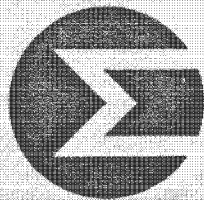


Z80 EDITOR ASSEMBLER PACKAGE FOR THE NASCOM 1 COMPUTER

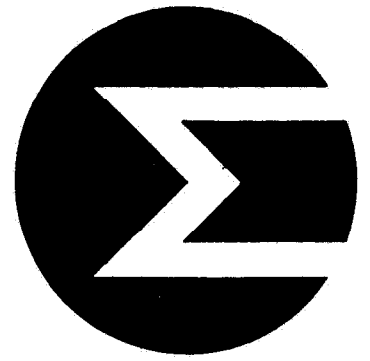
**** NB - IF YOU HAVE LESS THAN 16K OF ADDITIONAL MEMORY, YOU MUST MAKE A MINOR CHANGE TO THE FAST LOADER PROGRAM BEFORE EXECUTING IT. CHANGE THE STACK POINTER VALUE AT LOC 0C51 & 0C52 TO THE TOP OF YOUR MEMORY (LOW BYTE FIRST) WITH THE NASBUG M COMMAND. THEN EXECUTE FROM 0C50 AS DESCRIBED



Published by:
**SIGMA ACCOUNTING
& MANAGEMENT SERVICES LTD**
Software Unit
c/o Nascom Microcomputers
92 Broad Street, Chesham, Bucks.

JANUARY 1979

PRICE £30 00



Information Bulletin

Welcome to ZEAP! You have bought an extremely powerful software product which we hope you will enjoy using. It enables you to edit and assemble Z80 Assembly language programs on the NASCOM 1 computer.

It is important that you complete your software registration form and return it promptly. Only if this form is returned are we able to provide you with updates, patches or other information about ZEAP, or a replacement for a corrupted tape.

ZEAP has been extensively tested, but few packages as powerful as this are completely free of bugs. If you come across anything you believe to be a bug, please complete and return the ZEAP Comment Form in the back of your manual. We will try to take your comments into account on future updates.

Further enhancements to ZEAP are planned including a ROM based version with additional capabilities. These will be announced via your NASCOM dealer and the INMC newsletter.

If you have difficulty, please first check that you are following the correct procedures. The ZEAP manual should be read at least twice. It is a terse document. Similarly other documentation should be carefully studied. Some users have experienced difficulty because they have been accustomed to hand assembly in which abbreviations are followed and they have not used the Z80 Assembly language code in the exact manner defined - eg:

IN A, 2 instead of IN A, (2) (latter is correct)

If you cannot identify the cause of a coding error, study one of the Assembly Language manuals/books listed in the manual.

After you have loaded ZEAP into your NASCOM from the ZEAP tape, you are advised to make a back up copy by dumping ZEAP in NASBUG format (OFOO - IB11) to another tape. Then keep your original ZEAP tape in a clean, dry, dust free and (if possible) controlled temperature environment. Do not store it near mains power points, etc. Please remember that you may make back up copies of ZEAP for your own personal use. You may not make copies for use by others, as gifts, loans, or for sale.

We hope that ZEAP will help you write some good programs easily and quickly.

ZEAP REGISTRATION FORM

COPY NO

5

1) REGISTRATION

Please complete this form and send it to SIGMA in order to enable us to advise you when there are future enhancements to ZEAP. This registration will enable you to obtain updated copies of ZEAP 1.0 for a period of one year from the date the form is received at a nominal copy cost.

2) COPYRIGHT

ZEAP (including the manual, tape and associated documentation) is copyright 1978 by Sigma Accounting & Management Services Ltd. It is supplied for single end use only. It is a condition that purchasers agree:

- 1) Only to use ZEAP on their own NASCOM 1 machine.
- 2) Not to part with ZEAP to anyone, to lend it, to sell it, to dispose of it to any third party, nor to give it to a third party or friend.
- 3) Only to make copies for normal precautionary back up purposes and not to make copies to give away for supply either as a gift or for sale in any shape or form to anyone else.

You may not make copies for use by others as gifts, for loan or for sale.

It is further agreed that copying in any other form or unauthorised disclosure of the software will cause considerable damage to Sigma and that this damage is substantially greater than the value of the software, tapes and the documentation that may be involved.

3) RETURN OF THIS FORM

Two copies of this form are enclosed. Please return one, duly signed, to:

Sigma Accounting & Management Services Ltd
c/o Nascom Microcomputers
92 Broad Street
Chesham
Bucks

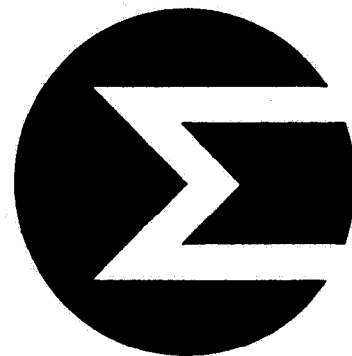
4) DECLARATION

On behalf of the organisation/individual purchasing this copy of ZEAP, I agree to the terms of this form.

Signature _____ Date _____

Name (CAPS PLEASE) _____

Address _____



ZEAP PROGRAM PACK

Contains the following:

1. ZEAP MANUAL

A full description of how to use ZEAP including:

1. A listing of the ZEAP object code
2. A listing of the ZEAP loader object code
3. ZEAP comment forms
4. The Z80 instruction set

2. ZEAP CASSETTE

Individually serial numbered and quality checked, contains copies of ZEAP preceded by a fast loader.

3. LOADING INSTRUCTIONS

A description leaflet explaining precisely how to load ZEAP into your NASCOM.

4. ZEAP REFERENCE CARD

A handy reference card containing the ZEAP editor instructions and assembler options.

5. SOFTWARE REGISTRATION FORM

Ensures that you receive any patches or other useful information about ZEAP.

<u>SECTION</u>	<u>CONTENTS</u>	<u>PAGE</u>
1	INTRODUCTION	4
1.1	AIMS OF ZEAP	5
1.2	COMPARISON WITH THE ZILOG ASSEMBLER	7
1.3	MACHINE REQUIREMENTS	8
2	THE ZEAP EDITOR	9
2.1	EDITOR OPERATION	10
2.2	ZEAP EDITOR COMMANDS	13
3	THE ZEAP ASSEMBLER	24
3.1	ASSEMBLER OPERATION	25
3.2	EXPRESSIONS	26
3.3	SOURCE STATEMENT SYNTAX	28
3.3.1	LABELS	28
3.3.2	INSTRUCTION FORMAT	29
3.3.3	COMMENT LINES	30
3.4	ASSEMBLER OPTIONS	31
3.5	ASSEMBLER DIRECTIVES	32
3.6	ASSEMBLY LISTING	33
3.7	OBJECT GENERATION	34
3.7.1	TAPE OBJECT	34
3.7.2	MEMORY OBJECT	34
APPENDICES		
A	ZEAP OPERATION	35
B	ZEAP ERROR CODES	36
C	ZEAP EDITOR COMMANDS	39
D	ASSEMBLER DIRECTIVES	42
E	ASSEMBLER OPTIONS	43
F	Z80 INSTRUCTION SET	45
G	ZEAP INTERNAL REGISTERS	49
H	ASCII CODE TABLE	50
I	OBJECT CODE LISTING	51
J	COMMENT FORM	56

PREFACE

This manual is laid out in two complementary parts.

Sections 1 to 3 describe the ZEAP package informally and are designed to be read in order.

The appendices following provide a useful reference section, and define all the elements of ZEAP formally, directing the user to the appropriate section in the first half of the manual where more information and examples are to be found.

Those familiar with the workings of micro-computer assemblers and BASIC-type line editors may find it easier to read the appendices first, although this is not recommended to those who do not fully understand the terms used.

The reader should not be dismayed, however. ZEAP is easy to use and yet powerful enough for his requirements.

If information or guidance is required on the Z80 Assembly Language itself, you are advised to consult the Mostek or Zilog Z80 Assembly Language manual. Other publications which may prove helpful include:

The Z80 Microcomputer Handbook by William Barden
(Published by Howard W Sams & Co., Inc.)

Z80 Instruction Handbook by Nat Wadsworth
(Published by Scientific Computer Consultants Inc.)

Z80 Programming for Logic Design by Adam Osborne et al
(Published by Osborne & Associates Inc.)

NOTATION

The following notation is used in this manual:

- E hexadecimal number
- (x) x is optional
- (x)... x is optional and may be repeated
 indefinitely

In general, output from ZEAP is underlined whereas user input is not.

1. INTRODUCTION

ZEAP (Z-80 Editor/Assembler Package) is a memory resident text editor and symbolic assembler designed for use with the NASCOM 1 microcomputer.

The assembler translates mnemonic codes as defined in the Z-80 microcode language into executable machine instructions, allowing user control over memory allocation, and symbolic names for MPU registers and instruction or data addresses. It incorporates comprehensive syntax checking and error message generation, and allows object code to be generated on cassette tape or stored directly in memory.

The editor allows for entry, examination, correction and permanent storage of source programs which are held in memory during editing and assembly.

The memory resident nature of ZEAP allows entry, assembly, testing, correction and re-assembly of source programs without the necessity of using cassette tape at any stage, since editor, assembler, source program and object program may reside in memory simultaneously. This makes ZEAP very easy and quick to use.

1.1 AIMS OF ZEAP

ZEAP was produced with the intention of providing a compact editor/assembler package for the NASCOM 1 microcomputer. The following requirements were laid down during the design of the package:

- * Minimum memory requirements
- * Minimum extra hardware requirements
- * Maximum compatibility with existing assemblers
- * Ability to edit, assemble, execute and then re-edit the program with the minimum use of external storage (eg. cassette tape)
- * Ability to store source programs on cassette tape and then re-load them at a later stage
- * Ability to store more than one source program at a time in memory
- * Maximum use of NASBUG sub-routines
- * Ability to drive an ASCII terminal attached to the UART
- * Ability to generate object code in NASBUG format, to be subsequently loaded using NASBUG's LOAD function

The result is an editor/assembler package requiring 5K bytes of user RAM (1K basic + 4K expansion kit), of which ZEAP uses under 3K bytes, leaving 2K bytes spare for source programs and object code.

The ZEAP editor provides the following functions:

- * Fully dynamic source buffer allocation
- * Insertion, deletion and replacement of lines
- * Context editing of individual lines
- * String searching
- * Automatic line number generation for block entry of source programs
- * Complete resequencing of source program line numbers
- * Loading and dumping of source programs to and from cassette tape
- * Listing of selected source program lines on the screen or on an ASCII terminal

- * Self checking checksum for easy detection of hardware faults or user program malfunction

The ZEAP assembler provides the following functions:

- * Full range of options including control of source listing, object generation and error processing
- * Numbered error messages pin-pointing the exact cause of the error
- * Object generation in NASBUG format onto cassette tape, or directly to memory
- * Formatted source listing on the screen or on an ASCII terminal

The editor, assembler, source program and optional object program may all reside in memory at the same time, enabling maximum ease of entry, assembly, testing, correction and re-assembly of source programs with minimum use of external storage.

The assembler source code follows closely that defined in the ZILOG assembler, the differences being noted in section 1.2.

Editor operation is described in section 2, while the assembler's function is defined in section 3.

It should be noted that because of the commitment to minimum memory requirements, error checking of user input is kept to an absolute minimum. Failure to follow the instructions precisely will thus in some cases result in unpredictable errors or ZEAP itself becoming corrupted. Limits, formats, arguments, etc must be adhered to precisely.

1.2 COMPARISON WITH THE ZILOG ASSEMBLER

The operation of the ZEAP assembler is very similar in most respects to the ZILOG Z80 assembler. The following differences should be noted, however:

- * Expressions may contain only the operators "+" and "-", and no parenthetical grouping is allowed. Expressions may be enclosed in parentheses to represent memory addresses. Evaluation is from left to right. A leading "-" is allowed.
- * Hexadecimal numbers must be preceded by a "f". The "H" suffix form is not supported. The default number base is decimal. Octal and binary numbers are not supported.
- * Labels must begin in the first column at the source line, directly after the single space following the sequence number. Only one label is permitted on a line. The use of a ":" suffix to indicate a label is not supported. Statements without labels must leave the first column blank, except for comments, which may begin in the first column with a ";".
- * The following assembler directives (pseudo-ops) are not supported:

MACRO
ENDM
COND
ENDC
DEFL
END

- * A single ASCII character code may be included in an expression by preceding it with a double quote sign, e.g. "A = f41. This facility replaces the DEFB 's' assembler directive.
- * Fields and/or expressions may be separated by one or more spaces and/or commas. The space and the comma are syntactically equivalent in all contexts within the assembly language.

1.3 MACHINE REQUIREMENTS

ZEAP uses under 3K bytes of memory, not including source program storage. Thus a minimum of 4K bytes of memory is required in addition to the basic NASCOM 1.

With a cassette recorder the user can store source programs on cassette tape for reloading at a later time. The assembler can output NASBUG format object code to tape which can be subsequently loaded using NASBUG's LOAD function.

ZEAP contains routines to drive an ASCII terminal attached to the UART for hard copy or source listings. However, this item is entirely optional and ZEAP will function perfectly without it.

The minimum system is:

- A working basic NASCOM 1
- A television or monitor
- A minimum of 4K bytes of additional memory
- A cassette recorder

2

THE ZEAP EDITOR

The ZEAP editor provides the means by which source programs may be entered, examined and altered by the user.

2.1 EDITOR OPERATION

After ZEAP has been loaded, control is passed to the editor as described in APPENDIX A.

The editor prompt will be displayed (":") indicating that the ZEAP editor is ready to accept editor commands.

The editor is a line editor in which source lines are identified by line numbers (sequence numbers), each line of source code being identified with a unique number. The editor also has powerful context editing capabilities not normally available with this type of editor.

A sequence number may be any decimal number from 1 to 9999. Leading zeros may be omitted. The sequence number is always followed by a single space to separate it from the actual source line, eg.

```
1000 SAMPLE LINE
```

The actual source line is "SAMPLE LINE". The source line itself may of course contain leading spaces, eg.

```
2000 ANOTHER LINE
```

The space after "2000" is the separator, but the next two spaces are part of the source line.

A line of source code may be entered by typing a sequence number, followed by a space, followed by the source line, followed by the New Line key. The editor stores the line of source code in memory and prompts (":") for the next editor command.

The source program is sorted automatically in ascending sequence number order. Thus

```
:20 THIS IS THE THIRD LINE  
:10 THIS IS THE FIRST LINE  
:12 THIS IS THE SECOND LINE  
:  
:
```

would cause the lines to be stored in the order indicated.

Typing a sequence number directly followed by a New Line causes that line to be deleted. Thus

```
:12
```

would cause line 12 to be deleted.

Typing the sequence number of a line which already exists followed by a new source line causes the old line to be replaced by the new line. Thus

```
:20 THIS IS NOW THE SECOND LINE  
:  
:
```

would cause line 20 to be replaced with the indicated text.

Thus all requirements for inserting, deleting and changing lines of source code are provided by the above techniques.

In addition to the above facilities, there are a number of commands for examining and manipulating the source program. To take full advantage of NASBUG's command decoding routines, these commands have been implemented with single letter mnemonic codes. These commands are described below in section 2.2.

All source lines are stored in an area of memory called the EDIT BUFFER. All editor commands operate on the information contained in the Edit Buffer. The size of the source program is limited only by the amount of memory available.

At all times during ZEAP operation the address of first free memory location is displayed in hexadecimal in the top right hand corner of the screen. This address is that of the first location not used by ZEAP for the source program and the symbol table. It is also the default origin address for the assembler. Care must be taken that this number does not exceed the address of the highest memory location.

Any time before the New Line key is depressed, a line may be edited using the Backspace key as described in the NASCOM 1 Software Notes. In addition, the character "!" (Shift "1") may be used to delete the entire line. When "!" is depressed, a "!" will appear on the screen at the current cursor position, indicating that the line has been deleted, and the editor prompt ":" is displayed ready for the next user input, eg.

```
:50 THIS LINE IS WRONK!      ("!" key pressed)  
:  
:                          (prompt displayed)
```

In this case, line 50 would not have been entered into the Edit Buffer.

At any time when ZEAP is in the process of displaying information (eg. when listing or assembling the source program) the user may interrupt the process by depressing the "!" key. ZEAP will immediately abandon its current processing and display the editor prompt ":" to indicate that it is ready to process editor commands.

Alternatively the "?" key (Shift "/") may be used under the same circumstances to temporarily hold the execution of ZEAP so that the contents of the screen can be examined at length. When the user wishes to resume execution, depressing any key will restart ZEAP where it left off, and processing will continue. In summary:

"!" Delete line; abandon execution
"?" Hold execution (resumed by pressing any key)

Error messages from the ZEAP editor are of the form

ERROR nn

where nn is the error number. An explanation of ZEAP error codes is given in Appendix B. The most common editor message is

ERROR 99

meaning that the last line of user input was illegal or unrecognisable as an editor command or line of source code.

If the first character of an input line is blank, the line is ignored by the editor.

2.2 ZEAP EDITOR COMMANDS

The following discussion is independent of any knowledge of the Z80 assembly language, and therefore the source lines shown are not suitable for assembly by the ZEAP assembler.

"v" Suppose the following lines are entered:

```
:20 LINE 2  
:10 LINE 1  
:30 LINE 3  
:
```

The user can examine the contents of part or all of the Edit Buffer using the "v" editor command. ("v" is a mnemonic for VDU List). Thus

```
:V 10 10  
0010 LINE 1  
:V 10 20  
0010 LINE 1  
0020 LINE 2  
:V 20  
0020 LINE 2  
0030 LINE 3  
:V  
0010 LINE 1  
0020 LINE 2  
0030 LINE 3  
:
```

Also note

```
:V 5 15  
0010 LINE 1  
:V 1 9  
:V 20 10  
:V 1000  
:
```

The last three commands cause no display.

In summary:

V m n	Display lines m to n inclusive
V m	Display lines from m to the end of the buffer
V	Display the entire contents of the source buffer

The space following "v" is optional, but if both m and n are specified, they must be separated by one or more spaces.

"U" When a source program has been entered by the user using the ZEAP editor, it is useful to be able to store all or part of it on cassette tape. This is achieved by the "U" editor command ("U" is a mnemonic for UART List). Its syntax is the same as that of the "V" command. Its operation is identical except that each line displayed is also output to the UART in a format which allows the line to be reloaded subsequently by the editor. Thus

```
:U  
0010 LINE 1  
0020 LINE 2  
0030 LINE 3  
:
```

would cause those lines displayed to be stored on an attached cassette recorder.

There is no identifiable Load command provided with ZEAP. Loading of source programs stored on tape using the "U" editor command is performed simply by switching the cassette recorder on while the editor prompt is displayed. ZEAP scans both the keyboard and the UART input during editor operation, and so source lines input from tape will be interpreted as if they had been entered manually. Thus playing back the above tape when the ZEAP editor prompt is displayed would cause the following display:

```
:0010 LINE 1  
:0020 LINE 2  
:0030 LINE 3  
:
```

and the three lines would be entered into the Edit Buffer as if they had been typed on the keyboard.

If the user attaches an ASCII terminal (teletype or equivalent) to the UART the "U" editor command can be used to obtain hard copy of all or part of the source program. The output of the "U" editor command is formatted with both NASBUG New Line characters and ASCII Carriage Return and Line Feed characters to support this facility. Thus, with an attached ASCII terminal

```
:U 10 20  
0010 LINE 1  
0020 LINE 2  
:
```

and the two lines displayed are also printed on the terminal.

"I" The ZEAP editor provides a convenient facility for the manual entry of blocks of source code, namely the "I" editor command("I" is a mnemonic for Auto Input). If the user enters

:I 40

the editor responds

:0040

and any input up to the New Line key is interpreted as Line 40. Suppose the following is typed:

:0040 LINE 4
:0050

After New Line is depressed the editor increments the sequence number by 10 and displays the new sequence number, ready for the entry of the next line of code, and so on:

:0050 LINE 5
:0060 LINE 6
:0070

Note that the necessary space following the sequence number is inserted by ZEAP, so that the user need not type it.

It is possible to edit the sequence number using the Backspace key. Entering these backspaces, followed by 95, followed by a space at this stage would result in the display

:0095

and then line 95 could be entered

:0095 LINE 7
:0105

Note that the increment of 10 is applied to the sequence number of the last line entered, and not of the last line displayed by ZEAP.

Exit from Auto Input mode (which is the name given to the above behaviour) is achieved by typing "!" (Shift "1") which deletes the current line and causes the usual editor prompt to be displayed, thus:

:0105 ! (user types "!"
:

Note that if it had existed prior to the above sequence of commands, line 105 would not have been deleted. Only the line of entry displayed would be deleted. To delete line 105, it would be necessary to enter the number 105 followed by the New Line key, not the "!" key as above.

If the number after the "I" is omitted, the editor displays

:0010

initially.

If a second number is typed after the "I", it is used as the sequence number increment. It must be less than 100. Thus:

:I 100 3	
<u>0100</u>	(New line pressed)
<u>0103</u>	(New line pressed)
<u>0106 !</u>	("!" pressed)
:	

So in summary

I	Enter Auto Input mode at line 10 with increments of 10
I s	Enter Auto Input mode at line s with increments of 10
I s i	Enter Auto Input mode at line s with increments of i

"X" Deleting a block of source code is made easier by the "X" editor command("X" is a mnemonic for eXpunge). "X" must always be followed by two numbers, separated by a space, which are the sequence numbers of the first and last lines to be deleted. All lines between and including these lines are deleted. Thus

```
:V
0010 LINE 1
0020 LINE 2
0030 LINE 3
0040 LINE 4
0050 LINE 5
0060 LINE 6
0095 LINE 7
:X 36 70
:V
0010 LINE 1
0020 LINE 2
0030 LINE 3
0095 LINE 7
:X 95
  ERROR 99
:X 95 95
:V
0010 LINE 1
0020 LINE 2
0030 LINE 3
:
```

Note that an attempt to use X with only one line number produced an error message.

To delete the entire edit buffer, the user should enter

```
:X 1 9999
:
```

This command does the job of a NEW or CLEAR utility in similar editors.

In summary

X m n Delete lines m to n inclusive

"Z" The limitation of line replacement as a method of correcting minor mistakes is clear from the following example:

```
:40 ILNE 4  
:  
:
```

To interchange the "I" and the "L" requires that the whole line be re-entered. A powerful alternative is provided in the ZEAP editor. Entering

```
:Z 40
```

causes the following two lines to be displayed:

```
:0040 ILNE 4  
↑
```

ZEAP has now entered Edit mode. The arrow under the first digit of the sequence number is the cursor. The user can advance the pointer to the position where the correction is to be made by depressing the space bar appropriately. After pressing it six times the display is:

```
:0040 ILNE 4  
↑ (6 spaces typed)
```

Now the offending letter "L" can be deleted by typing "<" (shift ","), thus

```
:0040 INE 4  
↑ ("<>" typed)
```

Note that all the characters to the right of the cursor have been moved up to fill the gap left by the deleted "L" Now, using the backspace key, the cursor can be positioned under the "I", before which an L is to be inserted:

```
:0040 INE 4  
↑ (Backspace typed)
```

Now to make room for the L the ">" (shift ".") is used:

```
:0040 INE 4  
↑
```

Note that all the characters above and to the right of the cursor are shifted one place right to make room for the insertion. Finally typing "L" will give

```
:0040 LINE 4  
↑
```

The "L" is inserted at the position of the cursor, which is then advanced one place.

3.3.3 COMMENT LINES

A comment line must begin with a ";", and all characters thereafter will be ignored by the assembler, except that they will appear on the assembly listing. The first 29 characters will be displayed on the assembly listing on the screen.

3.4 ASSEMBLER OPTIONS

"O" The "O" editor command allows various options to be set which define the output required from the assembler (O is a mnemonic for Options). The "O" may be followed by a single hexadecimal mask defining which options are ON and which are OFF. This mask is obtained by adding up the option codes of those options desired ON. Thus

```
:O 1A
```

would set assembler options MEMORY, TAPE and PASS 2 on, and NO LIST and TTY off (1A = 10 + 08 + 02 Hex). If no number follows the "O" all assembler options are set to the default values (ie. all off).

In summary:

O x	set assembler options from mask x
O	set all assembler options off

Appendix E contains a full account of each assembler option.

3.5 ASSEMBLER DIRECTIVES

The six assembler directives supported by ZEAP give the user the ability to control the generation of object code addresses, and to generate tables or liberal strings.

DEFB, DEFW and DEFM all cause the generation of object code for one or more bytes, words (double-bytes) and ASCII characters respectively.

EQU allows the direct assignment of an expression value to a symbolic name.

ORG and DEFS alter the assembly address ("\$\$") so that assembler programs may be assembled at any address, and to allow for space for storage of intermediate results and other variable information.

A full account of the assembler directives is given in Appendix D.

3.6 ASSEMBLY LISTING

A line of assembly listing takes the following form:

```
aaaa ccccccc ssss bbbbbb mmmm ppppppppppppppppppp
```

The explanation of the fields is as follows:

- aaaa 4 digit hexadecimal address of the instruction being assembled, except in a DEFB, DEFW or DEFM assembler directive, where it is the address of the first byte of code generated, and in a EQU, ORG or DEFS assembler directive, where it is the value of the expression in the operand field.
- ccccccc 2 to 8 hexadecimal digits representing the object code for the instruction, except in a DEFB, DEFW or DEFM assembler directive it contains only the first byte or word generated as appropriate.
- sss 4 digit sequence number of the current source line.
- bbbbbb 1 to 6 character label of the current source line. If no label is present, this field is left blank.
- mmm 2 to 4 character instruction mnemonic or assembler directive.
- ppp..... Operand and comment fields directly from source line.

If the source line is a comment (first character ";"), fields aaaa and ccccccc are left blank, and the comment is copied directly after the sequence number.

If the line contains an error, field ccccccc will contain

```
ERROR nn
```

and no object code is generated. A truncation error is reported on the following line, but the object generation is not suppressed.

Since the assembler formats the listing, there is no need to tabulate source programs. The fields of each source statement will be correctly formatted by the assembler. For example the source line

```
0040 BIM LD A,1
```

would appear in the assembly listing as

```
aaaa 3E01        0040 BIM        LD        A,1
```

where aaaa is the current value of the location counter ("\$").

3.7 OBJECT GENERATION

3.7.1 TAPE OBJECT

If the TAPE assembler option is on, object code is output through the UART to an attached cassette recorder in NASBUG format. Any block of object code in which the number of bytes generated is not an exact multiple of eight (the length of a NASBUG record) is padded out with random data. Provided the object code is generated in strict address order this will cause no trouble to the user.

Object code generated in this manner can be loaded using NASBUG's "L" command as if the data had been saved using "D". The user should make a note of the execution address of his program from the source listing so that he may correctly begin execution of his program.

The tape LED is used by ZEAP in the same way as it is by NASBUG, and may be used as a direct or indirect indication to start the cassette recorder as described in the NASCOM 1 documentation.

3.7.2 MEMORY OBJECT

If the MEMORY assembler option is on, object code is assembled direct to memory. Object instructions and data are written as they are assembled to the appropriate memory address. Great care must be exercised when using this option, as NO CHECK is made that object code is not overwriting the Edit Buffer or ZEAP itself, or even that there is RAM at the address where the object code is being written. If no ORG assembler directive appears in the source program, assembly will begin at the first available byte of RAM not being used by ZEAP, as displayed in the corner of the screen, but the user should bear in mind that the object program may overflow available memory with no warning.

A program so assembled may be executed by entering NASBUG using the ZEAP "N" editor command and executing the object code using NASBUG's "E" command. The object program should set the stack pointer to a free area of memory if the stack is to be used, so that ZEAP's own stack does not overflow.

If the object program works incorrectly it may be necessary to reload ZEAP from tape, and enter the source program again. For this reason it is recommended that the source program be saved on tape before testing an object program, in case valuable data is lost and has to be typed in again.

APPENDIX A

ZEAP OPERATION

ZEAP should be loaded from the tape provided. First the loader should be loaded using the "L" command. This will cause a short program to be placed at location £0C50. Object code for this program is given in the latter part of Appendix I. Zeap itself is then loaded by executing from £0C50. Any lines containing a check sum error will be scrolled up on the screen and may be corrected from the object code listing in Appendix I.

ZEAP loads at £1000 and is about 2.82K bytes in length. The area from £0F00 to £0FFF is used as ZEAP's register storage and stack space. The source buffer begins directly after ZEAP. The area from £0C50 to £0EFF is not used by ZEAP, and may therefore contain programs or other user information.

To execute ZEAP enter:

>EFOO

If the "N" editor command is used to return to NASBUG, ZEAP may be re-entered by entering:

>EFOO

provided that it has not been corrupted. In this case the edit buffer will be intact but the assembler options will have been reset.

A limit on the memory used for source program storage can be imposed, eg. to stop the edit buffer from overflowing higher than £3000 enter:

>EFOO 3000

when executing ZEAP. The default setting is the last limit specified (or £5000 initially).

APPENDIX B

ZEAP ERROR CODES

ERROR 00 CORE FULL

The source line just entered would cause an overflow of the edit buffer. The source line was not entered into the buffer. However, if the line was to replace an existing line, the original line was deleted.

ERROR 01 RESEQUENCE OVERFLOW

During the execution of a RESEQUENCE editor command the line number became greater than 9999. The source file is resequenced starting with line 1 in steps of 1.

ERROR 02 AUTO INPUT OVERFLOW

In AUTO-INPUT mode the line number became greater than 9999. AUTO-INPUT mode is abandoned.

ERROR 03 NON-EXISTENT LINE

An attempt was made to edit a non-existent line with the "Z" editor command.

ERROR 10 UNRECOGNISABLE STATEMENT

A label is more than 6 characters, or a mnemonic is more than 4 characters or omitted. The statement is ignored.

ERROR 20 UNKNOWN MNEMONIC

The op-code field contains an unrecognisable mnemonic. The statement is ignored.

ERROR 21 CONTEXT ERROR

The combination of op-code and operand types encountered is illegal or a mnemonic is too short. The statement is ignored.

ERROR 22 INDEX REGISTER ERROR

IX or IY is used where only HL is permitted, or in a JP (IX) or JP (IY) instruction, the displacement is non-zero. The statement is ignored.

ERROR 23 TRUNCATION ERROR

An 8 bit operand is greater than 255 or less than -128 or an index register displacement value is greater than 127 or less than -128, or a relative branch offset is greater than 129 or less than -126, or a bit number in a BIT, SET or RES instruction is greater than 7 or less than 0, or an address in an RST instruction is illegal, or the mode in an IM instruction is not 0, 1 or 2. The value in question is truncated and assembly of the statement continues.

ERROR 24 TOO MANY REGISTERS

A register symbol appears in an assembler directive operand, or more than one register appears in an instruction operand. The statement is ignored.

ERROR 25 REGISTER MISMATCHED

The combination of first and second operand types is illegal. The statement is ignored.

ERROR 26 ILLEGAL CHARACTER

The operands field contains a character whose meaning is unassigned in the syntax of the assembly language. The statement is ignored.

ERROR 27 ILLEGAL OPERAND

The combination of a register and a label or constant in this context is illegal. The statement is ignored.

ERROR 28 PARENTHESIS ERROR

A left parenthesis occurs in an assembler directive operand, or more than one left parenthesis occurs in an instruction operand. The statement is ignored.

ERROR 30 LABEL NOT FOUND

A symbol in an expression does not occur in the label field of any statement in the source code. The statement is ignored.

ERROR 31 LABEL REDEFINED

The symbol in the label field has previously appeared in a label field, or is a register name. The label is ignored and the rest of the statement is assembled.

ERROR 40 DIRECTIVE ERROR

In an assembler directive, too few or too many operands appear. The statement is ignored.

ERROR 41 ILLEGAL FORWARD REFERENCE

A label symbol in an EQU, ORG or DEFS assembler directive is defined after the directive is encountered. The statement is ignored.

ERROR 50 ERRORS IN ASSEMBLY

There were errors flagged in the previous assembly.

ERROR 90 CHECKSUM ERROR

Part of ZEAP has been corrupted due to hardware errors or user tampering. If ZEAP is not reloaded, unpredictable errors may occur.

ERROR 99 ILLEGAL COMMAND

An unrecognisable editor command or an ill-formed source code line was entered. The input line is ignored.

APPENDIX C

ZEAP EDITOR COMMANDS

The following symbols are used. All numbers are decimal unless otherwise stated.

- y sequence number (ie. source line number)
- m first sequence number to which command is applied (default 1)
- n last sequence number to which command is applied (default 9999)
- s starting sequence number (default 10)
- i increment (default 10)
- x hexadecimal option mask
- h hexadecimal number

Numbers are separated from the command letter and from each other by one or more spaces.

If n is explicitly specified then m must be also.
If i is explicitly specified then s must be also.

A m n ASSEMBLE SOURCE PROGRAM (ASSEMBLE)

Causes assembly of the indicated portion of the source program, with the options defined by the last SET ASSEMBLER OPTIONS command in effect. See section 3 for more details.

I s i ENTER AUTO-INPUT MODE (AUTO-INPUT)

Causes the ZEAP editor to enter AUTO-INPUT mode. The number s is displayed, followed by a space. The user may then enter a line of source code terminated by the New Line key, whereupon that line of code is entered into the edit buffer, i is added to s, and the new sequence number is displayed. The user may continue to enter source code as long as the sequence number remains less than 10000.

Exit from AUTO-INPUT mode is achieved by entering the line delete character, "!" (shift "1"). The editor then prompts for the next command.

N RETURN TO NASBUG (NASBUG)

Causes ZEAP to return control to NASBUG, allowing any of NASBUG's monitor commands to be used, for example to alter any of ZEAP's internal registers in accordance with Appendix G, or to execute a program assembled in memory.

Provided the area of memory used by ZEAP is unchanged during NASBUG operation, ZEAP may be re-entered with the edit buffer intact, in accordance with the procedure described in Appendix A.

F/string/FIND STRING (FIND)

FT Searches for a specified string in the edit buffer, and if found, opens the line containing it for editing.

The form "F/string/" is used to search from the beginning of the edit buffer for a character string of up to six characters. The "/" represents a delimiter character, which may be any character, except space, but which must follow directly after the "F". If the second delimiter is omitted or the string is more than six characters long the command is treated as an "FT" command (described below). If the string is found, the line containing it is displayed and opened for editing (see EDIT SOURCE LINE). If the string is not found the ZEAP editor prompts for the next command.

The form "F" is used to search for the string specified in the most recent "F/string/" command, starting from the last occurrence of that string found, instead of from the beginning of the edit buffer. Otherwise it is identical to the "F/string/" command described above.

The form "FT" is used to search for the string specified in the most recent "F/string/" command, starting from the beginning of the edit buffer. Otherwise it is identical to the "F/string/" command described above.

O x SET ASSEMBLER OPTIONS (OPTIONS)

Sets assembler options specified by the hexadecimal number x. The options and their hexadecimal codes are as follows. See section 3.4 for more details.

- + 01 SUPPRESS SOURCE LISTING (NO LIST)
- + 02 OBJECT CODE TO MEMORY (MEMORY)
- + 04 SOURCE LISTING TO TTY (TTY)
- + 08 OBJECT CODE TO TAPE (TAPE)
- + 10 FORCE SECOND PASS (PASS 2)
- + 20 ADJUST RELATIVE JUMP OFFSETS (ADJUST REL)

Initially all options are off.

3.3.3 COMMENT LINES

A comment line must begin with a ";", and all characters thereafter will be ignored by the assembler, except that they will appear on the assembly listing. The first 29 characters will be displayed on the assembly listing on the screen.

3.4 ASSEMBLER OPTIONS

"O" The "O" editor command allows various options to be set which define the output required from the assembler (O is a mnemonic for Options). The "O" may be followed by a single hexadecimal mask defining which options are ON and which are OFF. This mask is obtained by adding up the option codes of those options desired ON. Thus

```
:O 1A
```

would set assembler options MEMORY, TAPE and PASS 2 on, and NO LIST and TTY off (1A = 10 + 08 + 02 Hex). If no number follows the "O" all assembler options are set to the default values (ie. all off).

In summary:

O x	set assembler options from mask x
O	set all assembler options off

Appendix E contains a full account of each assembler option.

3.5 ASSEMBLER DIRECTIVES

The six assembler directives supported by ZEAP give the user the ability to control the generation of object code addresses, and to generate tables or liberal strings.

DEFB, DEFW and DEFM all cause the generation of object code for one or more bytes, words (double-bytes) and ASCII characters respectively.

EQU allows the direct assignment of an expression value to a symbolic name.

ORG and DEFS alter the assembly address ("\$\$") so that assembler programs may be assembled at any address, and to allow for space for storage of intermediate results and other variable information.

A full account of the assembler directives is given in Appendix D.

3.6 ASSEMBLY LISTING

A line of assembly listing takes the following form:

```
aaaa ccccccc ssss bbbbbb mmmm ppppppppppppppppppp
```

The explanation of the fields is as follows:

- aaaa 4 digit hexadecimal address of the instruction being assembled, except in a DEFB, DEFW or DEFM assembler directive, where it is the address of the first byte of code generated, and in a EQU, ORG or DEFS assembler directive, where it is the value of the expression in the operand field.
- ccccccc 2 to 8 hexadecimal digits representing the object code for the instruction, except in a DEFB, DEFW or DEFM assembler directive it contains only the first byte or word generated as appropriate.
- sss 4 digit sequence number of the current source line.
- bbbbbb 1 to 6 character label of the current source line. If no label is present, this field is left blank.
- mmm 2 to 4 character instruction mnemonic or assembler directive.
- ppp..... Operand and comment fields directly from source line.

If the source line is a comment (first character ";"), fields aaaa and ccccccc are left blank, and the comment is copied directly after the sequence number.

If the line contains an error, field ccccccc will contain

```
ERROR nn
```

and no object code is generated. A truncation error is reported on the following line, but the object generation is not suppressed.

Since the assembler formats the listing, there is no need to tabulate source programs. The fields of each source statement will be correctly formatted by the assembler. For example the source line

```
0040 BIM LD A,1
```

would appear in the assembly listing as

```
aaaa 3E01        0040 BIM        LD        A,1
```

where aaaa is the current value of the location counter ("\$").

3.7 OBJECT GENERATION

3.7.1 TAPE OBJECT

If the TAPE assembler option is on, object code is output through the UART to an attached cassette recorder in NASBUG format. Any block of object code in which the number of bytes generated is not an exact multiple of eight (the length of a NASBUG record) is padded out with random data. Provided the object code is generated in strict address order this will cause no trouble to the user.

Object code generated in this manner can be loaded using NASBUG's "L" command as if the data had been saved using "D". The user should make a note of the execution address of his program from the source listing so that he may correctly begin execution of his program.

The tape LED is used by ZEAP in the same way as it is by NASBUG, and may be used as a direct or indirect indication to start the cassette recorder as described in the NASCOM 1 documentation.

3.7.2 MEMORY OBJECT

If the MEMORY assembler option is on, object code is assembled direct to memory. Object instructions and data are written as they are assembled to the appropriate memory address. Great care must be exercised when using this option, as NO CHECK is made that object code is not overwriting the Edit Buffer or ZEAP itself, or even that there is RAM at the address where the object code is being written. If no ORG assembler directive appears in the source program, assembly will begin at the first available byte of RAM not being used by ZEAP, as displayed in the corner of the screen, but the user should bear in mind that the object program may overflow available memory with no warning.

A program so assembled may be executed by entering NASBUG using the ZEAP "N" editor command and executing the object code using NASBUG's "E" command. The object program should set the stack pointer to a free area of memory if the stack is to be used, so that ZEAP's own stack does not overflow.

If the object program works incorrectly it may be necessary to reload ZEAP from tape, and enter the source program again. For this reason it is recommended that the source program be saved on tape before testing an object program, in case valuable data is lost and has to be typed in again.

APPENDIX A

ZEAP OPERATION

ZEAP should be loaded from the tape provided. First the loader should be loaded using the "L" command. This will cause a short program to be placed at location £0C50. Object code for this program is given in the latter part of Appendix I. Zeap itself is then loaded by executing from £0C50. Any lines containing a check sum error will be scrolled up on the screen and may be corrected from the object code listing in Appendix I.

ZEAP loads at £1000 and is about 2.82K bytes in length. The area from £0F00 to £0FFF is used as ZEAP's register storage and stack space. The source buffer begins directly after ZEAP. The area from £0C50 to £0EFF is not used by ZEAP, and may therefore contain programs or other user information.

To execute ZEAP enter:

>EFOO

If the "N" editor command is used to return to NASBUG, ZEAP may be re-entered by entering:

>EFOO

provided that it has not been corrupted. In this case the edit buffer will be intact but the assembler options will have been reset.

A limit on the memory used for source program storage can be imposed, eg. to stop the edit buffer from overflowing higher than £3000 enter:

>EFOO 3000

when executing ZEAP. The default setting is the last limit specified (or £5000 initially).

APPENDIX B

ZEAP ERROR CODES

ERROR 00 CORE FULL

The source line just entered would cause an overflow of the edit buffer. The source line was not entered into the buffer. However, if the line was to replace an existing line, the original line was deleted.

ERROR 01 RESEQUENCE OVERFLOW

During the execution of a RESEQUENCE editor command the line number became greater than 9999. The source file is resequenced starting with line 1 in steps of 1.

ERROR 02 AUTO INPUT OVERFLOW

In AUTO-INPUT mode the line number became greater than 9999. AUTO-INPUT mode is abandoned.

ERROR 03 NON-EXISTENT LINE

An attempt was made to edit a non-existent line with the "Z" editor command.

ERROR 10 UNRECOGNISABLE STATEMENT

A label is more than 6 characters, or a mnemonic is more than 4 characters or omitted. The statement is ignored.

ERROR 20 UNKNOWN MNEMONIC

The op-code field contains an unrecognisable mnemonic. The statement is ignored.

ERROR 21 CONTEXT ERROR

The combination of op-code and operand types encountered is illegal or a mnemonic is too short. The statement is ignored.

ERROR 22 INDEX REGISTER ERROR

IX or IY is used where only HL is permitted, or in a JP (IX) or JP (IY) instruction, the displacement is non-zero. The statement is ignored.

ERROR 23 TRUNCATION ERROR

An 8 bit operand is greater than 255 or less than -128 or an index register displacement value is greater than 127 or less than -128, or a relative branch offset is greater than 129 or less than -126, or a bit number in a BIT, SET or RES instruction is greater than 7 or less than 0, or an address in an RST instruction is illegal, or the mode in an IM instruction is not 0, 1 or 2. The value in question is truncated and assembly of the statement continues.

ERROR 24 TOO MANY REGISTERS

A register symbol appears in an assembler directive operand, or more than one register appears in an instruction operand. The statement is ignored.

ERROR 25 REGISTER MISMATCHED

The combination of first and second operand types is illegal. The statement is ignored.

ERROR 26 ILLEGAL CHARACTER

The operands field contains a character whose meaning is unassigned in the syntax of the assembly language. The statement is ignored.

ERROR 27 ILLEGAL OPERAND

The combination of a register and a label or constant in this context is illegal. The statement is ignored.

ERROR 28 PARENTHESIS ERROR

A left parenthesis occurs in an assembler directive operand, or more than one left parenthesis occurs in an instruction operand. The statement is ignored.

ERROR 30 LABEL NOT FOUND

A symbol in an expression does not occur in the label field of any statement in the source code. The statement is ignored.

ERROR 31 LABEL REDEFINED

The symbol in the label field has previously appeared in a label field, or is a register name. The label is ignored and the rest of the statement is assembled.

ERROR 40 DIRECTIVE ERROR

In an assembler directive, too few or too many operands appear. The statement is ignored.

ERROR 41 ILLEGAL FORWARD REFERENCE

A label symbol in an EQU, ORG or DEFS assembler directive is defined after the directive is encountered. The statement is ignored.

ERROR 50 ERRORS IN ASSEMBLY

There were errors flagged in the previous assembly.

ERROR 90 CHECKSUM ERROR

Part of ZEAP has been corrupted due to hardware errors or user tampering. If ZEAP is not reloaded, unpredictable errors may occur.

ERROR 99 ILLEGAL COMMAND

An unrecognisable editor command or an ill-formed source code line was entered. The input line is ignored.

APPENDIX C

ZEAP EDITOR COMMANDS

The following symbols are used. All numbers are decimal unless otherwise stated.

- y sequence number (ie. source line number)
- m first sequence number to which command is applied (default 1)
- n last sequence number to which command is applied (default 9999)
- s starting sequence number (default 10)
- i increment (default 10)
- x hexadecimal option mask
- h hexadecimal number

Numbers are separated from the command letter and from each other by one or more spaces.

If n is explicitly specified then m must be also.
If i is explicitly specified then s must be also.

A m n ASSEMBLE SOURCE PROGRAM (ASSEMBLE)

Causes assembly of the indicated portion of the source program, with the options defined by the last SET ASSEMBLER OPTIONS command in effect. See section 3 for more details.

I s i ENTER AUTO-INPUT MODE (AUTO-INPUT)

Causes the ZEAP editor to enter AUTO-INPUT mode. The number s is displayed, followed by a space. The user may then enter a line of source code terminated by the New Line key, whereupon that line of code is entered into the edit buffer, i is added to s, and the new sequence number is displayed. The user may continue to enter source code as long as the sequence number remains less than 10000.

Exit from AUTO-INPUT mode is achieved by entering the line delete character, "!" (shift "1"). The editor then prompts for the next command.

N RETURN TO NASBUG (NASBUG)

Causes ZEAP to return control to NASBUG, allowing any of NASBUG's monitor commands to be used, for example to alter any of ZEAP's internal registers in accordance with Appendix G, or to execute a program assembled in memory.

Provided the area of memory used by ZEAP is unchanged during NASBUG operation, ZEAP may be re-entered with the edit buffer intact, in accordance with the procedure described in Appendix A.

F/string/FIND STRING (FIND)

FT Searches for a specified string in the edit buffer, and if found, opens the line containing it for editing.

The form "F/string/" is used to search from the beginning of the edit buffer for a character string of up to six characters. The "/" represents a delimiter character, which may be any character, except space, but which must follow directly after the "F". If the second delimiter is omitted or the string is more than six characters long the command is treated as an "FT" command (described below). If the string is found, the line containing it is displayed and opened for editing (see EDIT SOURCE LINE). If the string is not found the ZEAP editor prompts for the next command.

The form "F" is used to search for the string specified in the most recent "F/string/" command, starting from the last occurrence of that string found, instead of from the beginning of the edit buffer. Otherwise it is identical to the "F/string/" command described above.

The form "FT" is used to search for the string specified in the most recent "F/string/" command, starting from the beginning of the edit buffer. Otherwise it is identical to the "F/string/" command described above.

O x SET ASSEMBLER OPTIONS (OPTIONS)

Sets assembler options specified by the hexadecimal number x. The options and their hexadecimal codes are as follows. See section 3.4 for more details.

- + 01 SUPPRESS SOURCE LISTING (NO LIST)
- + 02 OBJECT CODE TO MEMORY (MEMORY)
- + 04 SOURCE LISTING TO TTY (TTY)
- + 08 OBJECT CODE TO TAPE (TAPE)
- + 10 FORCE SECOND PASS (PASS 2)
- + 20 ADJUST RELATIVE JUMP OFFSETS (ADJUST REL)

Initially all options are off.

R s i RESEQUENCE SOURCE CODE (RESEQUENCE)

Remembers all the statements in the edit buffer so that the first line is given the number s, and subsequent lines s+i s+2i, etc. as for the "I" editor command.

U m n LISTING TO UART (SAVE)

Causes the indicated portion of the source program to be output to the UART, and simultaneously displayed on the screen.

The output through the UART is formatted to drive either a cassette tape recorder, so that any portion of the source program may be stored permanently and loaded subsequently by ZEAP, or an ASCII terminal to obtain a hard copy listing of any portion of the source program.

V m n LISTING TO VDU (LIST)

Causes the indicated portion of the source program to be displayed on the screen.

X m n BLOCK DELETE (DELETE)

Causes all source lines numbered m to n inclusive to be deleted. Both m and n must be specified.

Z y EDIT SOURCE LINE (EDIT)

Displays line y and opens for edit. The following keys are available for specified functions:

Space	Move pointer right
Backspace	Move pointer left
">" (Shift ".")	Insert
"<" (Shift ",")	Delete
New line	Leave edit
"!" (Shift "l")	Abandon edit

P h SET MEMORY OFFSET (OFFSET)

Set to h the number to be added to the logical assembly address to obtain the physical location of the object code in memory when the MEMORY assembler option is on.

Q h SET I/O RATES (RATES)

Set the inter-character delay to cc hex and the end of line delay (for use with the U editor command and TTY assembler option) to dd hex, where h = ccdd.

APPENDIX D

ASSEMBLER DIRECTIVES

label EQU exp (; comment) EQUATE SYMBOL

The label is given the value of the 16 bit expression in the operand field. All symbols appearing in the expression must have been previously defined. No object code is generated. The label may not be redefined.

label ORG exp (; comment) SET ORIGIN

The location counter (\$) is given the value of the 16 bit expression in the operand field. All symbols appearing in the expression must have been previously defined. No object code is generated. Assembly continued at the new origin. If a label is present, it is given the value of the expression.

(label) DEFS exp (; comment) DEFINE SPACE

The location counter (\$) is increased by the value of the 16 bit expression in the operand field. All symbols appearing in the expression must have been previously defined. No object code is generated. Assembly continues after a block of memory of length exp. If a label is present, it is given the original value of the location counter (\$).

(label) DEFB exp (,exp).... (; comment) DEFINE BYTE

For each 16 bit expression one byte of code is generated with the value of that expression. Expressions may contain forward references. If a label is present, it is given the value of the address of the first byte of code generated.

(label) DEFW exp (, exp).... (; comment) DEFINE WORD

For each 16 bit expression two bytes of code are generated with the value of that expression, the low order 8 bits occupying the first byte and the high order 8 bits the second. Expression may contain forward references. If a label is present, it is given the value at the address of the first byte of code generated.

(label) DEFM /string/ (; comment) DEFINE MESSAGE

The "/" may be any character except blank or comma. For each character after the first delimiter until the second delimiter or the end of the line is encountered, one byte of code is generated having the value of the ASCII code for that character, with bit 7 zero. Any characters may appear between the delimiters. Characters after the second occurrence of the delimiter are ignored. If a label is present, it is given the value of the address of the first byte of code generated.

APPENDIX E

ASSEMBLER OPTIONS

Assembler options are set by the OPTIONS editor command. All assembler options must be explicitly specified as on or off, and remain in effect until the next OPTIONS editor command is issued, or until ZEAP is reloaded. All assembler options are initially off, and are all switched off whenever ZEAP is re-entered.

Assembler options are selected as ON by adding the hexadecimal option codes of the desired assembler options together. Thus the TTY and PASS 2 assembler options would be selected as on by entering the command "0 14". If no mask is specified, all options are set to the default off state.

+01 SUPPRESS SOURCE LISTING (NO LISTING)

During the second pass, no source listing will be displayed on the screen. Lines containing errors will, however, still be displayed.

+02 OBJECT CODE TO MEMORY (MEMORY)

During the second pass, the object code will be assembled directly into memory. No check is made to see that the object code is not overwriting parts of ZEAP and/or the edit buffer, nor that there is read/write memory at the address where code is being written. See section 3.7 for more details.

+04 SOURCE LISTING TO TTY (TTY)

During the second pass, any source listing will be listed on an ASCII terminal attached to the UART. If the NO LIST assembler option is on, only those lines containing errors will be listed. The output from the UART is not suitable for storage on cassette tape.

This assembler option may not be used in conjunction with the TAPE assembler option, described below.

+08 OBJECT CODE TO TAPE (TAPE)

During the second pass, the object code will be dumped in NASBUG format to a cassette tape recorder attached to the UART. The object program may be subsequently loaded using NASBUG's LOAD function, and executed under NASBUG control. No object code is written to memory (unless the MEMORY assembler option is on) so that object code cannot overwrite ZEAP or the edit buffer.

This assembler option may not be used in conjunction with the TTY assembler option, described above.

+10 FORCE SECOND PASS (PASS 2)

Normally if errors are detected during the first pass, the second pass is suppressed. If this assembler option is on, however, the second pass will be executed regardless.

+20 ADJUST RELATIVE JUMP OFFSETS (ADJUST REL)

Different standards in implementing the JR and similar instructions by different manufacturers. The assembler normally expects the argument to a relative jump instruction to be an expression which is the offset from the location of the current instruction to the destination, eg.

```
      JR Z,3            ; BRANCH ROUND LD INSTRUCTION
      LD (SWITCH), A
      RET
```

or, more conveniently

```
      JR Z, RETURN - §
      LD (SWITCH), A
RETURN RET
```

The ADJUST REL assembler option causes the assembler to automatically subtract the value of § from the argument of each relative jump instruction, so that the presentation of the source code is in line with absolute jump and call instructions. Thus with the ADJUST REL assembler option set, the following code now achieves the desired result

```
      JR Z, RETURN
      LD (SWITCH), A
RETURN RET
```

or

```
      JR §+3
      LD (SWITCH), A
      RET
```

Note that the first two examples would probably give a truncation error if the ADJUST REL assembler option is set. The convention adopted must be fixed throughout the whole program.

APPENDIX F

INSTRUCTION SET

The executable instruction set is defined in the ZILOG publication Z80-CPU Technical Manual, and in the MOSTEK publication Z80 Micro Computer Devices Technical Manual. For a full explanation of the instruction set one should have these manuals together with the assembly language programming manuals published by either company. A summary of the executable mnemonics is set out below.

EXECUTABLE INSTRUCTIONS

ADC	HL,SS	ADD WITH CARRY REG. PAIR SS TO HL
ADD	A,S	ADD WITH CARRY OPERAND S TO ACC.
ADD	A,N	ADD VALUE N TO ACC.
ADD	A,R	ADD REG. R TO ACC.
ADD	A,(HL)	ADD LOCATION (HL) TO ACC.
ADD	A,(IX+D)	ADD LOCATION(IX+D) TO ACC
ADD	A,(IY+D)	ADD LOCATION (IY+D) TO ACC.
ADD	HL,SS	ADD REG. PAIR SS TO HL
ADD	IX,PP	ADD REG. PAIR PP TO IX
ADD	IY,RR	ADD REG. PAIR RR TO IY
AND	S	LOGICAL 'AND' OF OPERAND S AND ACC.
BIT	B,(HL)	TEST BIT B OF LOCATION (HL)
BIT	B,(IX+D)	TEST BIT B OF LOCATION (IX+D)
BIT	B,(IY+D)	TEST BIT B OF LOCATION (IY+D)
BIT	B,R	TEST BIT B OF REG. R
CALL	CC,NN	CALL SUBROUTINE AT LOCATION NN IF CONDITION CC IF TRUE
CALL	NN	UNCONDITIONAL CALL SUBROUTINE AT LOCATION NN
CCF		COMPLEMENT CARRY FLAG
CP	S	COMPARE OPERAND S WITH ACC.
CPD		COMPARE LOCATION (HL) AND ACC.DECREMENT HL AND BC UNTIL CB=0
CPDR		COMPARE LOCATION(HL) AND ACC. DECREMENT HL AND BC, REPEAT
CPI		COMPARE LOCATION (HL) AND ACC. INCREMENT HL AND DECREMENT BC
CPIR		COMPARE LOCATION (HL) AND ACC. INCREMENT HL, DECREMENT BC REPEAT UNTIL BC=0
CPL		COMPLEMENT ACC. (1'S COMP)
DAA		DECIMAL ADJUST ACC.
DEC	M	DECREMENT OPERAND M
DEC	IX	DECREMENT IX
DEC	IY	DECREMENT IY
DEC	SS	DECREMENT REG. PAIR SS
DI		DISABLE INTERRUPTS
DJNZ	E	DECREMENT B AND JUMP RELATIVE IF B=0
EI		ENABLE INTERRUPTS
EX	(SP),HL	EXCHANGE THE LOCATION (SP) AND HL
EX	(SP),IX	EXCHANGE THE LOCATION (SP) AND IX
EX	(SP),IY	EXCHANGE THE LOCATION (SP) AND IY
EX	AF,AF'	EXCHANGE THE CONTENTS OF AF AND AF'
EX	DE,HL	EXCHANGE THE CONTENTS OF DE AND HL
EXX		EXCHANGE THE CONTENTS OF BC, DE, HL WITH CONTENTS OF BC', DE', HL', RESPECTIVELY
HALT		HALT (WAIT FOR INTERRUPT OR RESET)
IM	0	SET INTERRUPT MODE 0
IM	1	SET INTERRUPT MODE 1
IM	2	SET INTERRUPT MODE 2

LD	SP,IX	LOAD SP WITH IX
LD	SP,IY	LOAD SP WITH IY
LDD		LOAD LOCATION (DE) WITH LOCATION (HL), DECREMENT DE, HL AND BC
LDDR		LOAD LOCATION (DE) WITH LOCATION (HL), DECREMENT DE, HL AND BC; REPEAT UNTIL BC=0
LDI		LOAD LOCATION (DE) WITH LOCATION (HL), INCREMENT DE, HL, DECREMENT BC
LDIR		LOAD LOCATION (DE) WITH LOCATION (HL), INCREMENT DE, HL, DECREMENT BC AND REPEAT UNTIL BC=0
NEG		NEGATE ACC. (2'S COMPLEMENT)
NOP		NO OPERATION
OR	S	LOGICAL 'OR' OR OPERAND S AND ACC.
OTDR		LOAD OUTPUT PORT (C) WITH LOCATION (HL) DECREMENT HL AND B, REPEAT UNTIL B=0
O		LOAD OUTPUT PORT (C) WITH LOCATION (HL), INCREMENT HL, DECREMENT B, REPEAT UNTIL B=0
OUT	(C),R	LOAD OUTPUT PORT (C) WITH REG. R
OUT	(N),A	LOAD OUTPUT PORT (N) WITH ACC.
OUTD		LOAD OUTPUT PORT (C) WITH LOCATION (HL), DECREMENT HL AND B
OUTI		LOAD OUTPUT PORT (C) WITH LOCATION (HL), INCREMENT HL AND DECREMENT B
POP	IX	LOAD IX WITH TOP OF STACK
POP	IY	LOAD IY WITH TOP OF STACK
POP	QQ	LOAD REG. PAIR QQ WITH TOP OF STACK
PUSH	IX	LOAD IX ONTO STACK
PUSH	IY	LOAD IY ONTO STACK
PUSH	QQ	LOAD REG. PAIR QQ ONTO STACK
RES	B,M	RESET BIT B OF OPERAND M
RET		RETURN FROM SUBROUTINE
RET	CC	RETURN FROM SUBROUTINE IF CONDITION CC IS TRUE
RETI		RETURN FROM INTERRUPT
RETN		RETURN FROM NON MASKABLE INTERRUPT
RL	M	ROTATE LEFT THROUGH CARRY OPERAND M
RL		ROTATE LEFT ACC. THROUGH CARRY
RLC	(HL)	ROTATE LOCATION (HL) LEFT CIRCULAR
RLC	(IX+D)	ROTATE LOCATION (IX+D) LEFT CIRCULAR
RLC	(IY+D)	ROTATE LOCATION (IY+D) LEFT CIRCULAR
RLC	R	ROTATE REG. R LEFT CIRCULAR
RLCA		ROTATE LEFT CIRCULAR ACC.
RLD		ROTATE DIGIT LEFT AND RIGHT BETWEEN ACC. AND LOCATION (HL)
RR	M	ROTATE RIGHT THROUGH CARRY OPERAND M
RRA		ROTATE RIGHT ACC. THROUGH CARRY
RRC	M	ROTATE OPERAND M RIGHT CIRCULAR
RRCA		ROTATE RIGHT CIRCULAR ACC.
RRD		ROTATE DIGIT RIGHT AND LEFT BETWEEN ACC. AND LOCATION (HL)
RST	P	RESTART TO LOCATION P
SBC	A,S	SUBTRACT OPERAND S FROM ACC. WITH CARRY
SBC	HL,SS	SUBTRACT REG. PAIR SS FROM HL WITH CARRY
SCF		SET CARRY FLAG (C=1)
SET	B,(HL)	SET BIT B OF LOCATION (HL)
SET	B,(IX+D)	SET BIT B OF LOCATION (IX+D)
SET	B,(IY+D)	SET BIT B OF LOCATION (IY+D)
SET	B,R	SET BIT B OF REG. R
SLA	M	SHIFT OPERAND M LEFT ARITHMETIC
SRA	M	SHIFT OPERAND M RIGHT ARITHMETIC
SRL	M	SHIFT OPERAND M RIGHT LOGICAL
SUB	S	SUBTRACT OPERAND S FROM ACC.
XOR	S	EXCLUSIVE 'OR' OPERAND S AND ACC.

IN	A,(N)	LOAD THE ACC. WITH INPUT FROM DEVICE N
IN	R,(C)	LOAD THE REG. R WITH INPUT FROM DEVICE (C)
INC	(HL)	INCREMENT LOCATION (HL)
INC	IX	INCREMENT IX
INC	(IX+D)	INCREMENT LOCATION (IX+D)
INC	IY	INCREMENT IY
INC	(IY+D)	INCREMENT LOCATION (IY+D)
INC	R	INCREMENT REG. R
INC	SS	INCREMENT REG. PAIR SS
IND		LOAD LOCATION (HL) WITH INPUT FROM PORT (C), DECREMENT HL AND B
INDR		LOAD LOCATION (HL) WITH INPUT FROM PORT (C), DECREMENT HL AND DECREMENT B, REPEAT UNTIL B=0
INI		LOAD LOCATION (HL) WITH INPUT FROM PORT (C); AND INCREMENT HL AND DECREMENT B
IY		LOAD LOCATION (HL) WITH INPUT FROM PORT (C), INCREMENT HL AND DECREMENT B, REPEAT UNTIL B=0
JP	(HL)	UNCONDITIONAL JUMP TO (HL)
JP	(IX)	UNCONDITIONAL JUMP TO (IX)
JP	(IY)	UNCONDITIONAL JUMP TO (IY)
JP	CC,NN	JUMP TO LOCATION NN IF CONDITION CC IS TRUE
JP	NN	UNCONDITIONAL JUMP TO LOCATION NN
JP	C,E	JUMP RELATIVE TO PC+E IF CARRY=1
JR	E	UNCONDITIONAL JUMP RELATIVE TO PC+E
JP	NC,E	JUMP RELATIVE TO PC+E IF CARRY=0
JR	NZ,E	JUMP RELATIVE TO PC+E IF NON ZERO (Z=0)
JR	Z,E	JUMP RELATIVE TO PC+E IF ZERO (Z=1)
LD	A,(BC)	LOAD ACC. WITH LOCATION (BC)
LD	A,(DE)	LOAD ACC. WITH LOCATION (DE)
LD	A,I	LOAD ACC. WITH I
LD	A,(NN)	LOAD ACC. WITH LOCATION NN
LD	A,R	LOAD ACC. WITH REG. R
LD	(BC),A	LOAD LOCATION (BC) WITH ACC.
LD	(DE),A	LOAD LOCATION (DE) WITH ACC.
LD	(HL),N	LOAD LOCATION (HL) WITH VALUE N
LD	DD,NN	LOAD REG. PAIR DD WITH VALUE NN
LD	HL,(NN)	LOAD HL WITH LOCATION (NN)
LD	(HL),R	LOAD LOCATION (HL) WITH REG. R
LD	I,A	LOAD I WITH ACC.
LD	IX,NN	LOAD IX WITH VALUE NN
LD	IX,(NN)	LOAD IX WITH LOCATION (NN)
LD	(IX+D),N	LOAD LOCATION (IX+D) WITH VALUE N
LD	(IX+D),R	LOAD LOCATION (IX+D) WITH REG. R
LD	IY,NN	LOAD IY WITH VALUE NN
LD	IY,(NN)	LOAD IY WITH LOCATION (NN)
LD	(IY+D),N	LOAD LOCATION (IY+D) WITH VALUE N
LD	(IY+D),R	LOAD LOCATION (IY+D) WITH REG. R
LD	(NN),A	LOAD LOCATION (NN) WITH ACC.
LD	(NN),DD	LOAD LOCATION (NN) WITH REG. PAIR DD
LD	(NN),HL	LOAD LOCATION (NN) WITH HL
LD	(NN),IX	LOAD LOCATION (NN) WITH IX
LD	(NN),IY	LOAD LOCATION (NN) WITH (IY)
LD	R,A	LOAD R WITH ACC.
LD	R,(HL)	LOAD REG. R WITH LOCATION (HL)
LD	R,(IX+D)	LOAD REG. R WITH LOCATION (IX+D)
LD	R,(IY+D)	LOAD REG. R WITH LOCATION (IY+D)
LD	R,N	LOAD REG. R WITH VALUE N
LD	R,R'	LOAD REG. R WITH REG. R'
LD	SP,HL	LOAD SP WITH HL

PSEUDO INSTRUCTIONS

ORG NN	SETS LOCATION COUNTER (LC) TO NN
EQU NN	ASSIGNS VALUE NN TO LABEL
DEFS E	INCREMENTS LC BY VALUE OF EXPRESSION E
DEFB E(,E)...	DEFINES BYTE(S) AS E
DEFW E(,E)...	DEFINES WORD(S) AS E
DEFM /S/	ASSIGNS STRING S TO LABEL

APPENDIX G

ZEAP INTERNAL REGISTERS

The contents of a number of memory locations used by ZEAP may be of interest to the user. The user is cautioned to use these registers only as directed. Any uses other than those documented below may cause unpredictable results.

All 16 bit values are stored with the least significant 8 bits first.

EF09 - EFOA BUFP

This 16 bit value is the address of the edit buffer. The first two bytes of the edit buffer itself contain a 16 bit value which is one more than the address at the end of the edit buffer. Thus if BUFP contained 1B0D and 1B0E contained 1B83, then the extent of the edit buffer would be 1B0D to 1B82, and could be dumped under NASBUG control using

>D 1B0D 1B82

or

>W 1B0D 1B83 (using B-Bug or NASBUG 4)

EF22 - EF23 OUTCH

This 16 bit value is the address of the external output routine. It is initially set to the NASBUG entry point, SRLOUT. The user may substitute the address of a routine which outputs the ASCII character contained in register A. All registers must be preserved through this routine, except AF. A routine for driving a high speed parallel printer might be substituted for example. All output from the "U" editor command and under the TTY assembler option is routed through OUTCH, but output from the TAPE assembler option is directly through SRLOUT.

APPENDIX H
ASCII CODE TABLE

All values in hexadecimal. Bit 7 (parity) is zero.

NUL 00	DLE 10	20	0 30	@ 40	P 50	` 60	p 70
SOH 01	DC1 11	! 21	1 31	A 41	Q 51	a 61	q 71
STX 02	DC2 12	" 22	2 32	B 42	R 52	b 62	r 72
ETX 03	DC3 13	# 23	3 33	C 43	S 53	c 63	s 73
EOT 04	DC4 14	\$ 24	4 34	D 44	T 54	d 64	t 74
ENQ 05	NAK 15	% 25	5 35	E 45	U 55	e 65	u 75
ACK 06	SYN 16	& 26	6 36	F 46	V 56	f 66	v 76
BEL 07	ETB 17	' 27	7 37	G 47	W 57	g 67	w 77
BS 08	CAN 18	(28	8 38	H 48	X 58	h 68	x 78
HT 09	EM 19) 29	9 39	I 49	Y 59	i 69	y 79
LF 0A	SUB 1A	* 2A	: 3A	J 4A	Z 5A	j 6A	z 7A
VT 0B	ESC 1B	+ 2B	; 3B	K 4B	[5B	k 6B	{ 7B
FF 0C	FS 1C	, 2C	< 3C	L 4C	\ 5C	l 6C	7C
CR 0D	GS 1D	- 2D	= 3D	M 4D] 5D	m 6D	} 7D
SO 0E	RS 1E	. 2E	> 3E	K 4E	↑ 5E	n 6E	~ 7E
SI 0F	VS 1F	/ 2F	? 3F	O 4F	+ 5F	o 6F	DEL 7F

The following control codes are used by NASBUG:

1D	Backspace
1E	Clear screen
1F	New line

APPENDIX I

OBJECT CODE LISTING

Location 100E, 100F, 1010 & 1011 contain the Ascii equivalent of your copy no. If you enter ZEAP manually from the listing below, please substitute the correct Ascii values for your copy number.

1. ZEAP LISTING

ZEAP 1.0 (C) 1979 SIGMA ACCOUNTING & MGMT SERVICES LTD

01/19/79 2105 HRS

PAGE 1

LOC	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0F00	C3	09	18	1A	18	1A	18	00	-	50	0D	1B	00	00	00	10
0F10	00	A0	20	20	20	20	20	20	-	A0	00	00	10	1B	00	00
0F20	00	C3	22	13	80	00	00	00	-	32	0F	56	0F	21	10	F2
0F30	57	12	56	70	18	55	66	16	-	4E	00	00	52	78	16	4F
0F40	16	41	14	19	46	9C	16	58	-	4E	17	5A	E8	16	49	D5
0F50	50	5E	16	51	56	16	00	68	-	17	91	20	2C	98	3B	00
0F60	2B	29	27	C0	24	A0	23	22	-	80	00					
1000	C3	09	18	5A	45	41	50	20	-	20	20	31	2E	31	2F	30
1010	30	30	20	20	20	20	46	52	-	45	45	20	4D	45	4D	20
1020	54	80	00	01	4C	44	60	83	-	78	04	F4	3E	04	B1	0A
1030	E5	1A	04	F3	3A	14	EC	57	-	14	EE	5F	03	00	83	40
1040	F4	06	03	62	F2	21	04	F3	-	2A	03	6A	F2	31	04	E2
1050	14	F3	7B	03	0C	F2	01	14	-	F3	4B	03	73	E2	22	14
1060	43	04	E0	32	03	31	E0	02	-	03	65	E0	12	13	C9	A0
1070	D2	B0	13	C4	A8	14	D2	B8	-	13	6C	E0	47	13	6E	E0
1080	01	4A	52	F8	18	03	09	F8	-	20	02	50	F2	C3	03	06
1090	C2	43	E3	E9	01	50	55	53	-	48	8F	C5	02	4F	50	8F
10A0	01	43	41	4C	4C	F2	CD	05	-	06	F2	C4	02	50	83	B8
10B0	F4	FE	03	CC	2F	13	C9	A1	-	14	D2	B1	13	C4	A9	14
10C0	B9	02	43	C6	3F	81	45	51	-	D5	00	02	58	68	E8	08
10D0	64	E2	EB	03	6B	E2	E3	03	-	D8	D9	02	C9	FB	01	49
10E0	43	80	04	04	8C	03	03	60	-	F5	DB	14	E7	78	13	00
10F0	40	13	C9	A2	14	D2	B2	13	-	C4	AA	14	D2	BA	12	4D
1100	46	81	44	45	46	C2	04	84	-	D7	05	84	CD	03	84	D3
1110	03	43	80	05	04	8C	0B	02	-	4A	4E	5A	F8	10	02	41
1120	27	02	C9	F3	11	53	42	43	-	62	8C	42	04	60	83	98
1130	F4	DE	02	55	42	83	90	04	-	F4	D6	0A	4C	41	83	20
1140	52	41	83	28	0B	4C	83	38	-	0A	45	54	7A	83	C0	02
1150	C6	37	01	52	45	D4	C9	04	-	86	C0	14	C9	4D	14	CE
1160	0B	53	7A	83	80	0A	4C	43	-	83	00	04	C1	07	0B	83
1170	03	C1	17	13	C4	6F	0A	52	-	43	83	08	04	C1	0F	0B
1180	18	03	C1	1F	13	C4	67	02	-	53	54	FC	C7	01	4F	52
1190	B0	03	F4	F6	83	C7	01	02	-	55	54	75	E0	D3	14	67
11A0	41	14	C9	A3	14	C4	AB	12	-	54	49	D2	B3	13	44	D2
11B0	01	41	44	44	60	83	80	05	-	F4	C6	04	62	8C	09	03
11C0	60	83	88	05	F4	CE	14	62	-	8C	4A	02	4E	44	83	A0
11D0	F4	E6	01	58	4F	52	83	A8	-	04	F4	EE	09	42	49	54
11E0	83	40	11	4E	45	C7	44	02	-	4F	D0	00	01	48	41	4C
11F0	76	00	80	00	01	C8	26	02	-	CC	62	01	C1	60	02	C6
1200	01	C4	22	02	C5	64	01	C2	-	20	02	C3	30	01	DA	2C

ZEAP 1.0 (C) 1979 SIGMA ACCOUNTING & MGMT SERVICES LTD

01/19/79 2105 HRS

PAGE 2

LOC	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
1210	C3	66	01	4E	DA	2A	02	C3	-	2E	01	53	D0	6A	81	A4	00
1220	01	C5	24	01	CC	28	01	C9	-	6C	12	D8	62	32	D9	62	01
1230	CD	38	01	D0	36	02	C5	34	-	02	CF	32	01	D2	6E	00	60
1240	63	28	26	24	22	66	20	38	-	36	34	32	66	2E	2C	2A	6A
1250	62	64	30	68	62	64	30	38	-	3F	12	08	3F	12	38	47	12
1260	34	4B	12	44	4F	12	44	53	-	12	D9	E1	D1	E3	47	14	CB
1270	7E	20	77	23	7E	E6	7F	FE	-	70	38	32	4F	A8	0F	FE	08
1280	30	62	FE	05	38	73	08	FD	-	7E	F5	B7	20	1B	08	FE	06
1290	FD	7E	F4	38	0B	28	0C	FE	-	03	3D	30	0C	3C	28	01	3C
12A0	07	07	07	FD	77	F3	E6	C7	-	C4	18	13	18	4F	FE	20	30
12B0	30	E5	2A	30	0F	85	6F	30	-	01	24	78	08	7E	E6	0F	4F
12C0	7E	08	23	46	23	66	68	06	-	00	ED	B1	E1	47	20	15	08
12D0	CB	21	D6	10	30	FA	CB	39	-	FD	7E	F3	B1	FD	77	F3	18
12E0	1B	B8	28	18	CB	7E	23	28	-	FB	2B	23	23	7E	5F	E6	07
12F0	BA	CA	73	12	23	30	ED	D9	-	C9	FD	77	F2	E3	D5	E5	D9
1300	B7	C9	D9	E1	D1	E3	7E	FE	-	80	23	56	D5	D9	D1	7A	C9
1310	7A	F6	7F	A3	07	9F	92	C8	-	FD	CB	00	4E	C0	FD	CB	F6
1320	F6	C9	D3	01	DB	02	87	F8	-	18	FA	E5	2A	1B	0F	37	ED
1330	52	C1	D5	CD	90	13	38	04	-	23	22	1B	0F	EB	D1	B7	ED
1340	42	C5	E3	C1	C5	ED	B0	CD	-	98	13	C1	C9	21	00	00	1A
1350	D6	30	D8	FE	0A	D0	D5	54	-	5D	29	29	19	29	16	00	5F
1360	19	D1	13	18	EA	2A	0C	0C	-	3A	0B	0C	FE	02	3E	10	20
1370	03	3A	0E	0C	32	0F	0F	7C	-	B5	C0	3A	0F	0F	85	27	6F
1380	7C	CE	00	27	67	C9	23	23	-	AF	47	4F	ED	B1	3D	BE	C9
1390	2A	09	0F	5E	23	56	23	C9	-	2A	09	0F	73	23	72	C9	CD
13A0	51	00	3E	FF	B7	C8	C5	47	-	CD	35	00	CD	F6	18	10	F8
13B0	C1	C9	3E	0D	CD	0A	19	3E	-	0A	CD	0A	19	3E	1F	CD	0A
13C0	19	3A	24	0F	18	DE	C5	2B	-	E5	ED	5B	2E	0F	D5	16	00
13D0	D5	CD	88	14	CB	71	28	0D	-	CD	69	12	30	F4	CD	90	13
13E0	2A	1D	0F	18	2D	CD	02	13	-	38	F3	CB	7B	28	3D	ED	5B
13F0	1F	0F	18	37	CD	88	13	3E	-	30	CA	E3	17	23	CD	88	14
1400	30	F2	13	13	E3	CD	88	14	-	E3	CB	71	28	0A	BE	23	28
1410	F3	C1	C1	C5	C5	18	DD	CD	-	89	14	CB	71	20	F3	DD	E5
1420	E3	37	ED	52	E1	EB	2B	56	-	2B	5E	E1	C1	C1	C9	2A	09
1430	0F	CD	86	13	C8	5E	23	56	-	2B	E5	2A	0C	0C	B7	ED	52
1440	E1	3F	D0	C8	18	EB	23	B7	-	3E	A0	12	C8	2B	CD	82	14
1450	18	F4	CD	71	14	D8	E5	EB	-	CD	32	02	E1	23	CD	88	14
1460	CB	7F	11	00	08	E5	2A	18	-	0C	36	20	EB	22	18	0C	E1
1470	C9	7E	C6	01	9F	D8	5E	23	-	56	2B	E5	2A	0E	0C	ED	52

ZEAP 1.0 (C) 1979 SIGMA ACCOUNTING & MGMT SERVICES LTD

01/19/79 2105 HRS

PAGE 3

LOC	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
1480	E1	C9	7E	12	13	CD	EC	18	-	23	7E	FE	30	38	13	FE	41
1490	38	0A	0E	CO	FE	5B	D8	0E	-	98	FE	A0	C8	0E	EO	FE	3A
14A0	D8	E5	32	69	OF	21	59	OF	-	4E	23	CB	7E	20	FA	BE	20
14B0	F8	E1	CB	61	CO	37	C9	CB	-	59	CO	06	05	CB	61	20	07
14C0	CD	82	14	10	F7	04	C9	13	-	10	FD	CD	89	14	CB	41	C8
14D0	CD	88	14	18	F8	06	00	E5	-	FD	66	F6	2E	70	E3	11	00
14E0	00	CD	89	14	FE	28	20	20	-	E3	CB	54	20	68	CB	D4	2C
14F0	CB	98	E3	CD	88	14	38	1D	-	E3	FD	74	F6	CB	60	20	0A
1500	CB	48	20	0D	ED	53	F2	OF	-	18	07	7B	CD	14	13	FD	73
1510	F7	45	E1	18	B5	ED	53	0D	-	0F	CB	69	28	29	FE	22	20
1520	OF	CD	88	14	16	00	5F	FE	-	A0	20	18	1E	20	2B	18	13
1530	EB	FE	23	28	05	CD	4C	13	-	18	07	13	CD	5A	02	2A	13
1540	0C	EB	2B	E3	18	51	CB	71	-	28	6D	CD	C6	13	2B	E3	20
1550	3E	CB	48	3E	24	20	6E	CB	-	50	20	6A	CB	C8	7A	FE	62
1560	20	21	7B	E6	30	F5	B0	47	-	0F	0F	0F	0F	A5	E6	01	B4
1570	67	F1	CB	7C	20	06	CB	FC	-	B4	67	18	07	AC	E6	30	3E
1580	25	20	42	7D	E6	01	B2	6F	-	ED	5B	0D	0F	C3	FD	14	30
1590	06	CB	40	3E	41	20	2E	E5	-	CB	58	2A	0D	0F	20	03	19
15A0	18	03	B7	ED	52	EB	E1	CB	-	48	28	E1	CB	60	3E	27	28
15B0	14	CB	45	28	10	18	D5	CB	-	D8	E3	FE	2D	CA	F2	14	CB
15C0	49	20	C9	3E	26	C3	E3	17	-	D9	2A	1F	0F	FD	CB	00	4E
15D0	20	6F	FD	CB	01	5E	28	54	-	BF	01	0F	19	ED	43	4B	0C
15E0	ED	4B	0B	0F	5F	20	JA	04	-	05	28	1F	FD	CB	00	5E	20
15F0	20	F5	04	10	02	F1	C9	CD	-	2B	02	CD	3C	02	10	F8	CD
1600	47	16	F1	28	05	EF	2E	1F	-	00	C9	0E	00	CD	32	02	06
1610	08	7B	CD	2B	02	CD	3C	02	-	05	CC	47	16	ED	43	0B	0F
1620	01	E1	18	ED	43	4B	0C	7B	-	FD	CB	00	DE	FD	CB	01	4E
1630	28	08	E5	ED	5B	19	0F	19	-	77	E1	FD	CB	F6	5E	CC	44
1640	02	23	22	1F	0F	D9	C9	79	-	CD	44	02	C3	40	02	3A	0C
1650	0C	FD	77	01	E1	C9	2A	0C	-	0C	22	24	0F	E1	C9	2A	0C
1660	0C	22	19	0F	E1	C9	FD	CB	-	00	D6	CD	9F	13	CD	B2	13
1670	CD	2E	14	CD	BD	1F	18	FB	-	E1	CD	65	13	EB	2A	09	0F
1680	23	CD	87	13	C8	73	23	72	-	EB	CD	7A	13	EB	30	F2	3E
1690	01	32	0F	0F	CD	E7	17	11	-	01	00	18	E1	E1	3A	4C	0B
16A0	FE	20	28	34	21	53	0B	01	-	09	00	ED	B9	23	21	4C	0B
16B0	11	11	0F	ED	B0	1B	3E	A0	-	12	CD	90	13	23	7E	3C	C8
16C0	22	1B	0F	23	23	E5	11	12	-	0F	1A	13	FE	A0	28	13	BE
16D0	23	28	F6	E1	7E	B7	20	EC	-	2A	1B	0F	7E	3C	C4	86	13
16E0	18	DD	E1	2A	1B	0F	18	09	-	E1	CD	2E	14	3E	03	D2	E3
16F0	17	EF	3A	00	CD	BD	18	21	-	4A	0B	36	20	23	E5	7E	E6

ZEAP 1.0 (C) 1979 SIGMA ACCOUNTING & MGMT SERVICES LTD

01/19/79 2105 HRS

PAGE 4

LOC	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
1700	7F	77	11	40	00	19	36	5E	-	CD	3E	00	36	20	D1	D5	21
1710	79	0B	36	20	E5	B7	ED	52	-	E3	C1	FE	3E	20	09	54	5D
1720	2B	ED	B8	23	36	20	2F	FE	-	3C	20	06	62	6B	23	ED	B0
1730	2F	E1	FE	20	20	02	23	2F	-	FE	1D	2D	02	2B	2F	FE	1F
1740	28	26	FE	21	CA	1A	18	B7	-	FA	FD	16	77	18	E8	E1	3A
1750	0B	0C	FE	02	20	1B	CD	2E	-	14	E5	CD	71	14	38	05	CD
1760	86	13	18	F6	D1	C3	2A	13	-	11	4B	0B	D5	CD	4C	13	C6
1770	10	20	6E	D1	CD	5A	02	2A	-	13	0C	7C	B5	28	63	22	0C
1780	0C	21	76	0B	3E	20	36	A0	-	2B	AE	E6	7F	28	F6	E5	B7
1790	ED	52	E5	08	CD	2E	14	54	-	5D	DC	86	13	CD	2A	13	08
17A0	38	24	E1	E5	19	EB	13	13	-	13	E5	2A	07	0F	AF	ED	52
17B0	38	31	CD	96	13	E1	03	ED	-	B8	12	1B	C1	E1	ED	B8	21
17C0	14	0C	ED	A8	ED	AR	2A	0C	-	0C	CD	7A	13	3E	02	38	13
17D0	22	1D	0F	18	49	FD	CB	00	-	E6	CD	65	13	22	1D	0F	18
17E0	3D	3E	99	2A	03	0F	E5	11	-	8F	0B	CD	65	14	5F	EF	45
17F0	52	52	4F	52	20	00	7B	CD	-	44	02	FD	CB	00	FE	7B	FE
1800	23	3E	A0	CC	3B	01	C3	C6	-	18	AF	32	FF	0F	3A	0B	0C
1810	FE	02	20	06	2A	0E	0C	22	-	07	0F	AF	32	FE	0F	AF	67
1820	6F	22	0C	0C	2B	22	0E	0C	-	32	00	0C	3D	32	BA	0B	2A
1830	28	0F	22	45	0C	21	E1	18	-	22	4B	0C	21	1A	18	22	05
1840	0F	22	03	0F	FD	21	FE	0F	-	FD	F9	21	00	10	01	0D	0B
1850	AF	AE	ED	A1	EA	51	18	47	-	2A	2A	0F	7E	21	10	0F	B7
1860	28	07	7E	A8	3E	90	C4	E7	-	17	70	CD	90	13	CD	88	13
1870	28	0A	23	CD	88	14	30	F5	-	13	13	18	F1	EB	22	1F	0F
1880	11	EE	0B	CD	65	14	CD	32	-	02	11	8A	0B	CD	65	14	21
1890	03	10	11	CF	0B	01	1E	00	-	ED	B0	EF	3A	00	FD	CB	00
18A0	66	28	06	2A	1D	0F	CD	32	-	02	CD	DE	01	21	1A	18	E5
18B0	3A	4B	0B	2A	2A	0F	77	FE	-	20	C8	C3	89	02	CD	52	14
18C0	C2	1A	18	CD	46	14	FD	CB	-	00	56	28	13	CD	62	14	11
18D0	8A	0B	1A	CD	0A	19	13	1A	-	B7	F2	D3	18	CD	B2	13	3E
18E0	1F	E6	7F	FE	1D	D4	3B	01	-	FE	21	28	18	3A	25	0F	FD
18F0	CB	00	56	CC	A4	13	CD	4D	-	0C	D0	FE	21	28	06	FE	3F
1900	C0	C3	3E	00	2A	05	UF	E5	-	18	D5	CD	21	0F	18	E7	CD
1910	22	13	18	E2	21	06	1F	22	-	03	0F	2A	30	01	22	45	0C
1920	E1	E1	E1	2E	0B	E5	2A	1F	-	0F	E5	CD	67	19	E1	22	1F
1930	0F	E1	CB	7D	28	04	CB	64	-	28	29	CB	5C	C4	9F	13	7C
1940	E6	0D	6F	CB	55	C4	B2	13	-	E5	21	00	00	22	0B	0F	E5
1950	CD	67	19	E1	E1	3E	50	CB	-	7D	E5	C4	E7	17	E1	CB	5C
1960	C4	D9	15	E5	C3	1A	18	CD	-	90	13	D5	DD	E1	CD	2E	14
1970	2B	22	1D	0F	23	11	8A	0B	-	06	30	3E	2D	12	13	10	FC

ZEAP 1.0 (C) 1979 SIGMA ACCOUNTING & MGMT SERVICES LTD

01/19/79 2105 HRS

PAGE 5

LOC	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
1980	11	98	0B	CD	65	14	CD	52	-	14	CO	F5	30	0F	E5	2A	1F	
1990	0F	DD	75	00	DD	74	U1	DD	-	23	DD	23	E1	FE	3B	28	08	
19A0	06	07	CD	BC	14	CC	B7	14	-	F5	CD	46	14	F1	D1	E5	D5	
19B0	3E	10	C2	E3	17	F1	F5	21	-	00	00	E5	E5	E5	30	0B	21	
19C0	9D	0B	CD	C6	13	3E	31	D2	-	E3	17	FE	3B	28	79	11	8A	
19D0	7B	CD	65	14	2A	1F	UF	CD	-	32	02	2A	2C	0F	E5	16	00	
19E0	D5	21	A3	0B	18	08	CD	69	-	12	3E	20	DA	E3	17	CD	88	
19F0	14	38	F3	CD	CD	14	FD	CB	-	EE	7E	CA	89	1A	3E	40	30	
1A00	C6	CD	02	13	FD	CB	F6	D6	-	38	DF	F5	FE	03	28	3B	30	
1A10	57	F5	06	05	CD	D7	14	3F	-	30	E3	EB	11	8A	0B	CD	65	
1A20	14	CD	32	02	F1	FE	01	28	-	07	38	9C	ED	5B	1F	0F	19	
1A30	22	1F	0F	FD	CB	00	9E	FD	-	CB	F8	46	28	0A	FE	02	28	
1A40	06	DD	75	FE	DD	74	FF	C3	-	F7	1A	46	CD	88	14	B8	28	
1A50	F6	FE	AU	28	F2	CD	C8	15	-	FD	CB	F6	DE	18	ED	CD	10	
1A60	13	08	30	E3	FD	CB	F6	DE	-	06	04	CD	D7	14	08	7B	CD	
1A70	C8	15	FD	CB	F1	46	28	E6	-	7A	CD	C8	15	18	E3	CD	D5	
1A80	14	78	CD	69	12	3E	21	38	-	52	CB	59	28	F1	CD	02	13	
1A90	38	F3	C1	E1	CB	51	28	10	-	2B	2B	FD	CB	01	6E	28	08	
1AA0	D5	ED	5B	1F	0F	ED	52	D1	-	EB	7B	CB	51	C4	14	13	CB	
1AB0	49	C4	10	13	79	B7	7B	08	-	7A	CB	41	EB	E1	E5	37	F5	
1AC0	3F	F5	08	F5	7A	BD	UC	F5	-	3E	CB	CB	5B	28	01	F5	CB	
1AD0	65	28	1A	7C	CB	73	28	08	-	B7	3E	22	C2	E3	17	18	05	
1AE0	C1	CB	45	F5	C5	3E	DD	B5	-	F5	CB	63	20	EC	3E	ED	CB	
1AF0	63	C4	C8	15	F1	30	FA	FD	-	CB	00	46	CC	C6	18	FD	CB	
1B00	F6	76	3E	23	20	D5	31	F8	-	0F	E1	C3	75	19	11	1B	00	
1B10	FF																	

2. ZEAP LOADER PROGRAM

0C50	31	00	50	CD	51	00	CD	3E	-	00	FE	FF	20	F9	06	03	CD
0C60	3E	00	FE	FF	20	F0	10	F7	-	2A	18	0C	36	20	21	8A	0B
0C70	22	18	0C	CD	3E	00	B7	20	-	4E	EF	1F	2E	1F	00	CD	51
0C80	00	31	33	0C	C3	86	02	0E	-	00	CD	3E	00	67	CD	3E	00
0C90	6F	CD	3C	02	CD	3C	02	CD	-	32	02	E5	21	00	08	E5	06
0CA0	08	CD	3E	00	77	CD	2B	02	-	CD	3C	02	23	10	F3	CD	3E
0CB0	00	B9	F5	CD	2B	02	F1	E1	-	D1	28	05	CD	40	02	18	A8
0CC0	01	08	00	ED	B0	18	A1	FE	-	01	28	BC	18	9B	00		

APPENDIX J
ZEAP COMMENT FORM

COPY NO
5

To: Sigma Accounting & Management Services Ltd
c/o Nascom Microcomputers
92 Broad Street
Chesham
Bucks

Date _____

From: Name _____

Address _____

Please tear along dotted line

Comments/Bugs (Fullest possible explanation please including listings - even if written out by hand)

NB: We regret that no correspondence can be entered into over particular queries/suggestions. The aim of the comment form is to enable your opinions, etc to be incorporated in updates.

PTO for more space