ZEN

A Z80 Editor/Assembler

For the Grant Searle “Simple Z80” and RC2014 “Mini”, Micro” and “Classic II”

AVALON SOFTWARE
of England

copyright 1979

12Feb2022 Neal Crook: Re-created from OCR of scan of original document marked “Bill Powell”. Tweaked some layout and spelling. No intentional change to content.

07Mar2022 Neal Crook: Revise to reflect Nascom version. Add appendix on implementation details

21Mar2022 Neal Crook: Describe IN A,(C) bug-fix.

02Mar2023 Phil_G port to RC2014 Mini/Micro/Classic/Grant Searle “Simple Z80”
(some command changes to suit this hardware)

INTRODUCTION

Thank you for buying this copy of ZEN, we feel sure you'll be satisfied with your purchase. If you have any questions regarding ZEN please direct them, in writing, via our distributors.

ZEN is an integrated system for the production of Z80 machine code. Source statements are entered with the Editor. The source file may be assembled at any time and then executed, under control, with the Debugger.

ZEN is unique in that the editor, assembler, your source, the symbol table and the produced object code are all stored in memory which is perfect for simple systems lacking disks.

STARTING UP

ZEN is supplied as .HEX format (Intel Hex).

After loading the image use the monitor Go (G3000 or G9000) command to enter ZEN,
The entry point to ZEN is 3000H (GS56k version) or 9000H (32k version).

The video screen will clear and the command-loop prompt is displayed:

```
Z>
```

ZEN now wants a command.

COMMAND SET

GLOBAL COMMANDS:

H	HOWBIG
C	CLOSE (like 'NEW' in basic)
Q	QUIT to Phil_G monitor
S	SORT (selector parameter)
A	ASSEMBLE

INPUT/OUTPUT COMMANDS:

W	WRITE (lists source in a re-inputtable format by log or copy/paste in Teraterm)
E	Enter (by 'File, Send' or 'paste' from Teraterm)

SOURCE POINTER COMMANDS:

T	TOP
B	BASE/BOTTOM
U	UP (iteration parameter)
D	DOWN (iteration parameter)
P	PRINT (iteration parameter)
L	LOCATE (target string parameter)

SOURCE EDIT COMMANDS:

Z	ZAP (iteration parameter)
E	ENTER (enter source code)
N	NEW (replace current line)

USER INPUT

All commands are available at all times.

All user input to ZEN is via an intermediate text buffer; no action is taken until you hit the ENTER key.

All user input may be edited with the BACKSPACE (Left-arrow) key. While other control keys may have a visible effect they are NOT RECOGNISED by ZEN.

User lines will be restricted to 43 characters by ZEN.

Commands comprise the key letter and, in some cases, an optional parameter. Where more than one parameter is required by a given command you will be explicitly prompted for each parameter rather than entering them at command level.

Numeric parameters may be Decimal (default), Hex('H' postfix) or Octal ('0' postfix).

GLOBAL COMMANDS

H HOWBIG, displays the start and end addresses (SOF, EOF) of the source file in hexadecimal. Nominally 1800H above the ORG but see later explanation

C CLOSE, erases the source file from memory. After a CLOSE the start and end addresses of the file are equal. This is also the state of the file when ZEN is initially entered.

Q QUIT, leaves ZEN and returns to the system monitor. Quit does not close the current file, ie re-entering Zen will keep the previous state

S SORT, alphabetically sorts and displays the symbol table built during the previous assembly. You can add a selector letter to the command letter which will restrict display to symbols beginning with that letter. Upon entry you will be prompted for an output option, these are:
V ... VIDEO
E... EXTERNAL
Output characteristics are identical to those of assembly LIST output, see ASSEMBLY OUTPUT for more detail.

A ASSEMBLE, will assemble the source file from start of file to the END pseudo-op. Upon entry you will be prompted for an output option, these are:
V (List to video).
C (tabulated object format).
Null (just press return, the default option generates no screen output but generates & stores object code. This is the fastest mode and should be used until all source errors are eliminated.

CASSETTE COMMANDS

The original cassette commands have been changed but are still used to save and load ZEN format SOURCE files using the terminal buffer in the same way as for BASC programs.

W WRITE, will write the current source files to the terminal where it can be logged or copy-&-pasted into notepad. This format is without line numbers suitable for re-inputting.

E ENTER, used prior to paste or 'File, Send' from Teraterm or similar.
Use a character delay of (say) 1ms and a linedelay of (say) 300ms.

R READ, has been disabled as its not appropriate to the hardware, use 'E' instead.

SOURCE POINTER COMMANDS

As the Z80 Assembly Language is entirely line orientated the Editor in ZEN is line, rather than character,orientated. ZEN always maintains an internal pointer to the CURRENT LINE in the source file. The following commands all move the source pointer.

T TOP, pointer moves to start of file.

B BASE, pointer moves to end of file.

D DOWN, pointer moves down towards end of file (parameter) lines.

Example.... D37 moves pointer down thirty-seven lines. The default parameter is assigned a value of one, so D is equivalent to D1. The four editor commands which take an iteration parameter all take the same default value.

U UP, pointer moves up (parameter) lines.

P PRINT, displays(parameter) lines on the video display. The last line displayed is the new current line.

L LOCATE, will find an arbitrary target string in the source file.
Example .... LBIT 7, (HL) Moves the pointer to the first line containing that string. The file is searched downwards from the current line. If the string is not located the pointer is at EOF. There are no restrictions on the string content but unfortunately it doesnt locate line numbers!

SOURCE EDIT COMMANDS

Zn ZAP, erases (n) lines from the file, starting with current. Z alone erases one line.

E ENTER, enters text continuously into the file from the terminal keyboard, from a 'paste' or 'File, Send'. Upon entry the line number is displayed as a prompt.

Type in a line of text, terminating with the (ENTER) key. The next line number will then be displayed and so on. To exit from the command simply key a full stop '.' as the first character of the line.

You can enter text anywhere in tho file.Text is inserted at the current line with the old current line, and all following lines, moving downwards. the first character of the line.

You can enter text anywhere in tho file.Text is inserted at the current line with the old current line, and all following lines, moving downwards.

N NEW, lets you replace an old line with a new one. This command is just a more convenient way of performing a ZAP/ENTER sequence.

NOTE Although line numbers are displayed with most commands you are never required to actually enter them yourself. They are there solely as a positional guide and are computed dynamically by ZEN rather than being stored in the file along with the text.

NOTE Should you exceed available memory the MEMORY FULL error message will be displayed. The file will be in a safe state.

THE ASSEMBLER

ZEN expects source statements to be constructed according to the syntax defined in the Zilog Z80 Assembly Language Programming Manual.

Each line of the source file is a statement divided, conceptually, into at most four fields:

```
MESSAGE:LD HL, GREETING;Say hello
^       ^      ^        ^
Label                   Comment
        Operator
               Operand(s)
```

We say conceptually divided because the components of a statement don't have to be positioned into fields. As long as you use the correct separators (spaces, commas, etc.) ZEN accepts statements in free format.

COMMENTS Comments are ignored by the assembler. They are preceded by a semi-colon ‘;' and are terminated by end of line.

OPERATORS There are 74 generic operators (CALL, LD, BIT, etc.).In addition there are the PSEUDO-OPS which are detailed later.

OPERANDS The number of operands in a statement depends upon the operator. Examples:

NOP	No operands	
CP	One operand	
BIT	Two operands	
JR	One or two	JR SYMBOL JR NZ, SYMBOL

Operands may be:

- Register names (A,B,DE, IX,etc.)
- Condition codes (2,NZ,C, M, etc.)
- Numbers

The number group is the most complex. All the following are accepted as numbers:

- ASCII Literals: The assembler will generate the ordinal value of any character enclosed in single or double quotes.
- Numbers: Decimal, hex and octal bases are accepted with decimal the default. Hex numbers are 'H' post-fixed and octal numbers are 'O' post-fixed. Numbers must begin with a digit, a leading zero is sufficient.
- Symbols: These are explained in detail later on.
- Program Counter: This is the internal variable which simulates the run time PC. It's accessed by using $ (dollar).
- In addition all of the preceding data types may be elements of an expression formed using the infix math operators:
    - + Addition
    - - Subtraction
    - * Multiplication
    - / Division
    - & Logical AND
    - . Logical OR

An expression can be used anywhere that a simple number can be used. The following are all valid:

```
LD DE, START*2-782
LD A, ‘P’.80H
JR NC,$-5
```

Expressions are evaluated strictly left to right with no precedence ordering. Arithmetic is unsigned 16 bit integer and overflow will be ignored. Elements in an expression need not be delimited by separators as the math operators are implied separators.

LABELS

A label is a way of marking a statement. Each time you use an operator like JP, CALL, etc. you need a way of specifying the destination as an operand. Assembly language allows you to use a symbolic name as a label.

BASIC is an example of a language without this facility, the line numbers act as labels.

Symbols will be explained in greater detail.

SYMBOLS

A symbol is a name with an associated value, the name is used rather than explicitly stating the value. A symbol's value is declared to the assembler in one of two ways:

1. By placing it at the start of a statement. The assembler assigns the value of the program counter to it.
2. By using the EQU pseudo-op. This allows you to assign your own value to a symbol.
3. Example ...BACKSPACE: EQU 8

Whichever method is used a symbol must be post-fixed with a colon ':'. when declared. A symbol must begin with a letter but may contain letters or numbers after that. Letters may be upper or lower case. ZEN allows symbols of any length although symbols longer than seven characters will affect the listing. The symbol field width is easily changed. There are certain reserved keywords which cannot be used as symbols. These are:

Operator names, register names and condition code names.

Note that all keywords are uppercase, using the same name in lowercase would be perfectly acceptable.

PSEUDO-OPS

These are additional operators which have no equivalent in the Z80 instruction set but are understood by the assembler. They are used in the same way as the normal operators.

END	End assembly	(No operands)
DEFS	Define Storage	(One operand)
DEFW	Define Word	(One operand)
DEFB	Define Byte(s)	(Multiple operands)
EQU	Equate	(One operand)
ORG	Origin	(One operand)
LOAD	Load memory (object code goes here)	(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.

DEFS — Skips a number of object bytes leaving a gap in the code. Commonly used to reserve space for a text buffer, stack, etc., where the object code doesn't need to be defined

DEFW — Generates a word (two bytes) in the object file in reversed order as required by the Z80 sixteen bit instructions.

Example ..... BUFFER:DW VALUE

Would make locations BUFFER equal to VALUE.

DEFB — Generates the value of the operand(s) in the object file. Takes as many operands as desired, separated by commas.

Example ..... DEFB 6,93H,'T'.80H,NEWLINE

Each operand may be an expression but obviously no expression can have a value greater than two hundred and fifty-five decimal.

The program counter will be incremented after every operand as if each were on a separate line.

In addition to the usual data types any operand may be of the type ASCII literal string.

Example ..... MESSAGE: DEFB'STICKY FINGERS', NEWLINE

Strings may be of any length but, unlike single character ASCII literals, may not form part of an expression. A string is formed in the same way as a single character literal, by enclosing in matching quotes. Note that single and double quotes are implied separators like the infix math operators. You may use a quote, of either type, as a literal by using the OPPOSITE type of quote as the delimiters.

EQU Assigns a value to a symbol.

Example ..... NEWLINE: EQU 13

The operand may, as usual, be an expression but there is a restriction on the symbols you may use in the expression. This is because the operand must be capable of immediate resolution.

The value of any symbols used in the expression must already be known to the assembler, forward referenced symbols will result in the UNDEF error flag.

Example ..... NEWLINE: EQU BACKSPACE+5
BACKSPACE: EQU NEWLINE-5

This sequence is ILLEGAL because each symbol is defined in terms of the other.

The 'no forward reference' rule is designed to prevent you making such a mistake inadvertently.

In practise you will probably never encounter such a situation as most EQUATES have simple operands.

ORG Defines the origin of the object file. This operator may be used as often as desired throughout an assembly to produce sections of code at different locations, The operand must conform to the 'no forward reference' rule for obvious reasons.

LOAD Lets you load the object code into memory, at any address, as it is produced. The loading process is entirely independent of the output option specified upon entry to the assembler. If the operator is not used then no memory location outside ZEN will be altered. Note that use of a subsequent ORG operator turns the loading process off, each time you set a new origin you must specifically re-establish the load process.

ASSEMBLER ERROR HANDLING

If the assembler finds an error in the source code 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. The command loop is re-entered

You can now correct the error and assemble again.

ERROR MESSAGES

DOUBLE SYMBOL	You have declared the same symbol more than once.
UNDEF	You have used an undefined symbol.
RSVD	You have used a reserved keyword for a symbol.
SYMBOL	An obligatory symbol is missing (eg with EQU).
FULL	The symbol table is full (see later).
EOF	You have forgotten END and have hit EOF.
ORG	No origin specified.
HUH?	The line doesn't make sense.
OPND	Something wrong with an operand
	Examples ..... LD A, 256 BIT 9,B LD(DE),C
	Also any attempts to index out of range or to jump relative out of range.

The assembler will catch all incorrect statements. The only incorrect statement which will damage ZEN is a LOAD over its own code. If you need to assemble code at an address occupied by Zen, ORG it as required but LOAD it elsewhere, then copy the object code to its final location using the monitor 'C' command.

ASSEMBLY OUTPUT

LIST — ZEN supports two List devices, the video display and an EXTERNAL device such as a printer. You must insert your own external driver, currently the driver just returns.

ZEN generates an ASCII CR, LF at newline time and an ASCII Form feed at new page time. No other control characters are used.

Listings are generated a page at a time with a pause at the end of each page.

Pages are fifteen lines long for the VIDEO option.

EXTENDING THE SYMBOL TABLE

The ST is the area where each symbol is stored together with its sixteen-bit value. The ST is positioned between ZEN and the source file. The SOF is used as the end of table boundary. As supplied the ST is six hundred bytes long. This value can be decreased or increased by moving the source file pointers down or up. Procedure:

KILL the file
Locate the addresses that store SOFP and EOFP
Change BOTH pointers to the new file position
Save the new version.

SAVING ZEN

You can alter ZEN and then save a new version. You must save up to the VERY last byte shown in the listing.

APPENDIX: Hints and kinks

If the assembler terminates after pass 1 and reports “EOF” maybe:

- you forgot to include an END statement
- you edited (EOFP) to point to the CR (0DH) after the END statement (the last used location) - it needs to point to the first *free* location. Everything looks fine but the END statement is not recognised; very confusing!

The ZEN executable image includes its workspace and stack space; it is not ROMable in its normal form. ZEN does not re-initialise its workspace when executed and so there is no “warm start” address (or: warm start and cold start are the same thing).

Neal and Mike spent several happy evenings inspecting the source and adding comments. The annotated version is too large to be memory-resident for assembly, but Neal has the T4 version for reference.

The locations SOFP, EOFP point to the start of and end of the source code, respectively (refer to the listing file for addresses). The symbol table is built in memory between the end of the ZEN executable image and (SOFP). Change (SOFP) to expand the size of the symbol

table. You can load source code into memory by some other mechanism then change (SOFP) and (EOFP) to point to it before starting ZEN. EOFP points to one AFTER the last character of the source code – ie it points to the first free byte.

This version for the Grant Searle Simple Z80 and the RC2014 Mini/Micro/Classic was created from Neal Crook's Nascom version – thanks Neal!
Phil_G