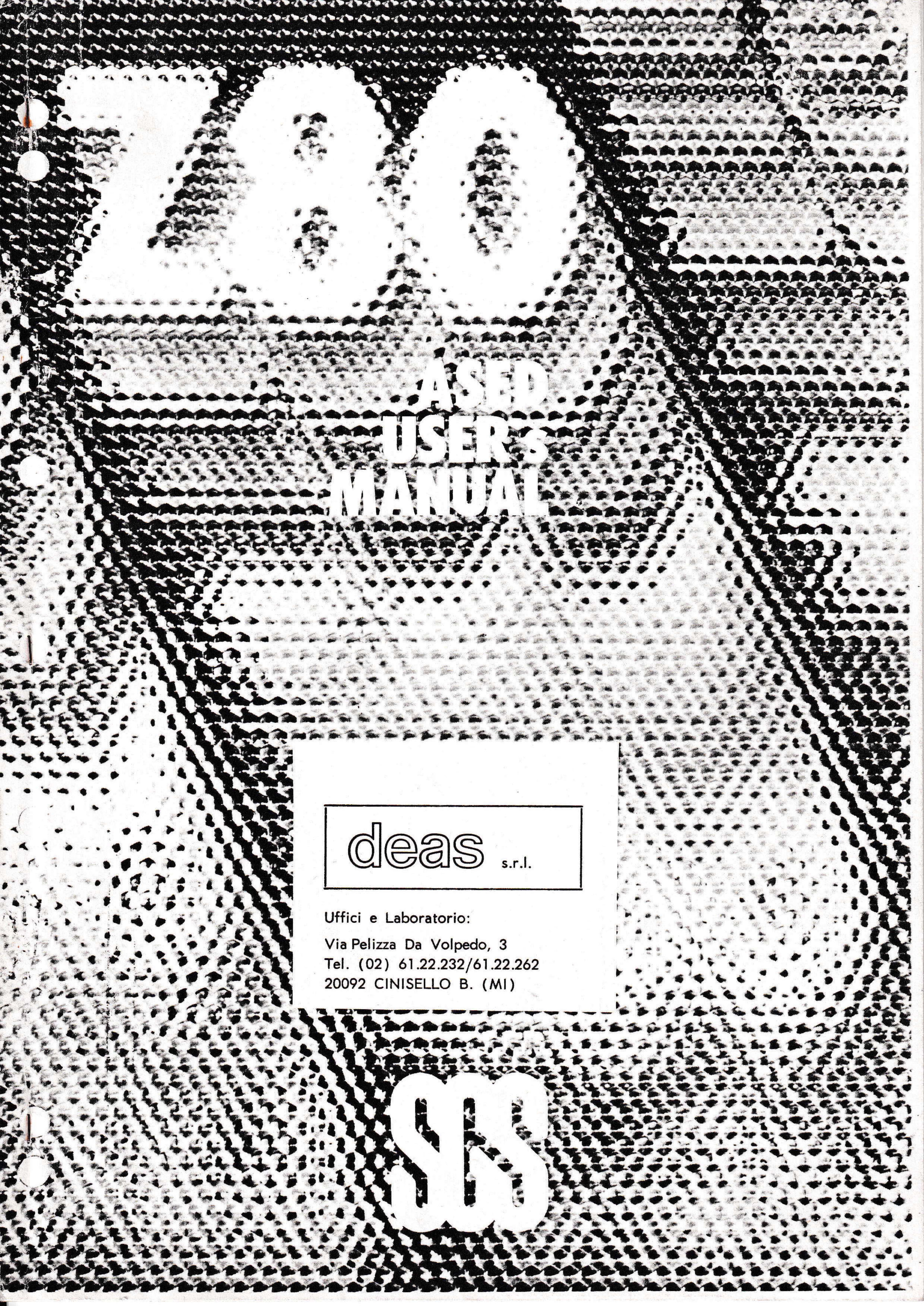

Z80 ASED USER'S MANUAL

deas s.r.l.

Uffici e Laboratorio:

Via Pelizza Da Volpedo, 3
Tel. (02) 61.22.232/61.22.262
20092 CINISELLO B. (MI)

SGS

ASED USER`s MANUAL

Edition 1 February 1982

INTRODUCTION

ASED is the Assembler-Editor Package designed to be used on the NBZ80-ASED Nanocomputer.

The present manual describes the ASED software. For installation or for a description of the system configuration, refer to the:

Z80 Nanocomputer NBZ80-HL and NBZ80-ASED Technical Manual (order code: DANBZHLTM/1).

In particular, refer to the Section 2.2 for the initialisation and to the Appendix A.2 for an example concerning the use of the Assembler and Editor.

Here a list of useful related manuals follows:

- Z80 Nanocomputer Training System Technical Manual (Order Code: DANBZTM/2)

- Z80 Nanocomputer NBZ80-HL and NBZ80-ASED Technical Manual (O. C.:DANBZHLTM/1)

- SGS Basic Language User´s Manual (O. C.:DABASLUM/1)

- UPZ80-HL Assembly Instructions Manual (O. C.:DAUPZ80HLAIM/1)

- EPZ80 EPROM Programmer Technical Manual (O. C.: DAEPZTM/1)

- RCZ80 Technical Manual (O. C.: DARCZTM/1)

Also useful are the following:

- Z80 microprocessor book 1 - Programming (O. C.: DAZ80NPROG/1)

- Z80 microprocessor book 3 - Interfacing (O. C.: DAZ80NINTF/1)

CONTENTS

LIST OF TABLES

CHAPTER 1

THE EDITOR

1.1 Introduction

In the following description the terms "text mode" and "command mode" are used frequently. In text mode lines of text are inserted into the memory buffer. All characters are ignored by the Editor Command Decoder (except ESCAPE and ALT MODE) and are inserted in the buffer as they are entered on the keyboard. In command mode the characters entered via the keyboard are examined by the Editor as prospective commands and, if recognised, are executed. The command mode is indicated by the prompt character "#".

NOTE

In the following sections the notation "CTRL/key" will be used to denote the simultaneous pressing of the keys CONTROL and "key".

1.2 Editor Command Format

In general a command consists of the fields: argument, command, option. Each command is a single character which can be preceded by one or two arguments and can be followed by one option. The command character tells the Editor what to do; the arguments specify which lines are to be modified; the option is related to the specific command.

Table 1.2.a shows the possible command formats. "E" indicates a command character; "m" and "n" the arguments; "o" the option. The Editor indicates that it is in command mode by printing the character #.

Note that unless stated otherwise, every command must be terminated by the CR (Carriage Return) key.

Type	Format	Operation
no argument	E	executes the operation E
one argument	nE	executes E on line n
two arguments	m,nE	executes E from line m to line n inclusive
no argument o X	EX	executes E with option X
two arguments o X	m,nEX	executes E from line m to line n inclusive with option X

Table 1.2.a - Editor Command Formats

The arguments m and n refer to the lines numbered in memory, both must be positive and n must be greater than m. Where two arguments are used they must be separated by a comma and the command E must always come immediately after the argument(s).

The Editor will consider only the first character of the command, the rest of the line will be ignored. Therefore:

```
#L
#LIST
#LIST ALL
```

are all equivalent.

Some of the Editor commands accept a single-character option. Where the options are valid this character must follow the command character.

Example:

```
#WT                                T option for the W command
```

This command tells the Editor to write the buffer in memory into the output device, converting TABs into spaces.

Each time the user requests non-existent information or enters an incorrect command, the Editor will print diagnostic messages. A list of these is given in Section 1-5.

1.3 Commands and special key functions

1.3.1 RETURN (CR or carriage return)

In both command mode and text mode, pressing the RETURN key will cause the Editor to process the line that has just been entered. If the Editor is in command mode the line will be treated as a command and executed, if possible. Commands are not executed until RETURN is pressed unless LINE FEED or the "less than" sign "<" are entered (see 1.3.9). In text mode the line of text will be stored in the memory buffer when RETURN is pressed.

1.3.2 CTRL/U (Erase)

The character "erase" is used to correct errors in both command and text mode. It consists of the keys CTRL (control) and U pressed simultaneously. This special key will delete the line and wait for a new line. The line can then be re-entered correctly.

1.3.3 RUBOUT or DELETE

RUBOUT (DELETE) is used to correct errors in both modes, but is ignored during READ commands. At any time, pressing RUBOUT (DELETE) will delete the last character typed echoing a backslash ( ) followed by the deleted character. Repeated depressions of RUBOUT will delete further characters from right to left up to the beginning of the line.

1.3.4 ALT MODE or ESCAPE

Either of these keys can be used to return the Editor to command mode. If ALT MODE is pressed in command mode it will be ignored. When the Editor is in text mode, the ESCAPE key returns it to command mode ignoring any text preceeding the key. After the commands LIST and WRITE the ESCAPE key interrupts the execution of the command, i.e. it stops the printing or the writing into the output device and the Editor waits for a new command. The operator can choose either to abort the command or continue.

1.3.5 ESCAPE

Escape aborts the operation and returns the Editor to command mode. Any other character makes the Editor recommence execution of the interrupted command.

1.3.6 "." or LINE NUMBER

The Editor maintains a decimal number to indicate which line is being manipulated. At any time, while in command mode, the point "." signifies the current line number and can be used as an argument in commands.

Example:

```
#.L
```

This tells the Editor to print the current line. After an Editor command the line number - and hence the point - depends on the particular command:

1. After READ and APPEND commands the line number is equal to the last line of the buffer
2. After INSERT, CHANGE and OVERLAY commands the line number is set to the last line entered
3. After a LIST command the line number points to the last line in the buffer
4. After a SEARCH command the line number points to the line where the object sought was found
5. After a DELETE command the line number is equal to the line number following the last line deleted
6. After a KILL command the line number is reset.

1.3.7 "/" or SLASH

The slash character represents the last line in the buffer and can be used as an argument, eg:

```
#/L
```

This command makes the Editor list the last line of the buffer.

1.3.8 LINE FEED

Pressing the line feed key in response to the command mode prompt makes the Editor print the line after the current line and increments the line number ".". This means that the line number will then be at the line which has just been printed. If the line feed key is pressed while a command or a line of text is being entered (i.e. before RETURN is pressed) the Editor will reprint the line. This is very useful for verifying lines when the RUBOUT key has been used.

1.3.9 "<" or Less Than

If this key is pressed the Editor prints the line before the current line. The line number will be decremented so that it indicates the line number of the line which has just been printed.

1.3.10 "=" or Equal

This symbol is used with the point "." and the forward slash "/". If it is pressed after the point the number of the current line is printed. If, on the other hand, it follows a forward slash the line number of the last line in the buffer is printed.

Examples:

```
#.=                          the current line number is printed
XXXX

#/=                          the number of the last line in the
XXXX                         buffer is printed
```

1.3.11 **CTRL/G**

If the control-G (BELL) key combination is pressed after a string search the Editor will look for the next occurrence of the same string in the buffer.

Example:

```
#S                           SEARCH command
HL                           for the string ´HL´
LD       HL,BUFF             the string has been found in this line
#CTRL/G                      continue the search
LD       A,(HL)              another line containing ´HL´ is found
#CTRL/G                      continue again the search
NO MATCH                     there are no more ´HL´ strings in the buffer
```

1.3.12 **CTRL/TAB** or Tabulation

The Editor is arranged so that it can simulate horizontal tabulation with intervals of 8 spaces. When the user holds down the CTRL key and presses the TAB key (or HT) the Editor tabulates one position, i.e. it shifts the carriage to the next tabulation position by moving between 1 and 8 spaces as necessary. In the memory buffer, the Editor stores only the TAB code (saving memory space) and this will be written into the output device unless the option "T" is specified.

1.3.13 **CTRL/C**

This key combination interrupts the Editor and returns control to the Monitor NC-Z.

1.4 Editor Commands

1.4.1 APPEND

#A

This command is used to enter a new text via the keyboard or to add it to the end of an existing text. When the APPEND command is received the Editor enters text mode and the user can enter any number of lines of text.

The new text will be added to the information already in the buffer until the user terminates the text mode by pressing ESCAPE or ALT. If ESCAPE is pressed at the end of a line the Editor will return to command mode and ignore that line.

If the key combination CTRL/U is entered while a line of text is being written, the line will be cancelled and the user can rewrite it. In text mode the RUBOUT key will cancel the last character. Pressing RUBOUT more than once deletes further characters from right to left until the beginning of the line is reached.

1.4.2 LOCATE Buffer

#B
STRING

The user types "B" followed by CR (carriage return) and STRING also followed by CR. STRING represents a string of characters which is separated from the command B by a space and terminated by carriage return.

When the Editor receives this command it will search through the entire file for the buffer that contains the string.

When (and if) the Editor locates the string it prints the line that contains it and returns to command mode. In the execution of this command the Editor searches for the string in the buffer and if it is not found the buffer will be transferred to the output file, cancelled and new text will be read in from the input device. The process is repeated - search, transfer, delete, read - until either the string is found or the END OF TEXT (Form Feed) is encountered. The last buffer (or the one in which the string was found) is not transferred to the output device.

"NO MATCH" will be printed if the string is not found.

1.4.3 CHANGE

```
#nC
OLD STRING
NEW STRING
NEW LINE IS PRINTED
```

With this command the Editor searches for the string OLD STRING in line n (the current line by default) and replaces it with the string NEW STRING. The new (modified) line is then printed.

If OLD STRING is not found in the specified line, the message "NOT MATCH" will be printed and the Editor will return to command mode.

Examples:

```
#.L
HAPPY DAYS WERE HERE ONCE
#C
WERE HERE ONCE
ARE HERE AGAIN
HAPPY DAYS ARE HERE AGAIN

#.L
IS AN EXAMPLE
#C
T
THIS
THIS IS AN EXAMPLE

#.L
YOU ARE WRRIGHT
#C
WR
(empty line)
YOU ARE RIGHT

#.L
EXAMMINE THE TEXT
#C
AMI
NO MATCH
#
```

1.4.4 DELETE

#nD

This deletes the line n. This line is removed from the memory buffer and the line numbers of all the successive lines are decremented by one.

#m,nD

With both arguments the lines from m to n (n>m) inclusive are deleted. The current line will then be the one immediately after the last line deleted.

The Editor remains in command mode after DELETE operations.

#D	is equivalent to:
#.D	and deletes the current line.
#Dn	deletes n lines starting from the current line.

1.4.5 EXIT

#E

The Editor writes the current buffer into the output device and transfers all the text from the input device to the output device. Control then returns to the Monitor NC-Z.

NOTE

> The READ command must be used at least once before EXIT can function properly, tranferring the input data from the mag/paper tape to the output device; otherwise EXIT behaves like a QUIT command loosing the remaining information at the input device.

The EXIT command can have one or two arguments.

#nE

#m,nE

In the first case only line n of the current buffer is transferred to the output file; in the second the lines from m to n are transferred. After this the transfer of all the text from the input device to the output device is effected.

See Section 1.4.15 for the option "T".

1.4.6 INSERT

#nI

This is used to insert new text before line n, up until **ESCAPE** or **ALT** is pressed. The Editor enters texts mode to acept the input and functions like APPEND as the text is being inserted. The first line to be entered becomes the new line n.

The line numbers of all the subsequent lines are incremented by the number of lines inserted and the current line pointer "." will be the number of the last line inserted. If the line number in the command is omitted the current line is assumed by default.

#I	inserts text before the current line
#.I	as **I**
#nI	inserts text before line n

1.4.7 KILL

#K

KILL?

The KILL command deletes all the text in the buffer. After receiving this command the Editor prints "KILL?" and waits for confirmation before executing the command. This is to minimize the chances of deleting the buffer accidentally If the response to this prompt is "Y" (YES) the command is executed and the Editor returns to command mode. If any other character is entered the KILL command is ignored and the Editor just returns to command mode.

1.4.8 LIST

#L	writes the entire buffer on the terminal
#nL	writes line n on the terrminal
#m,nL	writes lines m to n on the terminal
#Ln	writes n lines starting from the current line

The LIST command updates the current line pointer and the Editor returns to command mode.

1.4.9 NEXT

#N

The Editor writes the current buffer into the output device, deletes the buffer, reads new text from the input device to refill it and returns to command mode.
NEXT can have one or two arguments.

#nN

#m,nN

In the first case it writes only line n of the buffer into the output device; in the second it writes from line m to line n. The buffer is then completely deleted and new text is read from the input device.

See Section 1.4.15 for the option "T".

1.4.10 OVERLAY

#nO

With this command the Editor replaces line n (or the current line if n is omitted) with as many lines as are entered by the user.

This command is equivalent to:

#nD
#nI

Where n coincides with the last line to be cancelled and the new text added on the end.

#O	replaces the current line with new text
#.O	same as #O
#nO	replaces line n with new text
#On	replaces n line starting from the current line with new text
#mOn	n lines starting from the line number m are replaced with new text

1.4.11 QUIT

#Q

Whit this command the Editor writes the current buffer into the output device and the Editor ends. Control returns to the Monitor NC-Z.

See Section 1.4.15 for the option "T".

1.4.12 READ

#R

The Editor reads text from the input device until it encounters the code FORM FEED (corresponding to the key combination CTRL/L) or, alternatively, until the memory buffer is filled.

The READ command can be issued both when the buffer is empty and when it already contains some text. In the latter case all the text read, except the FORM FEED code, is added to the contents of the memory buffer.

During a READ command the RUBOUT, NULL and LINE FEED keys are ignored by the Editor. The Editor will also, during the READ command, declare that the buffer is full when the last line loaded leaves spaces for about 100 characters. This allows a limited number of extra characters to be inserted before the buffer is completely full.

The Editor prints:

```
XXXX LINES READ IN
```

where XXXX is the number of lines in the buffer and returns to command mode. At this point, every subsequent READ command will make the Editor reprint the message given above.

#RH

This is a command for the exclusive use of the system and it is not available to the user.
As with the command R without option except the Editor reads text from the high speed paper tape reader untill it encounters the code FORM FEED or alternatively untill the memory buffer is filled.

1.4.13 RETURN TO MONITOR NC-Z

#CTRL/C

This command aborts the Editor. Control returns to the Monitor NC-Z.

1.4.14 STRING SEARCH

#S STRING

STRING is a character string separated by a space from the command and terminated by carriage return. This command searches for a string of characters in the buffer. Every line of the buffer is examined and if STRING is found the line is printed and the Editor returns to command mode. If the string is not found the Editor prints "NOT MATCH" and returns to command mode. When the string is found the current line pointer "." is set to the line where the string was found (ie. the line that was printed).

1.4.15 WRITE

#W	writes the entire memory buffer into the output device
#nW	writes line n from the memory buffer to the output device
#m,nW	writes lines m to n from the memory buffer to the output device

The WRITE command updates the current line pointer and the Editor returns to the command mode.

An option "T" can be used with commands that write into the output device: W, N, Q and E. It must be entered immediately after the first character of the command.

Examples:

#WT **#QT** **#NT** **#ET**

This option converts all the TAB´s into the necessary spaces during the transfer of text from memory buffer to the output device.

1.5 Diagnostic Messages

SYNTAX or LINE ERROR	- The user has entered an illegal command - The argument(s) of a command is/are invalid(s) - The user has requested non-existent information (eg. a listing of a non-existent line) - In either mode, the line has more than 80 characters or no characters. The same applies during a READ command
KILL?	-After a KILL command, if the user presses "Y" (and return), the Editor deletes the buffer and returns to command mode. Any other character will cause the Editor to return to command mode leaving the buffer intact.
NO MATCH	- After a SEARCH or CHANGE command means that the string was not found.
BUFFER FULL	- The Editor can accept no more text. The user must therefore write all or part of the buffer into the output device and then free space in the buffer by using KILL or DELETE.
BUFFER EMPTY	- The buffer contains no text. This is given in response to commands which refer to the buffer (ie. L, I, C, O etc).
XXXX LINES READ IN	- Occurs after a READ command XXXX is the total number of lines transferred into the buffer.

After writing any of these messages the Editor returns to command mode.

1.6 Summary of the commands

A - adds symbolic text
B - searches for a string in the file
C - replaces one string with another one
D - deletes lines
E - exits from Editor
I - inserts before the current line
K - deletes the memory buffer
L - lists lines of the memory buffer
N - dumps the current buffer and reads new text
O - replaces lines with new text
Q - dumps the current buffer and closes the file on the output
    device
R - reads from the input device, filling the buffer
CTRL/C - aborts the Editor
S - searches for a string in the buffer
W - writes the current buffer into the output device

CHAPTER 2

THE ASSEMBLER

2.1 Assembly Language Instructions

The format of the assembly language instructions is the same as that found in most assemblers. There are four fields: label, operator, operand and comment. The label and comment fields are optional; the operator field is obligatory; the operand depends on the operator. The fields are separated by one or more spaces or tabs and the comment field must begin with a semi-colon. Labels can be up to 6 characters in length; the first must be alphabetic but the remainder may be alphabetic or numeric. Label definitions starting in column 1 can optionally be followed by a colon. Label definitions not starting in colomn 1 must be followed by the colon character.

Example:

```
Label   Op´tor  Op´nd                     Comment
-----------------------------------------------------------------------

                                  ; an instruction can be formed like
                                  ; this using only the comment field.
                                  ; In this case no object code is
                                  ; geneated. In practice these pseudo
                                  ; instructions are only used to make
                                  ; the program better documented
                                  ;
        LD      A,10              ; label field not used
                                  ;
        RLA                       ; label  used and operand field not
                                  ; field not required
                                  ;
TEST    LD      B,5               ; the label TEST is defined here
                                  ;
        JP      TEST              ; reference is made to the label TEST
                                  ; defined in another instruction
                                  ;
-----------------------------------------------------------------------
```

```
Label    Op'tor  Op'nd                 Comment
-------------------------------------------------------------------------------
                                       ;
 BETA:   LD      B,A                   ; the label BETA is defined here;
                                       ; As BETA begins not in colomn 1,
                                       ; the colon is obligatory to define
                                       ; correctly the label
                                       ;
         ADC     A,(HL)                ; the operand field is formed by A and
                                       ; (HL) separated by a comma; for the
                                       ; format of the operand(s), see the
                                       ; Z80 INSTRUCTION SET manual
                                       ;
ALFA     LD      A,(IY+4)              ; the label ALFA is defined here;
                                       ; this instruction needs 3 operands:
                                       ; A, IY  and  +4
                                       ;
         JR      ALFA                  ; JR is the mnemonic for the relative
                                       ; jump; ALFA is the address to which it
                                       ; jumps
                                       ; this address is calculted by the
                                       ; Assembler by subtracting from the
                                       ; address of this instruction the
                                       ; address of the instruction where
                                       ; ALFA is defined; the result will be
                                       ; inserted into 1 byte. Its value must
                                       ; be in the range: -128  +127
                                       ;
         JP      TEST                  ; JP is the mnemonic for the jump
                                       ; instruction;
                                       ; the destination address is found
                                       ; by searching for TEST in the
                                       ; symbol table;
                                       ; the address is 16 bits long
                                       ;
ALFA?    JP      TEST                  ; an error message will appear on
                                       ; the screen, as the character '?'
                                       ; is not valid
                                       ;
ABCDH:   INC     A                     ; valid label; the colon is optional
                                       ;
         LD                            ; error: the operand is missing
                                       ;
LABELLL: LD      A,B                   ; error; LABELLL has more than six
                                       ; characters
                                       ;
LA-BEL:  LD      (IX),B                ; error; the label LA-BEL contains the
                                       ; non-alphanumeric character " - "
-------------------------------------------------------------------------------
```

2.2 Writing the instructions

For the first three fields upper-case ASCII characters are used but lower case may be used in comments. For numbers the following should be noted:

a) numbers expressed in decimal have no special character appended

ex. 35 6454

b) hexadecimal numbers must end in H and must begin with a character from 0 to 9. When the number begins with A-F , it must be preceeded by 0

ex. 2AH 0A57BH 27H

c) data to be stored in one byte must be less than or equal to 255 (decimal) or FFH (hexadecimal). Numbers to reside in two bytes must be less than or equal to 65535 (dec) or FFFFH (hex). The Assembler will signal an error if this rule is not respected.

d) to introduce ASCII characters they must be preceded and followed by the character "´".

Example:

```
        LD      A,´C´           ; the ASCII code for ´C´ is loaded
                                ; into the accumulator. This is
                                ; equivalent to:
        LD      A,43H

        LD      HL,´GM´         ; the codes for ´GM´ are loaded
                                ; into the register pair HL.
                                ; This is equivalent to:
        LD      HL,474DH
```

For the mnemonics and other rules the Zilog standard is used. See "SGS-ATES INSTRUCTION SET MANUAL".

2.3 Pseudo Instructions

The pseudo-instructions are directives useful to simplify the programmer´s work. Seven pseudo-instructions are recognised by the Assembler. These are:

DEFB	define byte
DEFM	define message
DEFS	define storage
DEFW	define word (2 bytes)
END	end
EQU	equal to
ORG	origin

DEFB

DEFB is used to define the contents of the current address count.
Example:

```
ORG     100H
DEFB    3H
DEFB    20H
DEFB    7FH
DEFB    134
DEFB    ´2´
DEFB    ´B´
```

This will result in:

Address	Content	
100H	03H	starts at 100H, see ORG
101H	20H	
102H	7FH	
103H	0A2H	A2H=134 decimal
104H	32H	32H is ASCII code for 2
105H	42H	42H is ASCII code for B

DEFM

DEFM defines an ASCII character string. It is used principally to define messages to be printed and look-up tables. The ASCII character string must be preceeded and followed by the character "´".

Example:

```
        DEFM    ´ABCDEF´
```

This instruction will define six bytes containing the ASCII codes of the characters ABCDEF, ie. 41H, 42H, 43H, 44H, 45H and 46H. The character "´" is used simply as a string delimiter.

DEFS

DEFS reserves an area of memory leaving it unchanged when loading the object file. Usually this pseudo intruction is used to reserve buffer areas which will be filled during execution but obviously not at the assembly stage.

Example:

```
        ORG     0FFH     ; set the address counter to FFH
        DEFB    0AAH     ; AAH is placed at the address FFH
                         ; and the address is incremented to 100H
        DEFS    80H      ; 80H (128 decimal) bytes of memory are
                         ; reserved. An example of this would be
                         ; a sector buffer for floppy disk read
                         ; operations.
                         ; At this point the address counter is
                         ; incremented to 180H
        DEFB    0BBH     ; BB hex is stored at location 180H
        DEFB    0CCH     ; CC hex is stored at location 181H
```

DEFW

DEFW defines a word, i.e. two bytes, at the current address counter and the following location.

Example:

```
        ORG     100H
        DEFW    1234H    ; 34H is stored at 100H and 12H at 101H.
                         ; Note that the low-order byte always comes
                         ; first in Z80 programming.
```

END

END pseudo instruction is obligatory at the end of every symbolic file. It is used by the Assembler to decide when the end has been reached and to stop the scanning of source code. If an address is specified as an argument it will generate an autostart record. The autostart record is used to start execution of an object file automatically after loading.

Example:

```
        ...                          ; instructions
        ...                          ;
INIT:   ...                          ; starting point of the program
        ...                          ;
        ...                          ;
        ...                          ;
        END      INIT                ; program end and autostart address
```

EQU

With this pseudo instruction it is possible to equate a label to a numeric value and subsequently use it in the program.

Example:

```
GOON:   EQU      1000H
BEGIN:  EQU      200H
BYT:    EQU      7AH
WOR:    EQU      8BA7H
                                              ;
        ORG      BEGIN                        ;
        LD       A,BYT                        ; BYT is 1 byte
        LD       HL,WOR                       ; WOR is 1 word (2 bytes)
        JP       GOON                         ; GOON address is 1 word
```

This is equivalent to:

```
        ORG      200H                ; origin at 200H
        LD       A,7AH               ; load A with 7A hex
        LD       HL,8BA7H            ; load HL with 8BA7 hex
        JP       1000H               ; jump to 1000 hex
```

The use of EQU brings two advantages:

1. Instructions can be documented better by using mnemonics instead of numbers

   Example:

```
   MOTOR:       EQU  5               ; port 5 controls a motor
   START:       EQU  5AH             ; port configuration to start
                                     ; the motor

                LD   A,START         ; prepare A
                OUT  (MOTOR),A       ; send the contents of A to the
                                     ; port which controls the motor
```

2. The program can be parametrised, ie. the data bytes that determines a certain configuration can be equated to a label. Each time the configuration has to be modified all that need be done is to change the EQU instructions.

   Example:

```
MEMORY:     EQU  850H              ; defines the value of the
                                   ; label MEMORY
            ORG  MEMORY            ; sets the address counter to
                                   ; the value of memory
                                   ; that is 850H

VALUE:      DEFB XX                ; inserts a number at the current
                                   ; addrress

COUNT:      DEFB YY                ; inserts a number in the next
                                   ; location
            ...
            ...
            LD   A,(VALUE)         ; VALUE is found at the address
                                   ; 850H
            ...                    ;
            INC  (COUNT)           ; COUNT is found at the address
                                   ; 851H
            ...                    ;
```

If it is required to move the area of memory used, MEMORY can be redefined by changing the instruction:

```
MEMORY:    EQU  XXX
```

On reassembly, all the instructions referring to this area will be changed to suit the new addresses.

ORG

This pseudo instruction redefines the initial setting of the address counter. If ORG is missing assembly will start from the address 0000H.

2.4 Printer Format

The Assembler generates a line of text for every instruction. This line is subdivided into five fields:

Error field
Source code line number field (in decimal)
Address field (in hexadecimal)
Object code field (in hexadecimal)
Source statement field.

The error field is occupied by a character identifying the error type whenever an error is found in the assembly of the line in which it appears.

The line number field maintains a progressive and continuous count of the lines of source code which is useful when the Editor is used to modify source statements.

The address field contains the address of the first byte of the object code generated for each instruction. The possible 1, 2 or 3 remaining bytes of the instruction follow in the object code field. The object code field contains the assembled code in hexadecimal. The source statement field contains, unchanged, the source instruction. Some pseudo instructions do not generate any object code and consequently have blanks in the related address and object code fields Instructions consisting only of comments do not generate code either so the same applies.

2.5 Available routines

In the present section the software routines available to the user are described, with their entry point and main features.

2.5.1 WRITE

Entry point: E3B3 hex

Purpose: to serially transmit a byte

Description: The byte to be transmitted is contained in the A register. A test is performed on the content of the memory location pointed to by the location PRINT (0176 hex); the transmission is performed only if its content is different from zero.

Registers altered: none

2.5.2 READX

Entry point: E3B6 hex

Purpose: to serially receive one byte

Description: the byte is stored in the A register. A lower-case character is converted into the corresponding upper-case character.

Register altered: A

2.5.3 RTLFX

Entry point: E3B9 hex

Purpose: to serially transmit the sequence "Carriage Return, Line Feed"

Register altered: A

2.5.4 WRITEY

Entry point: E3C8 hex

Purpose: to send a byte in output

Description: the content of the A register is sent to the serial interface, to the cassette or to the printer, according to the content of the memory location pointed to by the FLAOUT location (0169 hex).

Registers altered: none

2.5.5 TAPEX

Entry point: E3BC hex

Purpose: to read a byte from an input

Description: the A register is loaded with the content of the serial interface or the cassette, according to the content of the memory location pointed to by the FLAINP location (0168 hex)

Register altered: A

2.5.6 DELAY

Entry point: E3C5 hex

Purpose: to create a software-controlled delay

Description: a delay is created, whosw value is 400 msec times the content of the A regikster

Register altered: A is cleared.

2.5.7 PRINTX

Entry point: E3DD hex

Purpose: to printout on the line printer

Description: sends the content of the A register on the line printer according to the control keys used to enable disable the printing

Register altered: none

2.6 Errors

The error listing onto the console or onto the printer has the format:

```
XX      YYYY    ZZ....
```

where

XX	= error code
YYYY	= line number where the error has been detected
ZZ...	= source statement

The error codes are as follows:

IC	syntax error
ID	field separator error
IN	error in the END argument
IT	label definition error
DT	label duplicated
TT	too many symbols: insufficient room in the symbol table. In this case the number of labels must be reduced.

APPENDIX A

EXAMPLES

A.1 EDITOR

The system is initialised as explained in the Section 2.2 of the Z80 Nanocomputer NBZ80-HL and NBZ80-ASED Tecnical Manual. The messages in bold are written by the system, the other by the operator.

PERIPHERAL ASSIGNEMENT

The Editor gains the control of the system and types:

```
SGS/ATES EDI-Z/N REL.1.0
INPUT (L,H,C)=
```

the operator types L for input from keyboard

```
INPUT (L,H,C)=  L
OUTPUT(L,H,C)=
```

the operator types C for output on the cassette

```
OUTPUT(L,H,C)=  C
#                       the Editor types the prompt character
```

CREATION OF A NEW TEXT

---

the operator types the command A to append a new text to the buffer that in our case is empty

```
# A
```

the Editor is in input mode and the operator inserts the text. Each line is terminated by the Carriage Return

```
        LD      SP,0C80H
        LD      HL,0A800H
        LD      A,20     ; comments
        LD      B,0FH    ; etcetera
        ...     ...
        ...     ...
        ...     ...
        JR      LAB1
        END
```

the operator exits from input mode typing the ESCAPE command key. The Editor answers with the prompt character (#)

```
#
```

LISTING THE BUFFER

---

the operator types L to list the whole buffer on the console

```
# L
```

the Editor lists on the console the complete text buffer

```
        LD     SP,0C80H
        LD     HL,0A800H
        ...    ...
        ...    ...
        ...    ...
        JR     LAB1
        END
#
```

the Editor ends typing the prompt character(#).

The operator asks the list of the line number 12 of the text buffer by typing 12L

```
# 12L
```

the Editor types the line #12

```
        BIT    2,A

#
```

the operator asks the list of lines #8 and #9 by typing 8,9L

```
# 8,9L

          POP    DE
          INC    BC

#
```

the line number is positioned on line #9

MODIFYING A TEXT

---

the operator inserts new text before the line #9 by typing I

```
# I
```

the Editor is in input mode, new text can be inserted

```
          LD      (HL),B
          INC     HL
          LD      (HL),C
```

the operator ends the input mode by typing ESCAPE

```
#
```

the operator queries the actual line number typing .=

```
# .=
```

the Editor answers with

```
0011
```

starting from line #9, 3 new lines have been joined

the operator asks the list of the lines inserted

```
# 8,12L

          POP    DE
          LD     (HL),B
          INC    HL
          LD     (HL),C
          INC     BC

‡
```

the operator asks how many lines are in the text buffer by typing /=

```
‡  /=
```

the Editor answers with

```
0021
```

21 lines are in the buffer position the line counter to the beginning

```
‡  1L
```

now the operator asks to the Editor to search the string 7,A in the text buffer by typing the command S for search then Carriage Return and the string requested

```
‡  S

7,A
```

the Editor searches the string 7,A and prints the line when it has been found

```
LAB3    BIT      7,A
‡
```

the operator wants to change the string 7,A with 6,A by typing C , Carriage Return, the old string (7,A) and the new string (6,A)

```
‡  C
7,A
6,A
```

the Editor answers with the modified string

```
LAB3    BIT      6,A
‡
```

the operator ends the Editor session by typing the command E to exit

```
‡  E
```

the Editor writes the text buffer on the output cassette and returns control to the Monitor NC-Z.

Now the operator can pass to the Assembly phase entering the Assembler or go back to the Editor entering again the Editor as explained at the beginning of the present Appendix. Remember that if he reenters the Editor he probably will work with input from the cassette and output on the cassette.

A.2 ASSEMBLER

The Assembler receives the control and types:

SGS/ATES ASS-Z/N REL.1.0	
IN (L/H/C)=	
	the operator selects the source input from cassette and types C
IN (L/H/C)= C	
	the Assembler asks for the output device of the object code
OUT(L/H/C)=	
	the operator selects the cassette and types C
OUT(L/H/C)= C	
	the Assembler asks for the listing device
LIS (L/H/C)=	
	the operator selects the printer and types L for the high speed printer
LIS (L/H/C)= L	
	the Assembler asks the number of the pass
PASS?	
	the operator types 1 for pass 1
PASS? 1	
	the Assembler lists the errors on the console
IC 0021 **POP YY**	
	the error occurred on the instruction at line 21: POP YY has an illegal operand; the Assembler asks for the next pass
PASS?	
	the operator selects the pass 2
PASS? 2	
	the Assembler types the number of errors detected: in this case 2 errors were detected
ERR 0002	
	the Assembler asks·for the next pass
PASS?	
	the operator asks for the object code and types 3 for this operation
PASS? 3	
	at the end the Assembler will ask for the next operation
PASS?	
	the operator exit by typing CTRL/C: the control will return to the Monitor MC-Z.

SGS-ATES GROUP OF COMPANIES

INTERNATIONAL HEADQUARTERS
SGS-ATES Componenti Elettronici SpA
Via C. Olivetti 2 - 20041 Agrate Brianza - Italy
Tel.: 039 - 65551
Telex: 330131-330141

BENELUX
SGS-ATES Componenti Elettronici SpA
Benelux Sales Office
B- 1180 Bruxelles
Winston Churchill Avenue, 122
Tel.: 02 - 3432439
Telex: 24149 B

DENMARK
SGS-ATES Scandinavia AB
Sales Office
2730 Herlev
Herlex Torv, 4
Tel.: 02 - 948533
Telex: 35411

EASTERN EUROPE
SGS-ATES Componenti Elettronici SpA
Export Sales Office
20041 Agrate Brianza - Italy
Via C. Olivetti, 2
Tel.: 039 - 6555287/6555207
Telex: 330131-330141

FINLAND
SGS-ATES Scandinavia AB
Sales Office
02210 Esbo 21
Kääntöpiiri, 2
Tel.: 90 - 881395/6
Telex: 123643

FRANCE
SGS-ATES France S.A.
75643 Paris Cedex 13
Résidence "Le Palatino"
17, Avenue de Choisy
Tel.: 01 - 5842730
Telex: 042 - 250938

WEST GERMANY
SGS-ATES Deutschland Halbleiter
Bauelemente GmbH
8018 Grafing bei München
Haidling, 17
Tel.: 08092-691
Telex: 05 27378
Sales Offices:
3012 Langenhagen
Hubertusstrasse, 7
Tel.: 0511 - 772075/7
Telex: 09 23195
8000 München 90
Tegernseer Landstr., 146
Tel.: 089 - 6925100
Telex: 05 215784
8500 Nürnberg 15
Parsifalstrasse, 10
Tel.: 0911 - 49645/6
Telex: 0626243
7000 Stuttgart 80
Kalifenweg, 45
Tel.: 0711 - 713091/2
Telex: 07 255545

HONG KONG
SGS-ATES Singapore (Pte) Ltd.
9th Floor, Block N,
Kaiser Estate, Phase III,
11 Hok Yuen St.,
Hung Hom, Kowloon
Tel.: 3-644251/5
Telex: 63906 ESGIE HX

ITALY
SGS-ATES Componenti Elettronici SpA
Direzione Commerciale Italia
20149 Milano
Via Correggio, 1/3
Tel.: 02 - 4695651
Sales Office :
00199 Roma
Piazza Gondar, 11
Tel.: 06 - 8392848/8312777

SINGAPORE
SGS-ATES Singapore (Pte) Ltd.
Singapore 1231
Lorong 4 & 6 - Toa Payoh
Tel.: 2531411
Telex: ESGIES RS 21412

SWEDEN
SGS-ATES Scandinavia AB
19501 Märsta
Box 144
Tel.: 0760 - 40120
Telex: 042 - 10932

SWITZERLAND
SGS-ATES Componenti Elettronici SpA
Swiss Sales Offices
6340 Baar
Oberneuhofstrasse, 2
Tel.: 042 - 315955
Telex: 864915
1218 Grand-Saconnex (Geneve)
Chemin François-Lehmann 22
Tel.: 022 - 986462/3
Telex: 28895

UNITED KINGDOM
SGS-ATES (United Kindgom) Ltd.
Aylesbury, Bucks
Planar House, Walton Street
Tel.: 0296 - 5977
Telex: 041 - 83245

U.S.A.
SGS-ATES Seminconductor Corporation
Scottsdale, AZ 85251
7070, East 3rd Avenue
Tel.: (602) 990-9553
Telex: SGS ATES SCOT 165808
Waltham, MA 02154
240, Bear Hill Road
Tel.: (617) 890-6688
Telex: 923495 WHA
Des Plaines, IL 60018
2340, Des Plaines Ave Suite 309
Tel.: (312) 296-4035
Telex: 282547
Santa Clara, CA 95051
2700, Augustine Drive
Tel.: (408) 727-3404
Telex: 346402
Woodland Hills, CA 91367
6355, Topange Canyon Boulevard
Suite 220
Tel.: (213) 716-6600
Telex: 182863
Orlando, FL 32792
1309, South Semoran Blvd.
Lakeview, 436 Office Park
Tel.: (305) 671-8599

SGS-ATES GROUP OF COMPANIES
SYSTEMS DIVISION

Italy – France – Germany – Malta – Malaysia – Singapore – Sweden – United Kingdom – U.S.A.