zen

Z80 Editor/Assembler/Debugger

SHARP
mz700

Kuma Computers Limited
Unit 12, Horseshoe Park
Horseshoe Road, Pangbourne, Berkshire, RG8 7JW
Tel: (07357) 4335 Telex: 849462 TELFAC.KUM

INTRODUCTION

Thank you for buying this copy of MZ-700 ZEN. Zen consists of an editor, a Z80 assembler, and an symbolic object debugger. The editor lets you create and modify a source file consisting of statements in the Z80 assembly language. The assembler translates these into object code in memory, and the debugger allows you to execute them in a controlled manner. Since the debugger can access the symbol table, it is possible to debug code easily, refering to symbols defined at assembly time.

The advantages of assembly programming over higher level programming are flexibility, speed, and compactness.

The disadvantage is that you have to be smarter!

STARTING UP

1) Type LOAD followed by the CR key (hereafter <CR>).
2) Insert Zen cassette, rewind if necessary.
3) Press the cassette recorder PLAY key.
4) Wait.

After a few seconds Zen will be found. After about 40 seconds, Zen will auto-run, and the prompt ZEN> will appear.

COMMAND LEVEL

Whenever the prompt ZEN> is displayed, you are at command level. You can select any one of the following commands, which are described in detail later.

Source Edit
Z ... ZAP numeric
E ... ENTER
N ... NEW

Source Position
D ... DOWN numeric
U ... UP numeric
T ... TARGET numeric
L ... LOCATE string
P ... PRINT numeric
B ... BOTTOM

Source Global
A ... ASSEMBLE
H ... HOWBIG
K ... KILL
S ... SYMBOLTABLE character

Tape, etc
R ... READ character
V ... VERIFY character
W ... WRITE character

Object/Debug
C ... COPY
F ... FILL
G ... GOTO numeric
I ... IN numeric
O ... OUT numeric
M ... MODIFY numeric
Q ... QUERY numeric
X ... XAMINE register

? ... EVALUATE numeric

Just typing <CR> will clear the screen.

COMMAND SYNTAX

To select a command type in the command letter, followed by a parameter if necessary, then press <CR>. The type of parameter required by a given command is shown in lower case in the above command table.

The types are:

character ... a single character.
string ..... a sequence of characters.
numeric ..... a number, symbol, or expression, see below.
register .... see XAMINE command.

All the commands which take a parameter will adopt a default value if you fail to supply one. All the commands are explained in detail later on.

If Zen doesn't understand anything you've typed in, it will display the error message HUH?, and return to command level.

USER INPUT

Whenever you are required to input from the keyboard, whether at command level or anywhere else, the following rules apply:

(1) Zen takes no action until you press <CR>.
(2) DEL can be used to backspace.
(3) You cannot backspace past the start of your input.
(4) You cannot type past the end of the video line.
(5) GRPH and ALPHA may be used to change case.
(6) All other characters are data characters.

NUMERICS

A numeric is a decimal, hexadecimal, or octal number; a symbol, or an expression containing numerics and arithmetic operators.

Decimal is the default base for numbers, Hexadecimal numbers are postfixed 'H', and octal are postfixed 'O'. All numbers must begin with a digit 0 to 9, so hexadecimal numbers may require a leading zero in some cases.

Symbols may be used in any place where a literal number may be used. They are only available after assembly.

Characters may also be used. Any single character enclosed in single or double quotes may be used wherever a number may be.

All of the above types may be part of an expression. The infix maths operators are:

+ Addition	/ Division
- Subtraction	& Logical And
* Multiplication	. Logical Or

An expression may be used in any place a simple number may be used. Expressions are evaluated strictly left to right with no precedence ordering. Arithmetic is 16 bit unsigned integer, and overflow is ignored. Elements in an expression need not be delimited, as the maths operators are implied separators.

SOURCE EDIT COMMANDS

Note: Most of the commands described below require a numeric parameter following the command letter, e.g. Z74. This is refered to as "nn" in the text to describe the general operation of the command.

Z - ZAP

Deletes (or "Zaps") nn lines from the source file, starting with the current line.

Z74<CR> would delete 74 lines,

Z<CR> would just delete 1 line.

The command cancels if it hits the end of file, and the message EOF appears. Otherwise the new line is displayed.

E - ENTER

Enters text into the source file. Zen will display a line number as a prompt. Enter a line of text, and type <CR>. Zen will then display the next line number, and invite you to enter another line. To exit from this line, simply type a full stop '.' at the beginning of a line. Text is entered at the current line, as indicated by the line number. The old current line, and all following lines, are moved downwards. Thus ENTER inserts lines before the current line.

N - NEW

This command lets you modify an existing line. The current line number, and line of text is displayed. The cursor is positioned at the end of the line. Change the line (by deleting characters and/or typing extra characters) and then type <CR>, and the new line will replace the old one in the file. If you are at the end of file, EOF will be displayed.

SOURCE POSITION COMMANDS

D - DOWN

Moves you down the file by the specified number of lines, setting a new current line.

D13<CR> moves down 13 lines.

D<CR> moves down one line.

The command cancels at EOF, as in ZAP.

U - UP

Moves the pointer up nn lines. Cancels at SOF (Start of File). (see DOWN).

T - TARGET

Moves you to a specified line, and makes that the current line. If no line specified, moves to line 1, the Top or SOF. Cancels at EOF as in ZAP.

T52<CR> moves to line 52

T<CR> moves to top, line 1

B - BOTTOM

moves to Bottom or EOF. Displays EOF message. Typing B<CR> U20<CR> P22<CR> will display the last few lines of the file.

L - LOCATE
Will locate a string of text in the text file.
LBIT 7,(HL)<CR>
moves to the first line containing the string BIT 7,(HL) and makes it current and displays it. It cancels at EOF as in ZAP. The file is searched from the line AFTER the current line, downwards. you'll see why when you've tried it a few times. There are no restrictions on the string content.

P - PRINT
Displays nn lines of the file on the VDU. beginning with the current line. The last line displayed is made the new current line, thus another PRINT would list the line again.
P22<CR> would display (print) 22 lines
P<CR> would display the current line
Cancels at EOF as in ZAP.

SOURCE GLOBAL COMMANDS

H - HOWBIG
Displays the start of file (SOF), end of file (EOF) and top of memory addresses in hex.

K - KILL
Kills the file. Erases it. Deletes it. (like NEW in BASIC). It is possible to recover from an accidentally KILLed file - see later.

A - ASSEMBLE
Assembles the source file from SOF to the END pseudo op. After typing A<CR> you will be prompted for a list option, these are:

V<CR> List to Video
E<CR> List to External device (e.g. Printer)
<CR> Don't list. This is the fastest.

When selecting E, you are prompted for a name. This will appear at the top of each page of the listing. It is wise to give the program name, and possibly a version number, so you can tell your listings apart at a later stage.

Use the null option to inhibit listing until all syntax errors are eliminated.

The listing process is independant of the object code generation process, which is controlled by the LOAD pseudo op.

S - SYMBOL TABLE

Alphabetically sorts and lists the symbol table built up during the previous assembly. The order is affected only by the first letter of each name, but the process is fast. You will be prompted for a list option: V<CR> or E<CR>. (defaults to E). If E is specified the name defined in the last ASSEMBLE to E is used.

It is possible to generate only part of the symbol table listing, by adding a selector to the command letter:

SK<CR>

would only generate symbols beginning with the letter K.

TAPE STORAGE

Zen can READ, WRITE or VERIFY source or object files. Source files are saved without line numbers, as pure text. Lines are ended by <CR>, code 13. Object files are saved in the standard SHARP format, as type 1 files. Note that Zen maintains full compatability with Zen on the other sharp micros: MZ-80K, MZ-80A and MZ-80B(at "slow" tape rate).

All sharp files have a type number. i.e.:

Object (Machine Code)... 1
MZ-80 BASIC Program 2
MZ-80 BASIC Data 3
Zen Source 4
MZ-700 S-BASIC Data 4
MZ-700 S-BASIC Program . 5

Note that Type 4 files are used by both Zen and S-BASIC. They are incompatable, i.e. you cannot load Zen Source into BASIC, or Basic Data into Zen.

To select object files, add an 'O' to the command. Not adding any letter will select Source files.

The commands are:

R<CR> READ source file	RO<CR> READ object file
V<CR> VERIFY source file	VO<CR> VERIFY object file
W<CR> WRITE source file	WO<CR> WRITE object file

Zen checks the file type, as well as the name, so there is no chance of picking up the wrong file, even if their names are the same.

You can abort from these commands by pressing SHIFT.BREAK. Zen will return to command level.

READ
Upon entry you will be prompted for a file name. If you supply one, then that file will be searched for. If you default, then Zen loads the first file of the correct type that it finds.

Source files are always loaded at EOF, allowing you to merge files together. To load without merging, Kill the old file before reading. After the file has loaded, the new file size is displayed.

Object files are loaded at the load address on the cassette. Object files are never auto-executed. Instead the execute address is placed in the User Program Counter, for use with the GOTO command.

VERIFY
Similar to read, only the file is not loaded. It is checked against memory to ensure that the tape is correct. The VERIFY command must be used immediately after the WRITE command, or not at all. Using the verify command at other times will cause an error.

WRITE
If writing the source file, you'll just be prompted for a name. Then the source file is saved.

To save an object file, more information is required. START, STOP, EXEC, and LOAD addresses will be prompted for. Memory is written from START to STOP inclusive. STOP must be greater than START or you'll get an error message EXEC defines the execution address of the file. The MZ-700 monitor will auto execute any file with an execute address greater than 11FFH. The LOAD address defines where the file should load into memory. For example, you might want to write from 8000H to 83FFH and have to file load back at 1200H

OBJECT COMMANDS

C - COPY
Copies a block of memory from START address to STOP address inclusive, to a DESTINATION address. You are prompted for all three parameters.

F - FILL
Fills a block of memory, from START to STOP inclusive with a DATA constant. You are prompted for all three.

G - GOTO

Executes a user program from a specified address, for debugging and testing.

G4604H<CR> jumps to that address.

GSTART<CR> jumps to the address label START.

G<CR> jumps to the address in the User Program Counter.

You will be prompted for a BREAKPOINT address. Inputting an address sets a breakpoint there, null (just <CR>) sets no breakpoint. A breakpoint is an address at which the execution of the user program will stop, and control returned to Zen, saving all the registers. The breakpoint cannot be set in ROM, and must be the first byte of an instruction. The GOTO command loads the Z80 registers from the User registers (as shown by XAMINE), and finally sets the PC via a JP instruction. Zen uses a RST 38H opcode as a breakpoint. When the Z80 hits this instruction, it pushes the PC onto the stack, and jumps to 38H. The monitor in ROM then jumps to 1038H, where there is a vector to the TRAP handler. All registers are then saved. The breakpoint location is restored to its old value, and the code at 1038H is restored. To continue from a breakpoint, use the G<CR> command. You can set another breakpoint then.

Note: the user is recommended not to use the RST 38H while using Zen Breakpoints.

Note: If you label the start of your program O: you can run it, for testing, by a simple GO<CR>.

I - IN

Reads one of the Z80 I/O ports.

I33<CR> would read port 33.

The data is read and displayed in Hex and Binary. Note that the SHARP MZ-700 uses various I/O ports, and peripheral devices can be connected to others. Due to the fact that some ports will swap memory banks in and out, special care is required to prevent disaster when using these.

O - OUT

Writes to one of the Z80 I/O ports.

O0FEH<CR> would output to port 254

You are prompted for a DATA parameter. Also see IN.

M - MODIFY

Lets you examine and modify memory contents.

M8213H<CR> would commence at that address.

The address and the byte at that address are displayed, both in Hex. If you supply a parameter, then the location is modified, otherwise (null input) Zen steps to the next address. To quit modify, then type a full stop, '.' as a parameter. If you default on the initial address, then the last used address displayed by MODIFY or QUERY (see below) is taken.

Q - QUERY

Displays a block of memory in Hex and Ascii. Sixty four bytes are displayed, in eight rows of eight bytes. Unprintable ascii characters are printed as dots.

X - XAMINE
Displays the user registers in Hex. The main registers are on the first line, and the alternate on the second line. The Flags, SZHVNC, are displayed to the right of the registers when set.
X<CR>
If a register name is appended to the command letter,
e.g. XHL<CR>
then the register is dislayed and can be modified. Any individual 2byte register may be examine/modified by this method. Also A, B, C, D, E, H, and L may in one byte form. In addition, the 2byte Zen registers SOF, EOF, and ML (Memory limit) may be modified by this command.

EVALUATE

? - EVALUATE
evaluates a number or expression, and displays the result on the vdu in decimal and hex. It is ideal for conversion between the two bases, or discovering the value of a symbol, character, the result of a calculation etc.
e.g. ?13
?1200H
?0DH
?START
? "Z"
?'Z'+1
?132+66H/4."@"
?END-START+1
Type these in, and Zen will print the answer in Hex and Decimal. Zen will give an UNDEFINED error if labels (e.g. START or END) are not defined.

ASSEMBLER SYNTAX

Zen expects source statements to be constructed according to the syntax defined in the ZILOG Z80 Assembly Language Programming Manual. A summary of the instructions is given later. The user is advised to obtain a book on Z80 programming if he/she is not already familiar with it.

Each line in the source file is divided conceptually into at most 4 fields, e.g.:

```
PICTURE:LD HL,(CURSOR);Pick up cursor

Label       PICTURE:
Operator    LD
Operand(s)  HL,(CURSOR)
Comment     ;Pick up cursor
```

The fields may be typed in in free format, i.e. they do not need to be positioned at any special part of the line. The correct separators (spaces, commas, colons, semicolons) must be used. Any or all fields may be omitted, although an operand without an operator is meaningless.

COMMENTS
Comments are ignored by the assembler. They are preceded by a semicolon and terminated by the end of line.

OPERATORS
There are 74 generic operators (LD, CALL, JP, etc). In addition Zen provides the pseudo ops defined later.

OPERANDS
The number of operands in a statement depends on the operator. e.g.:

```
NOP ... No operand
CP .... One operand
BIT ... Two operands
JR .... One or two operands
RET ... None or one operand
```

Operands may be:

```
Register names (A, B, HL, IY, etc)
Condition codes (Z, NZ, C, PO, V, NV, etc)
Numerics
```

Numerics
See section on numerics before. Note that $ may be used to access the program counter.

Condition codes
In addition to the Zilog standard set, two extras have been added. The Z80 has a parity/Overflow flag which is set for parity after logical operations, and for overflow after arithmetic operations. Zilog standard PO and PE are ideal for odd and even parity tests, but very misleading when used to test for overflow. The extras V and NV are provided for this purpose. V is identical to PE and NV to PO.

LABELS
A label is a way of marking a statement. Each time you use a JP, CALL, etc; you will need a way of specifying the destination as an operand. Zen allows you to do this using meaningful names. The label is a symbol.

SYMBOLS
A symbol is a name associated with a value. The name can be used instead of stating the value. A symbol is declared to the assembler in one of two ways:

(1) By using the EQU pseudo op (similar to LET in Basic). This allows you to assign your own value to a symbol.
e.g. CR:EQU 13

(2) By placing it at the start of a statement. The assembler will assign the value of the program counter to the symbol. The symbol is being used as a label.

In both cases the symbol must be postfixed with a colon, ":", when declared. A symbol must begin with a letter, but can contain letters and digits after that. Letters must be upper case. Strictly any non special character may be used, providing its ascii code is less than 128. This includes ?, !, =, etc, but not $ and . which are the program counter and OR arithmetic operator. Especially note that Sharp lower case are not allowed.

Symbols may be of any length, but long symbols will be listed in truncated form by the formatting code (see LISTING). Single character symbols, and symbols not beginning with a capital letter, are not listed in the symbol table.

Reserved words, Condition codes and register names, may not be used for symbols.

PSEUDO OPS
These are additional operators which are not part of the Z80 instruction set, but are understood by the assembler. They are used in exactly the same way as normal operators.

```
END ... End assembly .... No operand
DS .... Define storage .. One operand
DW .... Define word ..... One operand
DB .... Define byte(s) .. variable operands
EQU ... Equate .......... One operand
ORG ... Origin .......... One operand
LOAD .. Load memory ..... One operand
```

END
This operator must be used to terminate assembly. Failure to do so will result in an error message and an incomplete assembly.

DS
Skips a number of object locations. Commonly used to reserve space for a stack or text buffer where memory contents don't need to be initialized.

DW
Generates a word (2 bytes) in the object file in Zilog Z80 format, i.e. low byte first, high byte second. This format is used by all Z80 sixteen bit instructions.

DB
Generates byte(s) in the object file. Takes as many operands as desired separated by commas. Each operand may be an expression but clearly must have a value between 0 and 255. It is possible to "chop off" the high byte by ANDing with 255 (&255). In addition to the usual data types, a literal character string may be used:

```
MESSAGE:DB "I'm a SHARP MZ-700"
```

Strings may be of any length, but cannot form part of an expression. Strings may be enclosed in single of double quotes. Do not confuse a string with a single character, which can form part of an expression.

```
MESSAG2:DB "I'm a SHARP MZ-70","0".80H
```

The above example shows a string, with the last character ORed with 80H to set bit 7.

EQU
Assigns a value to a symbol. The operand may be any numeric including a symbol or an expression, but the value must be known to the assembler. Forward referencing is not allowed, and will result in an UNDEFINED message.

ORG
Defines the origin, or assembly address, of the object code. This operand may be used as often as required to provide different sections of code at different addresses. The same restrictions apply to the operand as in EQUate.

LOAD
Lets you load the object code straight into memory as it is produced.
e.g. LOAD 8000H
You may load anywhere in memory, independant of the actual origin. If the LOAD is not specified then no memory is altered in any way. Note that another ORG turns the loading process off, and it needs to be switched on specifically with another LOAD.

ERROR HANDLING

If the assembler encounters an error, then the following will happen:

(1) Assembly terminates.
(2) An error message is displayed.
(3) The incorrect line becomes the editor current line.
(4) The line is displayed.
(5) Zen returns to command level.

You may correct the error, and assemble again. The N command is very useful for this if the error is a simple one just in the one line. It is impossible to make an error which will damage Zen or the Source file. Assembly runs at around four thousand lines a minute, so errors are found very quickly.

The error messages are:

```
UNDEFINED ... You've used an undeclared symbol.
SYMBOL ..... No symbol in EQU, or symbol of zero length.
RESERVED .... You have used a reserved name for a symbol.
FULL ........ The symbol table is full, see later.
DOUBLE SYMBOL A symbol has been declared twice.
EOF ......... There is no END. Zen has reached EOF.
ORG! ........ No origin supplied.
HUH? ........ Zen is completely baffled.
OPERAND ..... Something is wrong with an operand,
          e.g.   LD A,256
                 BIT 9,B
                 LD (DE),C
                 INC AF
Also included are relative jumps & indexing out of range,
                 JR $+999
                 LD B,(IX-187)
```

The assembler will catch all incorrect statements, so don't be afraid to experiment.

ASSEMBLY LISTING

Zen supports two list devices, a 40 character per line (cpl) video display, and an 80 cpl external device, usually a printer. If it is desired to change to, say, an 80 cpl video device, or a 132 cpl external device, then this can be done simply by changing the listing field widths table (Zen listing...COMWIDTH). The external device is set up to be the printer/plotter, or external sharp printer. but this can be changed easily (Zen listing...EXTERN) Zen generates listings a page at a time, video is set to 20 lines per page, and extern to 60. This can be changed easily (Zen listing...PAGE).

The external device has only to recognise two characters, Ascii formfeed (12, 0CH), which is given at the start of each page, and Ascii Carriage Return (13, 0DH), which is given after each line. Spare bytes are left before the external device driver (Zen listing...EXTERN) to allow codes to be modified to suit the printer. In addition the title can be printed double width if the printer can support the option. The correct code should be inserted at PBUFF, and the code for normal at the end of the buffer.

PAUSE CONTROL
When assembly is in progress, you may use the SHIFT key to pause. Pressing either shift key will hold the listing at the END OF PAGE, allowing you to change paper on a sheet fed printer, etc. To restart, press any key except '0', which causes you to Quit assembly, and return to command level.

FORMATTING
The symbol, operand, and comment fields of a statement may be of indefinate length. To produce a neat and readable listing, Zen will truncate these fields if necessary, or pad them with spaces. The object code field is never truncated in any way.

The sizes of fields used are set in COMWIDTH and SYMWIDTH. The first byte is Extern, second is Video. The object code and line numbers take 20 characters at the start of each line. COMWIDTH contains sizes of the remaining line. SYMWIDTH contains symbol field width. This is used both for assembly and symbol table listing. The next line is the operators, 5 is always sufficient. The next is the operands. The last line is the comments. The total of these four, plus 20 should equal the line length.

SYMBOL TABLE OUTPUT
Similar to the assembly listing. On video 3 symbols are listed on each line. On Extern this is changed to four, and the field is extended.

ODDS AND ENDS

SYMBOL TABLE
The symbol table is where symbols are stored with their 16bit value. The table is positioned directly above Zen and underneath the Source file. Sooner or later, you'll want to make it larger. As the start of file pointer is also used as the symbol table upper limit, the Source file needs to be shifted up.
(1) Save the source file, if necessary.
(2) Kill the file.
(3) Use XAMINE to change SOF.
(4) Kill the file again. This sets EOF to SOF.
(5) Assemble, to close the symbol table.
(6) Save Zen. (see below)

SAVING ZEN
(1) Kill the file.
(2) Assemble to close the symbol table.
(3) Write Zen to tape (etc) using WO.
You must save up to the very last byte shown in the listing. Start, Load, and Execute addresses are all 1200H.

To verify - use VO.

MEMORY PROTECTION

The source file cannot grow indefinately. Sooner or later you will run out of memory. There are three commands which extend the source file, ENTER, NEW, and READ. All these commands look ahead and see what will happen if they are executed. If they will exceed the top of memory you will get a message of the form:

xxxx MEMORY FULL

when xxxx is the address in Hex where the file would grow.

Normally the top of user RAM is taken to be the memory limit, however, you can add a limit of your own. Use the XAMINE command to change Zen's ML (memory limit) register. If this is zero, then Zen will take the top of user RAM as a limit. If you set it to a non zero address, then Zen will let the source file grow up to, and including, that address, but no further.

STACK & INTERUPTS

Zen uses the standard stack at 10F0H. This is the one used by the monitor. Zen will run quite happily under interupts, providing enough stack space is available. When executing user programs, Zen sets up a user stack for the program, which is small. The user program should set its own stack as soon as posible.

SOF, EOF, ML

These three "Zen registers" are to be found, under the names SOFP, EOFP, and LIMIT, in the Zen listing. They can be modified by MODIFY if desired. It is easier, and has been encouraged in this manual, to use XAMINE to alter them. Don't get mixed up with these and ZBO registers.

RECOVERY from KILL

When you Kill a file, all Zen does is to set EOF to SOF. The text is still there in memory. Find the last byte in the file (use QUERY, etc). The last byte will be an Ascii CR, following END. Use XAMINE to change EOF to this address plus one. The source file will re-appear.

Good Luck!

EXAMPLE

This is an example of a typical session, starting with loading Zen, and ending by saving a complete program. The program we are developing is a trivial one to print out the character "A" to the screen. The program then returns to the monitor. For testing we breakpoint the code to return to Zen.

```
** MONITOR 1Z-013A **                    Comments
*L↵                                                  User input
* PLAY                                               is Underlined
LOADING ZEN X <-------------- Loading ZEN
ZEN>E↵                                               ↵ is 'CR'
   1 ORG 8000H↵
   2 LOAD↵ <------------------- Enter Program.
   3 ↵
   4 0:CALL 7↵ <--------------- Note: LOAD needs a parameter,
   5 LD A,"A"↵ <--------------- Note: CALL 7 should be CALL 6
   6 CALL 12H;Character out↵
   7 CALL 6↵
   8 E1:JP 0ADH;Back to MON↵
   9 ↵
  10 END↵
  11 .↵ <--------------------- "Dot" out
ZEN>I5↵
   5 0:CALL 7
ZEN>N↵
   5 0:CALL 6;Carriage return↵          Modify line
ZEN>I↵
   1 ORG 8000H
ZEN>P22↵                                List program
   1 ORG 8000H
   2 LOAD
   3
   4 0:CALL 6;Carriage return
   5 LD A,"A"
   6 CALL 12H;Character out
   7 CALL 6
   8 E1:JP 0ADH;Back to MON
   9
  10 END
EOF
ZEN>A↵                          Assemble.
OPTION>↵
OPERAND <------------------- Error!!
   2 LOAD <------------------
ZEN>N↵
   2 LOAD_$↵          Correct the error,
ZEN>A↵                      and re-assemble.  OK.
OPTION>↵
ZEN>GO↵            Run the program,
BKPT>E1↵                    breakpoint at RET

A <------------------------ IT WORKS!!!
```

```
ZEN>X↓                                        Examine registers.
 HL   DE   BC   AF   IR
0000 0000 0000 CD02 004E N        (Exact details may vary)
0000 0000 0000 0000
 IX   IY   SP   PC
0000 0000 13BA 800B
ZEN>@8000H↓                                   "Query" memory.
8000 CD 06 00 3E 41 CD 12 00 /..>A/..
8008 CD 06 00 C3 AD 00 FF FF /../u.nn
8010 FF FF FF FF FF FF FF FF nnnnnnnn
8018 FF FF FF FF FF FF FF FF nnnnnnnn
8020 00 00 00 00 00 00 00 00 ........
8028 00 00 00 00 00 00 00 00 ........
8030 00 00 00 00 00 00 00 00 ........
8038 00 00 00 00 00 00 00 00 ........
ZEN>M8004H↓                                  Modify "A" to "B"
8004 42 CD .  <-- "B"↓ .↓ typed <--          Note: Zen redisplays
ZEN>GO↓                                      the line.
BKPT>E1↓
                Run.
B <-------------- It works, printing "B".

ZEN>I↓
   1 ORG 8000H
ZEN>N↓
   1 ORG 1200H↓     Change to ORG 12ØØH for saving.
ZEN>D↓
   2 LOAD $
ZEN>N↓
   2 LOAD 8000H↓       but still LOAD 8ØØØH.
ZEN>A↓              Assemble.
OPTION>↓
ZEN>?E1↓            Find value of E1
120BH   46:9
ZEN>WO↓
START>8000H↓ ┐
STOP>800BH↓  ┘ <--- Write correct memory locations.
EXEC>0↓ <---------- Load & Exec at label 0
LOAD>0↓ <---------- Load & Exec at label 0
NAME>DISPLAY A↓ <-------------------------- Note: RECORD.PLAY
WRITING DISPLAY A
ZEN>VO↓               Verify program
NAME>↓
FOUND DISPLAY A (TYPE 1)
VERIFYING DISPLAY A
ZEN>  <---------- All OK.
```