbb (1)
hhhh jjjj kkkk ll cccccccc (2)
PC SP IX IY I R
mmmm nnnn oooo pppp qq rr (3)
```

(dddd, eeee and so on are four-digit hexadecimal numbers and gg, kk, and so on are two-digit hexadecimal numbers; bbbbbbbb and cccccccc are sequences consisting of eight zeros and ones)

The row marked (1) is the Main Registers, (2) is the Alternative Registers and (3) is the Special Purpose Registers. Please refer to the Alter Register (A) command for an explanation of the different register names.

BREAKPOINT (B) Command

This command is used to set breakpoints in a machine language program.

Syntax: B [<address> [<count>]] <CR>

<address> is the address of the breakpoint and <count> is a count which specifies how many times the instruction (at address <address>) is to be executed before the breakpoint is activated. For example if <count>=01 a break will occur the first time the instruction at the breakpoint address is reached and if <count>=02 a break will occur the second time. If <count> is specified it must be entered as a two-digit hexadecimal number, and if not it is assumed to be 01.

To remove a breakpoint, set <count> to 00. To remove all breakpoints, see the Clear Breakpoints (K) command.

If only a B is entered, all breakpoints presently set will be displayed together with their count values.

The maximum number of breakpoints that can be set is 8. If the user tries to set more than 8 breakpoints the error message 'Over' will

appear.

When a breakpoint is set, the memory contents at the specified address will be unaffected. The only time the memory contents will be changed is when the Go (G) command is used. Please refer to that command for further details.

The breakpoint count value, <count>, will always be decremented every time an instruction is executed at the corresponding address, and when the count reaches zero the breakpoint will be activated. After a break has occurred the corresponding breakpoint will be cleared.

Remember that it is important to set the breakpoints at the first byte of a machine code instruction consisting of more than one byte. Otherwise the breakpoint will never be activated, and when the Go (G) command is used strange things may happen.

CLEAR BREAKPOINTS (K) Command

This command is used to remove all breakpoints presently set.

Syntax: K <CR>

Note that this command only clears an internal table (in PA-MON) holding the breakpoint addresses and their corresponding count values.

LIST USER STACK (/) Command

This command is used to display a specified number of 2-byte values on top of the user stack.

Syntax: / [<count>] <CR>

<count> is a four-digit hexadecimal number representing the number of values to be displayed. If <count> is not specified, the top seven values of the stack will be displayed (one row on the screen).

Example:

Suppose that the following machine code program exists at address 5000H and upwards:

```
5000 LD SP,5000
5003 LD HL,1234
5006 PUSH HL
5007 .....
```

After the program between address 5000H and 5006H has been executed with, for example, the Execute (E) command, the Display Register (X) command will show that PC=5007H and SP=4FFEh.

Now, if the command

/ 0003

is entered, then the following line is displayed:

```
1234 0031 2150
```

1234 is the value pushed on to the stack by the PUSH HL instruction. The last two values reflect the contents in the memory area 5000H to

5003H and are not a part of the user stack.

The SPACE bar can be used to discontinue and resume the listing. The BREAK key aborts the command.

EXECUTE (E) Command

This command uses a Z80 simulator to execute a machine code program. The simulator makes it possible to single-step through a machine code program, without having to set a breakpoint at every new instruction to be executed. It is also possible to single-step programs which reside in ROM or PROM.

Syntax: E [-] [<count>] <CR>

<count> is the number of machine code instructions to be executed before the command terminates and it must be entered as a four-digit hexadecimal number. If no <count> is specified, the value 0001 is used. If <count> = 0000 the execution will continue until the BREAK key is pressed.

The contents of the internal registers define the state of the Z80 simulator. This means that, for example, the Program Counter (PC) always points to the next instruction to be executed.

Before a new instruction is executed, the contents of all the registers are displayed together with a line showing the next instruction to be executed (this line can be suppressed, see the Flags (%) command, flag 5). The contents will also be displayed when the number of instructions specified by <count> have been executed.

Breakpoints are recognized, and when one is encountered, the execution will terminate and the contents of the registers will be displayed. Note that the instruction where it is set, will not be executed.

If the optional Minus ('-') is specified, all subroutines will be executed in real time, not simulated. This can be useful if, for example, a time consuming subroutine, which is known to work properly, is going to be executed. Note that PA-MON has no control of the execution when the subroutine is executed, and if the subroutine does not return to the instruction immediately following the CALL (or RST) instruction, control will be lost.

The execution will be temporarily disabled if the SPACE bar is pressed; to resume, press the SPACE bar once more. The BREAK key will terminate the execution and the computer returns to the command level.

Some notes:

Illegal Z80 instructions will normally not be executed (see the Flags (%) command, flag 6), and if one is encountered the illegal instruction will be displayed and the execution terminate.

No check is done whether the program alters the contents in write protected areas of the memory during execution (see the Write Protect Off (@) command).

Execution of programs using interrupts should be avoided!

The machine code instruction HALT, hexadecimal 76H, is never executed.

UNTRACED EXECUTE (U) Command

This command is identical to the Execute (E) command, except that the register contents are not displayed before the execution of a new machine code instruction.

Syntax: U [-] [<count>] <CR>

The only time the register contents are displayed is when the count reaches zero or a breakpoint is encountered.

GO (G) Command

This command is used to execute a machine code program in real time, using breakpoints.

Syntax: G <address> <CR>

<address> is the starting address of the execution.

When this command is entered the execution address <address> will replace the value of the Program Counter (PC) in the internal registers. Next, at all addresses where a breakpoint is set, a RST 38 (one-byte equivalent to CALL 0038) instruction will be inserted. This makes it possible for PA-MON to get into control when a breakpoint is encountered. Finally all the Z80 CPU registers are loaded with the contents of the internal registers and the execution starts.

When a breakpoint is encountered, the contents of all the CPU registers are saved in the internal registers. Thereafter all the RST 38 instructions are removed, and finally the contents of the registers are displayed.

If the executing program jumps to address 0038H (or 1038H) from a point where no breakpoint is set, the error message '38H?' will be displayed. The same message appears if an interrupt generates a jump to this address.

JUMP TO SUBROUTINE (J) Command

This command is used to jump to a specified memory location and execute the machine code program there.

Syntax: J <address> <CR>

where <address> is the desired jump address.

To return to PA-MON from the executing program, a RET instruction, hexadecimal C9H, can be used. Another way is to force a jump to the warm entry point of PA-MON (see the Where Am I (?) command).

Before the execution at address <address> begins, all registers of the Z80 CPU, except the Program Counter (PC) and the Stack Pointer (SP), will be set to the values currently in the internal registers of PA-MON.

Note that breakpoints will have no effect on the execution.

SPECIAL COMMANDS

RELOCATE PA-MON (\$) Command

This command is used to relocate PA-MON to another area of the memory.

Syntax: \$ <address> <CR>

where <address> is the desired starting address of the new version of PA-MON.

After relocation of PA-MON, a jump to address <address> will be done (i.e a cold start of the new version). Note that the old version will be left unaffected.

The Where Am I (?) command can be used to find out where in memory PA-MON is located at the moment.

WHERE AM I (?) Command

This command is used to find out where in memory PA-MON resides at the moment.

Syntax: ? <CR>

After entering this command the computer will display:

<address1>-<address2> <address3>

where <address1> is the address to the first memory location occupied by PA-MON and <address2> the last. <address1> is also the Cold Start of PA-MON, which means that after a jump to this address all internal registers, breakpoints, flags, and so on will be cleared.

<address3> is the Warm Start address and can be used to jump back to PA-MON without clearing any of PA-MON's internal variables.

COMPUTE (=) Command

This command is used to compute the sum and difference between two four-digit hexadecimal numbers.

Syntax: = <number1> <number2> <CR>

<number1> and <number2> are four-digit hexadecimal numbers.

Execution of this command will cause the sum (<number1>+<number2>) and the difference (<number1>-<number2>) to be displayed.

Examples:

```
*= 2125 2000
4125 0125
*=FFFF 0002
0001 FFFD
*= 0001 0003
0004 FFFE
```

The first number displayed is the sum and the second the difference.

PORTS (P) Command

This command is used to read from or write data to one of the 256 Z80 input/output ports.

Syntax: P <port> [<byte>] <CR>

<port> is a two-digit hexadecimal number representing the port number and <byte> is a two-digit hexadecimal number or a single character. If <byte> is not specified, the contents in port <port> will be read and displayed on the screen (as a hexadecimal number). If <byte> is specified, the value <byte> will be written to port <port>. If a single character is used, the ASCII code for that character will be written.

COMMENT (;) Command

This command is used to write comments on the screen.

Syntax: ; <any text> <CR>

The command does nothing and it is included to make it possible to write comments without getting error messages. This can be useful if, for example, the printer is enabled (see the Printer On/Off (#) Command).

FLAGS (%) Command

This command is used to display and/or change seven different flags (described below), which are used by PA-MON to decide how to act in some situations.

Syntax: % <CR>

When this command is executed, the flags will be displayed as a sequence of eight zeros and ones. The cursor appears on the first flag and it is now possible to alter one or more of these, just rewrite the old values with the desired ones. Note that the cursor keys can be used. When the CR key is pressed the new flags will be saved by PA-MON and used in the future.

If no flag is to be changed, just press the CR key and the computer returns to the command level.

Below, in the description of the different flags, it is assumed that they are numbered from 1 to 8. Flag number 1 is displayed to the left and flag 8 to the right.

A flag is set if it is one and reset if it is zero.

Flag 1: If this flag is set, the most significant bit will always be reset before the memory contents are displayed as ASCII characters. This flag affects the D and T commands and upon start of PA-MON it is reset.

- Flag 2: If this flag is set, all bytes with the most significant bit set will be ones-complemented before the ASCII character interpretation is displayed. Note that flag 1 has to be reset, otherwise this flag will have no effect. This flag only affects the D and T commands and upon start of PA-MON it is reset.
- Flag 3: If this flag is set, the ASCII character interpretation will be displayed when the D and T commands are used. Upon start it is set.
- Flag 4: If this flag is set, everything that is entered from the keyboard is also copied to the printer (if it is enabled). Upon start this flag is set.
- Flag 5: If this flag is set, a disassembled line with the next machine code instruction to be executed is displayed together with the register contents in the E, G, and U commands. Upon start of PA-MON this flag is set.
- Flag 6: If this flag is set, illegal Z80 machine code instructions will not be executed when the E and U commands are used. Instead the execution will stop and a disassembled line with the illegal instruction will be displayed. This flag can also affect the G command, but only if a breakpoint is set at an illegal instruction. Upon start this flag is set.
- Flag 7: If this flag is set a trailing 'H' will be added to the hexadecimal numbers when the D command is used. Upon start this flag is reset.
- Flag 8: Not used.

PRINTER ON/OFF (#) Command

This command is used to turn the printer on or off.

Syntax: # <CR>

The first time this command is executed, the printer will be enabled. Entering the command once more disables the printer, and so forth. When the printer is enabled, all that is written on the screen will be copied to the printer, also the commands entered from the keyboard. It is possible to disable the copy of the entered commands, see the Flags (%) command (flag 4).

When the Modify (M) command is used with the printer enabled (and flag 4 set), every line on the screen will be printed twice. First the line before the modification and thereafter the line after the modification.

If no printer is connected, the power is not turned on, or some other error occurs, the message "Printer Error" will be displayed, and the printer will automatically be disabled.

Note that this command will work properly only if a Sharp printer is used.

WRITE PROTECTION OFF (@) Command

Normally the memory area occupied by PA-MON, and also the memory areas 0000H - 11FFH and TOPRAM - FFFFH, are write protected. TOPRAM is the first address where no RAM is installed (or if there are 48 kBytes of RAM, D000H). This address is determined by PA-MON when a cold start is done (the time delay between the display of the header and when the prompt appears). Note that the contents of the memory area outside PA-MON are not altered during this process. This command is used to disable the write protection.

Syntax: @ <CR>

Execution of this command will cause the prompt to change to a '@' and the write protection to be disabled. To enable the write protection, see the Write Protection On (*) command.

The following commands are affected by the write protection:

C, F, L, M, R, Z, and \$.

Upon start of PA-MON, the protection will be enabled.

WRITE PROTECTION ON (*) Command

This command enables the write protection (please refer to the Write Protection Off (@) command for further details).

Syntax: * <CR>

When the command is executed the prompt changes to a '*'.

If the user tries to alter an area of memory which is protected, the message 'Protected' will appear and the computer returns to the command level.

MONITOR (!) Command

This command is used to exit PA-MON and return to the monitor.

Syntax: ! <CR>

APPENDIX A:

COMMAND SUMMARY

In the following summary, the words '2-digit' and '4-digit' are used to indicate that an argument should be entered as a two-digit or four-digit hexadecimal number, respectively.

All commands are terminated with <CR>.

Whenever relevant the BREAK key can be used to abort the command and the SPACE bar to discontinue and resume the listing.

MODIFY:

Modify memory using hexadecimal numbers and single characters.

Syntax: M [I] <address>

<address> - start address of the modification (4-digit)

Z80 MNEMONIC:

Enter machine code instructions into memory using standard Z80 mnemonics.

Syntax: Z <address>

<address> - start address of where the instructions are to be entered (4-digit)

FILL:

Fill memory with a data sequence.

Syntax: F <address1> <address2> <data sequence>

<address1> - starting address of the area to be filled (4-digit)

<address2> - end address of the area to be filled (4-digit)

<data sequence> - data sequence to fill memory

COPY:

Copy a memory block from one part of the memory to another.

Syntax: C <address1> <address2> <address3>

<address1> - start address of the area to be copied (4-digit)

<address2> - end address of the area to be copied (4-digit)

<address3> - start address of the destination area (4-digit)

BREAKPOINT:

Set a breakpoint or display all currently set breakpoints.

Syntax: B [<address>[<count>]]

<address> - breakpoint address (4-digit)

<count> - breakpoint count, default 01H (2-digit)

CLEAR BREAKPOINTS:

Remove all breakpoints currently set.

Syntax: K

LIST USER STACK:

Display the top of the user stack.

Syntax: / [<count>]

<count> - number of two-byte numbers to be displayed, default 0007H (4-digit)

EXECUTE:

Execute a machine code program, using a Z80 simulator, and display the contents of all the registers before the execution of every new instruction.

Syntax: E [-] [<count>]

<count> - number of instructions to be executed, default 0001H (4-digit)

UNTRACED EXECUTE:

Execute a machine code program, using a Z80 simulator, without displaying the register contents.

Syntax: U [-] [<count>]

<count> - number of instructions to be executed, default 0001H (4-digit)

GO:

Execute a machine code program (in real time) using optional breakpoints.

Syntax: G <address>

<address> - start address of the execution (4-digit)

READ:

Read a cassette file into memory.

Syntax: R	(Read)
R G	(Read and Go)
R H	(Read Header)
R <address>	

<address> - start address of the memory area where the file is to be loaded (4-digit)

VERIFY:

Verify a cassette file.

Syntax: V

SEARCH:

Search the memory for a specified sequence.

Syntax: S <address1> <address2> <data sequence>

<address1> - start address of the area to be searched (4-digit)

<address2> - end address of the area to be searched (4-digit)

<data sequence> - sequence to be searched for

COMPARE MEMORY BLOCK:

Compare the contents of two different memory blocks.

Syntax: Y <address1> <address2> <address3>

<address1> - start address of one of the memory blocks (4-digit)

<address2> - end address of this block (4-digit)

<address3> - start address of the other block (4-digit)

ALTER REGISTER:

Alter the contents of the internal registers.

Syntax: A	(alter main registers)
A X	(exchange main registers and alternative registers)
A S	(alter special purpose registers)

DISPLAY REGISTER:

Display the contents of the internal registers.

Syntax: X

RELOCATE:

Relocate a Z80 machine code program.

Syntax: L [I] <address1> <address2> <address3> <address4> <address5>

- <address1> - start address of the area where the program was copied from (4-digit)
- <address2> - end address of the area where the program was copied from (4-digit)
- <address3> - start address of the area where the program was copied to (4-digit)
- <address4> - lower limit of the range of adjusted memory references (4-digit)
- <address5> - upper limit of the range of adjusted memory references (4-digit)

DISASSEMBLE:

Display the contents of memory using standard Z80 mnemonics.

Syntax: D <address1>[<address2>]

- <address1> - start address of the memory area to be disassembled (4-digit)
- <address2> - end address of the memory area to be disassembled, default FFFFH (4-digit)

TABULATE:

Display the contents of memory using hexadecimal numbers.

Syntax: T <address1>[<address2>]

- <address1> - start address of the memory area to be displayed (4-digit)
- <address2> - end address of the memory area to be displayed, default FFFFH (4-digit)

WRITE:

Write the contents of memory to a cassette file.

Syntax: W [<type>] <address1> <address2> <address3>[<filename>]

- <type> - file type, default 01H (2-digit)
- <address1> - start address of the memory area to be written (4-digit)
- <address2> - end address of the memory area to be written (4-digit)
- <address3> - auto start address (4-digit)
- <filename> - filename, default no filename

JUMP TO SUBROUTINE:

Execute a machine code subroutine and return to PA-MON when ready.

Syntax: J <address>

<address> - start address of the execution (4-digit)

RELOCATE PA-MON:

Relocate PA-MON to another memory area.

Syntax: \$ <address>

<address> - start address of the new version of PA-MON (4-digit)

WHERE AM I:

Display where in memory PA-MON resides at the moment.

Syntax: ?

COMPUTE:

Compute the sum and difference of two hexadecimal numbers.

Syntax: = <number1> <number2>

<number1> - hexadecimal number (4-digit)

<number2> - hexadecimal number (4-digit)

PORTS:

Read from or write to a Z80 input/output port.

Syntax: P <port>[<byte>]

<port> - port to be written to or to be read (2-digit)

<byte> - data to be written

COMMENT:

No action.

Syntax: ; <any text>

FLAGS:

Display and/or alter the flags which tells PA-MON how to act in different situations.

Syntax: %

PRINTER ON/OFF:

Turn the printer on or off.

Syntax: #

WRITE PROTECTION OFF:

Disable the memory write protection.

Syntax: @

WRITE PROTECTION ON:

Enable the memory write protection.

Syntax: *

MONITOR:

Exit PA-MON and return to the monitor.

Syntax: !

APPENDIX B:

MEMORY LOCATIONS AFFECTED BY PA-MON

In addition to the memory area occupied by PA-MON (displayed by the Where Am I (?) command), the following memory locations will be affected by PA-MON:

Address	Remark
103BH-103AH	Only used by the Go (G) command.
10FOH-116FH	Used by the File Write and Read commands. The Write (W) command only affects 10FOH-1107H.
1170H-11A2H	Used by I/O routines.

APPENDIX C:

USING PA-MON TOGETHER WITH OTHER SOFTWARE

When a cold start of a machine language program, such as the Basic SA-5510 or Sharp Systems Program, is done, the contents of the memory will be destroyed during the initialization. A method to overcome this problem and make it possible for PA-MON to be co-resident with such software will be described below:

Move PA-MON to the top of the memory, using the Relocate PA-MON (\$) command. For a 48 KByte machine, the new starting address can be chosen, for example, to BCOOH.

Load the desired program, using the Read (R) command.

Use the Modify (M) command to alter the contents in that memory location which is specified below. Before the modification, the contents is DOH and this should be altered to the high byte of the address of the first memory location to be unaffected during the initialization. In our example, the value BCH (most significant byte of BCOOH) can be used.

The Write (W) command can then be used to save the modified version.

Memory locations which have to be altered:

Program	Address
Basic SA-5510	1204H
Basic SP-5025	120BH
Xtal Basic 2.2S	2DEDH
Text Editor SP-2202	2223H
Relocatable Loader SP-2301	1222H
Symbolic Debugger SP-2401	1233H

SHARPSOFT

Sharpsoft Ltd., 86-90 Paul Street, London EC2A 4NE Tel: 01-739 8558