Z8E USER´S MANUAL

by
RICK SURWILO
23 PLMOUTH ROAD
STAMFORD,CONNECTICUT 06906
(203) 324-3809

TABLE OF CONTENTS

Z8E - Z80 DEBUGGING MONITOR

I. INTRODUCTION

Z8E is a professional quality interactive debugging tool designed to speed the testing of Z80 assembly language programs. Originally written as a standalone monitor, Z8E was used in the developement of the world′s largest Touch-Tone Input/Voice Response system. Now redone to run in a CP/M or TurboDOS environment Z8E contains more features in less memory than any comparable software product. Occupying less than 9K of memory, Z8E includes the following among its many features:

- Full screen animated display of the program under test while it is being executed by the Z80

- Complete Z80 inline assembler, with labels, symbols, expressions, and directives, using Zilog mnemonics

- Interactive disassembly with labels and symbols to console or disk allows the user to specify output formats and add comments

- Fully traced program execution including a full screen single step command that instructs Z8E to disassemble code and to move the cursor to the next instruction to execute

- Up to 16 user settable breakpoints with optional pass counts

- True symbolic debugging using the input from multiple Microsoft MACRO-80 .PRN and LINK-80 .SYM files and Z80ASM .LST and SLRNK and Z80ASM .SYM files from SLR Systems.

- Dynamic relocation of Z8E at load time to the top of user memory regardless of size. No user configuration of any kind is required.

You may want to spend some time familarizing yourself with the manual and Z8E′s command structure, especially the EXAMINE memory command, before turning to the INSTALLATION section.

This Page Intentionally Left Blank.

II. INSTALLATION

First make a working copy of Z8E, then place your original diskette in a safe place. Make all modifications to the working copy, not the original.

Z8E's (E)xamine memory command will be used to change memory contents. This command is described briefly below. For a more detailed explanation please refer to SECTION V of this manual, COMMAND INPUT.

Z8E requires an addressable cursor which can be patched symbolically as follows:

First instruct Z8E to load itself as well as the symbol file:

```
A>Z8E Z8E.COM Z8E.SYM
```

The symbol file Z8E.SYM contains the name and address of each parameter which may need to be modified.

Use the (E)xamine memory command to change the required bytes. You may enter commands in response to Z8E's asterisk prompt. Once you enter "E" followed by the symbolic name of the address you which to change, Z8E will respond by displaying the actual address followed by the hex and ASCII representation of the byte being examined (non-printable characters are shown as a "~"). For example:

```
*E  MXYCP <cr>

285E   2   ~   XX   <cr>   ;XX represents your input
285F   1B  ~   XX   <cr>
2860   3D  =   XX   <cr>
2861   00  ~   .           ;PERIOD ENDS COMMAND
*
```

IMPORTANT:
Always patch using the symbolic name of the variable; the addresses shown in the example above are for demonstration only and do not necessarily reflect the actual locations of the variables in memory.

Listed below are the symbolic names of the addresses which may have to be patched for your CRT.

MXYCP - Cursor addressing lead-in string. The first byte (the number 2 in the above example) represents the number of bytes in the string. The string may be up to 10 bytes long. This actual lead-in string should immediately follow the count byte.

Default is the two character string:

1B (Hex), 3D (Hex)

ASCII ESCAPE, followed by EQUAL SIGN.

ROWB4? - Set this byte as follows:

NOT ZERO - Row is sent before Column
ZERO - Column is sent before Row

Default is NOT ZERO, row sent before column.

ROW - Set this byte to contain the value which is to be added row number before it is sent to the screen.

Default is 20 Hex, ASCII space.

COLUMN - Set this byte to contain the value which is to be added column number before it is sent to the screen. Default is 20 Hex, ASCII space.

CASE - This byte controls whether you prefer entering symbol names in upper or lower case. It also controls whether disassembly will be done in upper or lower case. Patch as follows:

FF - lower case (DEFAULT)
00 - UPPER CASE

MAXLEN - This is the maximum length of permitted for symbol names. The permissable values are 6 and 14. If patched to any other value then Z8E will use 6. The maximum length of the symbol is required by Z8E in order to allocate space for loading the symbol table. If MAXLEN equals 6 then Z8E reserves 8 byte per symbol, 6 for the name and two for the address. If the number 14 is used then Z8E reserves 16 bytes per symbol. Hence MAXLEN impacts the amount of TPA available to the program since a symbol table of 16 bytes per entry obviously takes up twice as much space as one with 8 byte entries.

If, while reading in the symbols from disk, Z8E encounters a symbol longer than the value specified in MAXLEN the symbol name is truncated to MAXLEN.

 6 - Maximum Symbol Length (DEFAULT)
14 - Optional Symbol Length

TO SAVE THE PATCHED PROGRAM:

```
*W ANYNAME.COM  (Writes the File to Disk)
```

This completes the installation of Z8E. Typing in ^C (Control C) in response to Z8E´s asterisk prompt will return you to the operating system.

USER CODED CONSOLE I/O

The following section provides details on a method of optionally replacing the BDOS calls for Console I/O which Z8E uses with physical console I/O routines or direct BIOS calls.

To modify them use the symbol names listed below and assemble your routine at the appropriate address (via Z8E′s (A)ssemble command - See Manual).

TTYQ: This routine checks the status of the console. If a character is waiting it is read; otherwise, TTYQ returns a zero in A to indicate that no character is waiting.

TTYI: Read a character, waiting until one arrives. Return Character in A.

TTYO: Output a character, waiting until it is sent. Character passed in A.

Listed below is the code that Z8E uses to do console I/O; use it as a model. Your routines should replace the instructions with the double semicolons. Be sure to save the reqisters as show below. The size of each routine must not exceed 32 bytes.

```
TTYQ:   push    bc
        push    de
        push    hl
        ld      c,11      ;;Check Console Status
        call    BDOS      ;;BDOS returns:  A = 00  No Character
                          ;;               A = NZ  Input Waiting
                          ;;
        and     a         ;;Character Here?
        ld      c,6       ;;
        ld      e,0ffh    ;;
        call    nz,BDOS   ;;If Character Here Read It...
                          ;;   Else Fall Thru
        pop     hl
        pop     de
        pop     bc
        and     7fh
        ret

        org     TTYQ+32

TTYI:   push    bc
        push    de
        push    hl
TTYI00: ld      c,06      ;;Unadorned Console Input
        ld      e,0ffh    ;;Tell CP/M this is Input Request
        call    BDOS      ;;
        and     7fh       ;;Strip Parity
        jr      z,TTYI00  ;;Loop til Input Arrives
        pop     hl
        pop     de
        pop     bc
        ret

        org     TTYI+32


TTYO:   push    af
        push    bc
        push    de
        push    hl
        ld      c,02      ;;
        ld      e,a       ;;
        call    BDOS      ;;Console Output
        pop     hl
        pop     de
        pop     bc
        pop     af
        ret

        org     TTYO+32
```

The symbols TTYQ, TTYI, and TTYO are included in Z8E.SYM. Therefore these routines can be patched symbolically using Z8E′s assemble command, for example:

```
*A TTYQ
1F76     C5        TTYQ:    PUSH     BC
```

Z8E also contains a provision for user installed initialization code. As soon as Z8E is loaded, but before it relocates itself into high memory, it makes a call to INIT. As presently configured INIT merely contains a RET instruction. However the user may add up to 127 bytes of initialization code. This code may be used for any purpose, for example, to change your SIO or Uart from interrupt driven to non-interrupt driven in the event that the Z8E console routines were replaced. Any code installed at INIT is executed once and is not moved to high memory with the rest of Z8E. You need not save any registers.

SUPPLYING YOUR OWN CURSOR ADDRESSING ROUTINE

If your computer requires a custom cursor addressing routine it can be easily added by following the steps listed below:

1. Examine the Z8E.SYM file that to determine the address of Z8E's standard cursor addressing routine which is called XYCP. Associated with name XYCP in the file is its absolute address.

2. Using your own text editor code your routine and preface it with the following puedo-ops:

```
ASEG
.PHASE    XXXXH
```

   Where XXXX represents the absolute hexadecimal address obtained in step 1.

Z8E will pass the row address in the B REGISTER and the column address in the C REGISTER. Row numbers range from 0 to 23 while column numbers range from 0 to 79. Your job is to translate these two number into a cursor postion on the screen of your CRT.

Save all registers including BC. Use the following skeleton as a guide:

```
                    ASEG
                    .PHASE  XXXX        ;From  Z8E.SYM

          YOURS:
                    PUSH    BC
                    PUSH    DE
                    PUSH    HL

                    CURSOR ADDRESSING CODE HERE

                    POP     HL
                    POP     DE
                    POP     BC
                    RET

                    END
```

Use Z8E′s output routine TTYO as described above (or your own routine) to output the characters in your cursor addressing sequence. Obtain the absolute address of TTYO from the file Z8E.SYM. Code the call to the subroutine using the absolute address in hexadecimal. For instance, if Z8E.SYM contains the entry:

```
                    2FE2   TTYO
```

then code your call statements as:

```
                    CALL   2FE2H
```

Z8E imposes only one restriction on the code you write. In order to guarantee that your routine can be relocated into high memory by Z8E do not load any 16 bit constants into register pairs; instead do two 8 bit loads. For example, do not use the following statement:

```
                    LD     HL,1234H
```

Rather, code it like this:

```
                    LD     H,12H
                    LD     L,34H
```

This is the only restriction other than the maximum code length which is placed on your code which is 128 bytes.

3. Assemble your routine with either Macro-80 or Z80ASM. Link it with either Link-80 or SLRNK.

4. Load Z8E.COM using Z8E:

```
A>Z8E Z8E.COM Z8E.SYM

*
```

5. Now overlay Z8E's cursor address code with your own:

```
*I YOURCODE.COM,XYCP
```

   Z8E will load your cursor addressing routine on top its own beginning at the address associated with the symbol XYCP.

6. Save the new file using a name of your choosing:

```
*W  NEWDEBUG.COM
```

7. Exit back to the operating system be entering a Control-C at the asterisk prompt.

This Page Intentionally Left Blank.

III. INVOKING Z8E AT THE CP/M COMMAND LEVEL

Upon invokation at the CP/M command level Z8E loads at the low end of the Transient Program Area (TPA) which begins at absolute address 100H. The TPA is the area in memory where user programs are executed.

Once loaded Z8E determines the size of the TPA by examining the address field of the jump instruction at location 5. This address represents both the entry point into CP/M and the end of the TPA. Z8E lowers this address by approximately 9K bytes and relocates into this area by adjusting all addresses within itself to reflect its new location. The jump instruction at location 5 is similiarly modified to reflect the new size of the TPA. Thus all programs which use this address to determine the amount of available memory can run unchanged. Z8E completes its initialization by storing a jump instruction to its breakpoint handling software at absolute address 38 (hexadecimal).

Symbols which are loaded from files are stored by Z8E in a symbol table at the top of the TPA just below Z8E. Z8E will dynamically allocate the storage necessary to hold all symbols loaded from files; however, Z8E also allows the user to enter his own symbols from the keyboard via the (A)ssemble command. Z8E does NOT reserve ANY space in memory for user generated symbols. The user must explicitly request memory space on the CP/M command line. This is accomplished by entering the number of symbols for which space should be reserved as a decimal number. This number must be enclosed in parentheses and must appear as the first argument on the command line as shown below:

```
A>Z8E (32)
```

In this example the user has requested space for 32 user defined symbols. If MAXLEN has be set to 6 (See INSTALLATION Section) then each symbol requires 8 bytes of storage, hence, in this example Z8E will set aside 256 bytes of memory for user defined symbols.

Subsequent action is based on the format of the remainder of the command line as entered by the user. In the examples that follow bear in mind that any of these command lines may contain the argument requesting memory space for user symbol table entries. The argument would appear immmediate after "Z8E" in every case.

1. A>Z8E

   Z8E resides as a standalone program in memory.

2. A>Z8E USERFILE.COM

   USERFILE.COM is loaded at the beginning of the

TPA and is ready to be acted on by Z8E commands.

3. A>Z8E USERFILE.COM USERFILE.SYM [,bias]

USERFILE.SYM is read in by Z8E and all symbol names contained in the file are entered into a table which begins at the starting address of Z8E (the ending address of the "new" TPA) and extends downward in memory. The optional bias, if specified, is a 16 bit value which will be added to the 16 bit address associated with each symbol in the file. (In this example a .SYM file is shown; however, since all addresses appearing in a .SYM file are absolute the optional bias would probably not be used.)

USERFILE.COM is loaded at the start of the TPA only after the .SYM file has been read and the symbol table built.

4. A>Z8E USERFILE.COM USERFILE.PRN [,bias]

As in the previous example USERFILE.COM is loaded at the beginning of the TPA, but in this instance a .PRN file is used to construct the symbol table. The optional bias becomes very useful if the .LST or .PRN file represents the listing of a relocatable program. Relocatable programs linked using Microsoft's LINK-80 default to a load address of 103H with the three bytes of memory located at 100H containing a jump to the entry point of the program. Therefore, if the user supplies a bias of 103 in the command line all relocatable symbols in the file will be associated with their actual addresses in memory. Any bias specified will only be added to those symbols which are flagged as code relative in the .PRN file. A bias will not be added to any symbol flagged as ABSOLUTE, EXTERANL, OR COMMON.

USERFILE.COM is loaded at the start of the TPA only after the .LST or .PRN file has been read and the symbol table built.

5. A>Z8E USERFILE.COM USERFILE.SYM [,bias] NFILE.LST [,bias]

The true power of Z8E′s symbol loading is best evidenced when loading multiple symbol tables from several files. The first file is generally a .SYM file specifying all the global symbol names in the program to be tested. The subsequent files specified on the command line are usually .PRN or .LST files of the individual source modules that were originally assembled and then linked (which produced the .SYM file). Although only two files (USERFILE and NFILE) are shown in this example, in actuality the number of .SYM and .PRN files specified in the command line is limited only by the size of Z8E′s input buffer which is 80 characters long.

USERFILE.COM is loaded at the start of the TPA only after all .SYM and .PRN/.LST files have been read and the symbol table built.

Note:
If no bias is specified, Z8E will use a bias of zero.

If more than one .LST or .PRN file is being loaded, then each file name can be specified with its own bias. The bias may be entered in the form of a symbol name, hexadecimal number, decimal number, or any combinatiion of the three in an expression using the + and - operators. If the individual module has a global entry point, the name of which was previously loaded, the user can bias all symbols with the value associated with this name. In this way all symbols, both absolute and relocatable, are associated with their actual location in memory.

Z8E as presently configured can build a symbol table from the list files produced by the following programs:

1.	Microsoft	MACRO-80	V3.37	.PRN Files	May 8, 1980
2.	Microsoft	MACRO-80	V3.44	.PRN Files	Dec 9, 1981
3.	Microsoft	LINK-80	V3.44	.SYM Files	Dec 9, 1981
4.	SLR Systems	Z80ASM	V1.07	.LST Files	
5.	SLR Systems	SLRNK	V1.07	.SYM Files	

Z80ASM and SLRNK may be configured for 80 or 132 column output.

Z8E uses the file name extension (the three characters appearing to the right of the period) to determine the format of the file. Each of the above file types has a distinguishing format. The characteritics of each type are described in APPENDIX A.

During the loading process Z8E displays status and error messages on the console relating to the activity in progress as shown below:

	STATUS MESSAGE	DESCRIPTION
1.	Loading: USERFILE.COM	Z8E is attempting to open the named file (in this case, USERFILE.COM)
2.	Number of symbols loaded:	Following the loading of all symbols from a listing file or a .SYM file, the number of symbols loaded from the specified file is displayed as a decimal number.
3.	Loaded: 100 YYYY Pages: ZZZ	Z8E displays the starting and ending memory addresses of the target file (the first file specified on the CP/M command line and the one which is going to be debugged). "Pages:" refers to the decimal number of pages and is the count of 256 byte pages in the file. This number may be subsequently used with the CP/M SAVE command once the debug session ends.

	ERROR MESSAGE	DESCRIPTION
1.	File not found	The file specified in the command could not be found on the specified drive.
2.	Symbol table not found	The specified file was found but did not contain a properly formatted symbol table.
3.	Invalid offset - using 0000	The user has specified an invalid offset to be added to each loaded symbol. Z8E will continue to load this symbol file but will not add any bias to the sym-

bols. This error may have occured because the user specified an offset in the form of a symbol which had not been previously loaded, or the user may have specified a numeric value which contained an illegal character.

4. Syntax Error — The file name was incorrectly specified.

After all user files, both symbol files and the .COM file to be debugged, have been loaded Z8E displays current memory usage as follows:

```
Total Symbols:    XXXX
Symbol Table:     XXXX - XXXX
Z8E relocated:    XXXX - XXXX
Top of memory:    XXXX
```

It is important to note that Z8E expects the files appearing in the command line to be appear in a specific order. The first file name appearing in the command line is assumed to be the target file which is to be debugged. It is always the last file to be loaded. All file names following the target file name are assumed to be symbol input files and they are loaded in the order in which they appear.

The first file named in the command line is always loaded starting at address 100 hex. The "I" command contains an option to allow the file to be loaded at a different address. This feature is not available at the CP/M command line level.

For a discussion of the format of symbol files see APPENDIX A.

This Page Intentionally Left Blank.

IV. INITIALIZATION

Once Z8E has been loaded, and has in turn loaded all files specified on the command line, it initializes all user registers to 0 with the following exceptions:

> The user′s program counter contains address 100 hex which is the start of the TPA.
>
> The user′s stack pointer is set to the starting address of Z8E (the top of the TPA) minus two. These two bytes are set to zero in accordance with CP/M convention. When CP/M loads a program it initializes a stack for the loaded program by pushing the address of the jump to the system warm boot routine onto it. Thus user programs (STAT.COM is an example) can choose to terminate themselves and return to CP/M by executing an RET through this address on the stack. Z8E accomplishes the same objective: the 0000 on the stack permits the user program to return to CP/M via address 0000 which always contains a jump to the system′s warm boot routine.
>
> The user I (interrupt) register is set to the value contained in the I register when Z8E was loaded. Modify at your own risk.

All input and output by Z8E is accomplished using buffers contained within itself. Z8E does not use the default DMA buffer at absolute location 80 nor does it use the default File Control Block (FCB) at absolute location 5C.

> Note:
>
> When CP/M finishes loading any program, including Z8E, it moves the command line tail to the default DMA buffer at absolute address 80 (hex) and initializes the default FCB at absolute address 5C to the name of the first file (or first two files if two or more are specified) appearing in the command line. Z8E makes use of this information in order to load the user program and any symbol files. If the program to be tested also expects an initialized FCB and/or DMA buffer (as is very often the case), then the user must effect this before attempting to execute the program.
>
> For example, many text editing programs are invoked by typing the name of the editor program followed by the name of the

program to edit on the CP/M command line, as in hypothetical case:

```
A>EDIT B:FYL2EDIT.BAS
```

Once the program EDIT.COM is loaded it may expect to find the default FCB to be already set up to read the file FYL2EDIT.BAS. EDIT.COM may also expect the DMA buffer to contain the number of characters in the command line at address 80, as well the the text of the command line starting at address 81. In this example location 80 would contain a hexadecimal F (decimal 15) representing the number of characters, and locations 81 through 8F would contain the 15 characters (space through S). Similiarly, the first byte of the default FCB at address 5C would contain the number 1 (numeric equivalent of drive B) and the next 11 bytes would contain the file name FYL2EDIT in ASCII. If the name FYL2EDIT was shorter than 8 characters, then the remainder of the file name field in the FCB would be filled with ASCII spaces. The next 3 bytes would contain the file type in ASCII; in this example the file type is BAS. If no file type was specified, this field would contain 3 ASCII spaces.

Now, if the user was to debug the EDIT program using Z8E, this initialization of the default DMA buffer and default FCB must be accomplished "by hand" prior to attempting to debug EDIT.COM, owing to the fact that CP/M has already set up these to areas with the data from the command line which was typed in to load Z8E. In short, EDIT must be tricked into believing it was loaded by CP/M and not by Z8E and the user must perform the initialization of these two areas. The user may use the E command (to store both ASCII and numeric data in memory) to simulate an initialized command line buffer and FCB. Further information regarding the format of the FCB and DMA buffer may me found in Digital Research's CP/M 2.0 INTERFACE GUIDE.

DEBUGGING HINT:

It is not necessary to initialize the default FCB and/or the default (command line) DMA buffer every time a program to be tested is loaded (if indeed this program utilizes them). Instead follow the procedure listed below (If you haven′t read the individual command summaries the following may make more sense later):

> Once you have loaded the program to test perform the required initialization of the FCB′s at 5CH and 6CH and the command line buffer at 80H using the E command. Use the ASCII string option with the E command to set the text portions. Use the numeric input function to intialize the drive specification at address 5C and the character count at 80H.
>
> Use the W command to write out memory starting at address ZERO. As in:
>
> ```
> *W  NEWFILE.COM  0   XXXX
> ```
>
> Where XXXX is the highest address you wish to save. Now the next time you load this file it will of course load at address 100H. Use the M (move memory command) to move it to location 0000. Your FCB and DMA buffer are initialized.
>
> ```
> *M  100  XXXX+100  0
> ```

This Page Intentionally Left Blank.

V. COMMAND INPUT

Once file and symbol table loading has been completed, Z8E prompts the operator for command input by displaying the "*" character. The operator can then type any of Z8E's single letter commands. Some commands require no arguments while others require between one and four. Arguments may be in any of the forms listed below (except as noted in the description of the individual commands):

SYMBOL: Any symbol previously loaded or previously entered via the keyboard (see A command) may appear as a command argument. All symbols are treated as 16 bit values.

HEX: A 16 bit hex number may be entered as an argument. Only the last four characters entered are treated as significant input if Z8E is expecting a 16 bit argument. In those instances where Z8E expects a 8 bit argument, only the last two characters are significant. As such, the user may elect to correct mistakes by either backspacing and retyping, or by continuing to enter the number and ensuring that the erroneous digit does not appear in the rightmost four (or two) characters as shown in the following example:

```
*E 1E21F4
```

If a 16 bit argument is expected Z8E would ignore the first two digits (1 and E) and would examine the contents of memory location 21F4.

If no symbol table is present in memory then hexadecimal numbers (8 or 16 bits in length) may begin with any digit 0 - F. However, if a symbol table is in memory then all hexadecimal numbers which begin with a digit in the range A - F are evaluated first as symbol names. If no corresponding name is found in the symbol table then Z8E attempts to reevaluate the name as a hexadecimal number. For example, the token DEAD is a valid symbol names as well as a valid hexadecimal number. If a symbol table is present then Z8E first searches the symbol table looking for the string DEAD. If no match occurs then Z8E treats DEAD as the hexadecimal number ODEAD. To force Z8E to evaluate an argument as a hexadecimal number prefix the argument with a leading zero as in ODEAD.

REGISTER:

Valid Z80 16 bit register names are permitted as arguments. If a 16 bit register name is entered, Z8E uses the 16 bit value currently contained in the specified register pair in the user's register set as an argument.

```
*D HL 8
```

instructs Z8E to dump the first eight of memory bytes which are located at the address contained in the user's HL register pair

Valid 16 bit register names:

```
AF - Accumulator and Flag
BC - BC register pair
DE - DE register pair
HL - HL register pair
SP - Stack Pointer
P  - Program Counter
PC - Program Counter
IX - IX index register
IY - IY index register
```

Note that the program counter may be specified in either of two ways. The single character "P" can be used to specify the program counter provided it does not appear in an expression. To include the current value of the user's program counter in an expression the mnemonic "PC" must be used.

If an expression used as an argument contains a register pair as one of its terms, the register pair must be the first term. Also, only one register pair may be included in an expression:

HL+4	valid expression
5+DE	invalid expression - register pair is not the first term
HL+BC	invalid expression - more than one register pair was

specified

P-3 invalid expression - "PC" must be used to include the current value of the program counter in an expression

To differentiate between the hexadecimal numbers AF, BC, and DE and the Z80 register pairs of the same name be sure to prefix the numerical version with a leading 0.

Note also that the Z80 prime register names are not allowed as arguments except in the R command.

REGISTER INDIRECT:

Z8E allows the user to specify the data contained in the memory location pointed to by a register pair as an argument. For instance, if the user's HL register pair contained 18EE and the addresses 18EE and 18EF contained the bytes 42 and 61 respectively, then the command *E (HL) would examine the contents of memory location 6142. Note that register indirect memory references are indicated by enclosing the register pair name in PARENTHESES which follows the ZILOG mnemonic method of signifying "the contents of".

The most useful application of register indirect arguments is to set breakpoints at subroutine return addresses. Consider the situation of a program which is currently suspended via a breakpoint somewhere in the middle of a subroutine. The user is no longer interested debugging the body of the subroutine; he only cares about getting back to the instruction that follows the CALL that got him into the subroutine. Register indirect format allows him to enter:

```
*B (SP)
```

This informs Z8E to set a breakpoint at the address pointed to by the stack pointer register.

DECIMAL:

Decimal numbers in the range 0 - 65535 may be entered as arguments. All digits of the number must be in the range 0-9. A decimal number must be followed by a "#" character, otherwise Z8E will treat it as a hex number. The following example shows a decimal number being input as part of the E command:

```
*E 512#
```

instructs Z8E to examine memory location 512 decimal (200 hex)

LITERAL:

ASCII literals up to 78 bytes in length are permitted as arguments (Z8E's input buffer is 80 characters long less the opening and trailing quote characters). ASCII literals must be enclosed in quotes. The quote character itself is the only character not permitted as a literal. Commands which do not permit the use of ARGUMENT-STRINGs (see below) will still accept input in the form of quoted strings. In such a case Z8E will ignore all but the last two characters of the quoted literal, treating the input as a 16 bit number. For example if the user entered:

```
*Z 'ABCD'
```

Z8E would begin treat 'BC' as a 16 bit number and begin disassembling at address at 4243.

ARGUMENT-STRINGS:

The F (find), E (examine memory), N (query I/O ports without pre-read), Q (query I/O ports), and Y (fill memory) commands permit the use of ARGUMENT-STRINGS, which are simply combinations of all valid argument types separated by commas. ARGUMENT-STRINGs may be any length up to the limit of Z8E's input buffer which is 80 bytes long. ARGUMENT-STRINGs may be terminated by either a carriage return or the first space character not appearing in between quote characters. The following is an example of a 15 byte ARGUMENT-STRINGS string which combines SYMBOLS, LITERALS, HEX, and DECIMAL numbers:

```
SYMBOL,'xyZ',4F,12E4,9,21#,511#,'ABc'
```

Assuming that SYMBOL is equal to 177H then the above ARGUMENT-STRING would evaluate to:

```
01 77 78 79 5A 4F 12 E4 09 15 01 FF 41 42 63
```

Again, ARGUMENT-STRINGS are terminated by either a carriage return or by the first space character that does not appear in a quoted literal string.

Z8E permits expressions using the + and - operators. Any argument type may be combined with any other type. The length of an expression is limited only by the size of the input buffer. Expressions are evaluated from left to right and the use of parentheses is not permitted.

Z8E indicates argument errors by printing a question mark.

Arguments may be line-edited using the standard CP/M control characters:

backspace:	erase the last character typed
control X:	erase the entire line
control C:	return to CP/M via warm boot

All input is truncated to the size of Z8E′s input buffer which is 80 characters long.

All alphabetic input to Z8E may be in uppercase or lowercase. All output by Z8E follows the dictates of the CASE byte as patched by the user (see INSTALLATION).

In this manual the appearance of square brackets [ ] around an argument always indicates that the argument is optional.

This Page Intentionally Left Blank.

VI. BREAKPOINTS

Breakpoints are those addresses in the program under test at which the user wishes to suspend execution and return control to Z8E. The user may set, clear, and display breakpoints at any time, via the appropriate command in response to Z8E′s asterisk prompt. Z8E′s implementation of breakpoints does not force the user to tediously enter breakpoint addresses every time execution is resumed. Rather, the user may enter up to 16 breakpoint addresses and each breakpoint, once set, is stored in one of Z8E′s internal tables and remains in effect until explicitly cleared by the user via the Clear breakpoint command (see C command).

Z8E also allows you to specify a pass count to be associated with any breakpoint that is set. Pass counts indicate the number of times a particular instruction must be executed before Z8E will regain control.

Furthermore, Z8E does not modify any code in the user program until a GO command is issued (see G command). This permits the user to examine code, and make patches if desired, at any point in the debug session.

When a breakpoint is reached in the user program and Z8E regains control, the message: *BP*XXXX is displayed where XXXX represents the hexadecimal address of the breakpoint. In addition, Z8E will display the symbolic name of this address if one exists in the symbol table. Z8E follows this with a display of the asterisk prompt indicating it is ready ready for command processing.

The message: *ERROR*BP*XXXX is displayed on the console whenever Z8E determines that control has been regained without a valid breakpoint having been reached. This is generally caused by a user program which has gone off the deep end. If the user examines the current contents of the registers (via the X command) the current program counter will most assuredly contain an address which had not previously been set as a breakpoint. Things to look for when this situation arises include: a program that blew its stack, a program that performed a 2 1/2 gainer with a full twist indirect through a register; ie. JP (HL) into the great unknown, and attempting to trace where wise men fear to tread (BIOS and BDOS I/O routines).

Z8E will allow you to single step (trace) and set breakpoints anywhere in memory. However, bear in mind that as you enter the BIOS and BDOS netherworld your stack pointer will at some point be saved directly in memory as CP/M switches to its own stack (your stack pointer is not saved on a stack by CP/M). If a breakpoint has been set at an instruction somewhere in BDOS or in the BIOS (after this save of your stack pointer has occured) and this breakpoint is reached, Z8E will itself call a BDOS routine in an attempt to display the *BP*XXXX message on the console. At this point CP/M will save Z8E′s stack pointer and overlay yours in memory. When BDOS eventually restores the stack pointer and executes a RET instruction you will not return to your program and your stack pointer will be gone. These routines can be traced, albeit with difficulty, but you must keep an eye on what CP/M is doing with the stack pointer.

This Page Intentionally Left Blank.

A Assemble

The A command permits the user to effect inline assembly of Z80 assembler source code, including labels and symbols, using the full Z80 instruction set. In addition, the assembler accepts standard Zilog mnemonics (APPENDIX B), expressions using the + and - operators, as well as the following five assembler directives: ORG, DEFB, DDB, EQU, and DEFW. The format of the command is:

```
*A  ARG1  <cr>
```

where ARG1 represents the starting address at which assembly will take place

ARG1 may be of any type

Z8E initially prompts the user by first disassembling and displaying the instruction currently located at the address specified by ARG1. This is done as a convenience to permit the user to ensure that any patches will be assembled into memory at the intended location. Z8E then outputs a carriage return/line feed, displays the address specified as ARG1, and awaits input. Z8E will not disassemble before every line of source code entered by the user, only before the first one.

Z8E expects assembler input in the following format:

```
LABEL:  opcode  [operand1] [,operand2]
```

The label field is always optional, the opcode field is mandatory only if no label was entered, and the operand field must naturally be included for those Z80 instructions which require one. The three fields may be separated from one another by spaces or tab characters.

Z8E does not automatically reserve space within itself for user supplied symbol names. User supplied symbols, as opposed to those loaded from files, are entered from the keyboard in the label field using the (A)ssemble command. Symbol table space to hold user supplied symbol names must be explicitly requested on the CP/M command line as explained in the section "INVOKING Z8E at on the CP/M COMMAND LEVEL". These user supplied symbols, once entered, may be referenced in the operand field of any subsequent assembly statement or in the argument field of any Z8E command. These symbols come in handy when disassembling .COM files for which no source listing exists and also when patching code.

The assembler is a one pass assembler and forward references to symbols which do not already appear in the symbol table are flagged as errors. However, Z8E allows the use of the ORG directive (see discussion below) which allows the user to manipulate the assembler′s location counter, which helps to minimize the no forward reference limitation.

Labels may begin in any column, but all labels must be followed by a colon even those appearing in an EQU statement. Labels may be of any length but only the first 6 characters are significant. Z8E always assigns the 16 bit value of the assembler′s current location counter to the label being entered, unless the statement is an EQU directive. Labels need not be followed by an opcode and this (as well as the EQU directive) provides a convenient way to assign a value to a symbol name. Merely set the assemblers location counter (via the ORG directive or as ARG1 in the command line) to the value you wish to assign, then type the symbol name followed by a carriage return. No object code is produced and no user memory areas are modified but the symbol and its associated value are entered into the user symbol table. Z8E does not treat duplicate symbol names as errors. Rather, if the user enters a symbol name which already appears in the symbol table, the value associated with the new symbol replaces the one associated with the old. For example, if the symbol ENTRYP exists in the symbol table and is associated with the value 23DA and the user assembles the following instruction:

```
     41FF 0E 04       ENTRYP: LD C,4
```

Z8E would replace 23DA with 41FF.

Assembler statements which do not contain labels may begin in any column, including column one. There is NO need to insert a leading space or tab before an opcode if the opcode is not preceded by a label.

Operands appearing in the operand field of the instruction to be assembled may be any of the following types subject only to the proviso that 16 bit values cannot appear as operand for those Z80 instructions which require 8 bit values. Expressions combining any of the following four types (with the + and - operators) are also permissable.

```
               SYMBOL  (from symbol table)
               HEX
               LITERAL (two bytes maximum)
               DECIMAL
```

In addition the dollar sign ($) may also appear in both the operand field of any instruction in which a 16 bit operand is allowed, and also in the operand field of any relative jump instruction. The dollar sign represents the current value of the assembler′s location counter, that is, the address appearing on the line at which the assembly is taking place.

The operand field of a relative jump instruction can be entered in either of two ways. The user may code the operand using the dollar sign mentioned above as in the following examples:

```
JR   NZ,$+11        ;jump to address PC+11 (hex)

DJNZ  $-24#         ;jump to address PC-24 (decimal)
```

The user may alternatively specify a 16 bit value in the operand field of a relative jump instruction and let Z8E calculate the relative displacement from the assembler's program counter as shown below:

`JR    C,LABEL`	Assuming LABEL exists, in the symbol table Z8E will calculate the offset. LABEL must be within +129 or -126 bytes from the assembler's location counter or an assembly error will result.
`JR      NZ,1080`	Z8E calculates the displacement between the assembler's current location counter and the address 1080 (hex).

Z8E indicates error-free input by first displaying the resultant object code and then displaying (on the next line) the next address at which assembly will take place.

Assembly errors are always indicated by a double pair of question marks which appear following the location counter. An error flag is also printed and will be one of the following:

ERROR FLAG	MEANING
L	Label starts with numeric character
O	Invalid opcode
S	Syntax error
T	Symbol table full
U	Instruction references an undefined symbol name
V	Value error - a 16 bit value was specified as an operand for an instruction which permits only 8 bit numbers.

If an error occurs, Z8E will reprompt the user with the same location counter address.

As was mentioned previously the Z8E assmebler uses standard Zilog mnemonics. The one exception to this is the EX AF,AF′ instruction. To assemble this instruction the trailing quote character must be omitted.

Z8E supports the ORG directive which allows the user to change the value of the assembly location counter. The operand field of the ORG directive may be a 16 bit argument of any type. After setting the new assembly location counter Z8E displays the disassembled instruction at the new address.

Z8E supports the DEFB, DEFW, and DDB directives which give the user the ability to assemble data constants into memory. DEFB accepts an 8 bit operand; the value of which in placed into memory at the address of the assembler′s current location counter. DEFW allows the user to specify a 16 bit operand value, the low order byte of which is placed into memory at the address of the assembler′s current location counter, while the high order byte of the operand is placed into memory at the address of the assembler′s current location counter plus one. This is in accordance with the 8080/Z80 convention of storing the high order byte of 16 bit data toward the high end of memory. The DDB (define double byte) directive allows the user to specify a 16 bit value which, in contrast to the DEFW directive, is stored in memory with the high order byte toward the low end of memory. That is, a DDB directive instructs Z8E to store the most significant byte of the 16 bit operand value in memory at the address of the assembler′s current location counter, and the least significant (low order) byte is placed into memory at the address of the assembler′s current location counter plus one.

The EQU directive allows the user to assign a value to a symbol. An EQU directive does not generate object code. It merely allows the user to reference a numeric value by a symbolic name in subsequent assembly statements or monitor commands. It is especially useful when used prior to disassembling (see Z command) code for which no symbol table exists. The EQU directive requires the user to supply a symbolic name in the label field of the instruction. If Z8E indicates errors in an EQU statement by printing question marks. If an EQU statement is correctly assembled by Z8E, the address of the assembler′s current location counter is erased since an EQU statement generates no object code. Operands appearing in EQU statements are evaluated to a 16 bit value. Z8E will display the value of this 16 bit number as four hex digits in the object code field on the console.

B Set Breakpoint

Breakpoints are those addresses at which the user wishes to suspend execution of the program under test. Breakpoints may be set at any time in response to Z8E's asterisk prompt. Z8E allows the user to set up to 16 individual breakpoints in his program. Z8E also allows the user to specify a pass count to be associated with any breakpoint.

The command is invoked as follows:

```
*B ARG1[,pass count] [ARG2... ARGn] <cr>
```

where each argument represents the address in the user program at which a breakpoint is to be set

Normally, that is when no pass count is specifed, execution of the user program stops and control returns to the Z8E command level as soon as a breakpoint is reached. Pass counts are used to inform Z8E that execution of the user program should halt only when the specified breakpoint is reached the number of times times indicated by the pass count.

Pass counts are specified by following the breakpoint address with a comma and then entering a pass count immediately following the comma.

An existing pass count may be changed to a different value by re-entering the same breakpoint address, following it with a comma, and then specifying the new pass count.

To break on a multi-byte Z80 instruction the address specified as the breakpoint address must be that of the first byte of the instruction. Users who fail to observe this rule will generally find their programs hopping the next bus to never-never land. If a patch is made at an address of a breakpoint currently in effect be sure the breakpoint address is still pointing at the first byte of the new instruction.

Multiple breakpoints may be set with the same B command by separating each one with a single space. If multiple breakpoints are specified and Z8E detects an erroneous argument (a non-existent symbol for example) a question mark will be printed, and the command terminates. All valid breakpoints specified up to the invalid one will be set.

Z8E displays a question mark when a attempt is made to set a seventeenth breakpoint.

C Clear Breakpoint

The C command clears individual breakpoints previously set by a B command. The format of the command is:

```
*C  ARG1 [ARG2...ARGn] <cr>
```

where each arg may be any valid argument type which evaluates to an address previously set as a breakpoint

Multiple breakpoints may be cleared by the same C command by separating each argument with a single space.

Z8E displays a question mark when an attempt is made to clear a non-existent breakpoint.

To clear ALL breakpoints enter: *C * where the asterisk indicates ALL.

D Dump

The D command allows the user to dump memory in both hexadecimal and ASCII to the console in user specified block sizes.

The format of the command is:

```
*D  [ARG1]  [ARG2]  <cr>
```

where ARG1 = the starting address to dump

ARG2 = dictates the dump format depending on its value. If ARG2 is in the range 0 - 255 then it is treated as a block size and represents the number of bytes to be displayed (0 is treated as 256). If ARG2 is greater than 255 then ARG2 is treated as an ending address and memory will be dumped non-interactively to the console.

ARG1 and ARG2 may be of any argument type.

If ARG1 is omitted then the dump resumes from the last memory address +1 as displayed via the previous invocation of the D command. If no previous D command had been given then memory is dumped starting at address 100H.

If ARG2 is omitted then the most recent value of ARG2 (from the last D command) is used.

The dump command displays the contents of memory in hexadecimal on the left side of the console while the ASCII equivalent of each byte is shown on the right side.

During a block by block dump (ARG2 < 256 signifies a block by block dump) Z8E waits for user input after each block is displayed. A carriage return entered by the user causes the next sequential block to be dumped while any other character causes the command to terminate.

For non-interactive dumps, starting address to ending address, pressing any key terminates the dump.

The dump command provides an especially easy way of examining tabular data, for example in scanning the disk parameter headers in your BIOS. That is, by specifying the base address as ARG1 and the table size as ARG2 the user can walk through memory, table by table.

E Examine Memory

The E command allows the user to examine and optionally modify the contents of memory. The format of the command is:

```
*E   ARG1  <cr>
```

where ARG1 is the address of the first byte to examine

ARG1 may be any symbol type

Upon receipt of ARG1 Z8E will read the contents of the specified memory address and display the byte in both hex and ASCII. At this point the user has two options. The user may specify replacement data to be written to memory starting at the current address, or he may choose to continue to passively examine memory. The choice is determined by the character(s) which are input after the contents of an address are displayed.

If the user wishes to modify memory starting at the current memory address, then an ARGUMENT-STRING may be entered following the displayed byte. Z8E will evaluate the entire string and write the evaluated equivalent of the string into consecutive memory locations starting with the current memory address. For example the user could enter the following ARGUMENT-STRING:

```
*E 45F9
45F9  42  B  'This is a string',0D,0A,13,4F,9,'More Text',05
             ^                                            ^
```

The user input apprears between the arrows and would be evaluated to the following 31 bytes:

```
54 68 69 73 20 49 73 20 61 20 73 74 72 69 6E 67
0D 0A 13 4F 09 4D 6F 62 65 20 74 65 78 74 05
```

These 31 bytes would be stored into memory locations 45F9 to 4617 and the next address displayed on the screen would be 4618.

```
4618  23  #
```

Remember that ARGUMENT-STRINGS may be terminated by either a carriage return or by the first space character which does not appear in a quoted literal string. The choice of terminator determines the which address will be displayed next. If a carriage return is used to terminate the ARGUMENT-STRING, then Z8E will display the next sequential memory address. For example:

```
*E 1002
1002  45  E  12,8F,00 <cr>
1005  28  (
```

The user entered an ARGUMENT-STRING 12,8F,00 which was evaluated to 3 bytes. Since the ARGUMENT-STRING was terminated by a carriage return the next address displayed was 1002+3 or 1005.

By terminating the ARGUMENT-STRING with a space the user can verify the contents of memory just modified. ARGUMENT-STRINGS terminated by a space cause Z8E to redisplay the starting address; this makes the data just entered availalbe for re-inspection:

```
*E 1002
1002  45  E  12,8F,00  <space>
1002  12  ~
```

If the user does not want to write any data to the current memory address, then the character entered should be a space character, up arrow (carret) character, or a carriage return.

CHARACTER	ACTION
space	read next sequential memory address
up arrow	read previous memory address
<cr>	read next sequential memory address command
period	terminate command

The user may also change the current memory address by entering an equal sign "=" followed by a valid argument. The address obtained by evaluating this argument becomes the new current memory address as shown below:

```
*E 1344
1344  89  ~  <cr>
1345  6F  o  <cr>
1346  52  R  =9F34 <cr>
9F34  63  c
```

F Find

The find command allows the user to search memory for multi-byte strings in memory. The format of the command is:

```
*F   ARG1  ARG2 <cr>
```

where ARG1 = the starting address at which to begin the search, it may be of any type

ARG2 = is an ARGUMENT-STRING representing the pattern to search for; the user may specify any combination of arguments separated by commas or spaces up to the limit of Z8E′s command line buffer which is 80 bytes long. The actual number of bytes searched for depends on how the string is ultimately evaluated.

Z8E will display every address which contains data matching ARG2. The search continues until the end of memory is reached.

The user may elect to cancel the search at any time by depressing any key on the keyboard.

If ARG2 is a single argument (as opposed to an argument string) and if this argument is a symbol name then Z8E will reverse the order of the two bytes comprising the 16 bit operand. Most 16 bit values in Z80 programs are stored with the least significant byte at a given address and the most significant byte at the given address+1 (toward the high end of memory). This is in accordance with the Z80 convention of storing the most significant byte of a 16 bit argument toward the high end of memory.

The following are examples of the FIND command:

```
*F 0 SYMBOL
```

Assuming that the symbol "SYMBOL" is associated with the hex value 3BF then Z8E would attempt to find all address containing the byte pair BF and 03 in that order, with the search beginning at address 0000. Note that the order of the two bytes is reversed because the symbol "SYMBOL" exists in the symbol table. To search for the byte pair 03 and BF in that order the user should enter the argument as either a 16 bit hex number (3BF) or as two 8 bit hex numbers (03,BF).

```
*F 100 87,32#,′ABCD′,0C3,symbol,′p′,271F
```

Assuming that the symbol "symbol" is associated with the hex value 3BF then Z8E would attempt to find all starting addresses of the following 12 byte string:

```
87 20 41 42 43 44 C3 03 BF 70 27 1F
```

Notice that Z8E would search for the two byte pattern 03 BF as the value for "symbol". If the user happened to be trying to find the instruction JP symbol the search would fail because as mentioned above 16 bit values are stored low order byte first. The user should have entered C3 BF 03.

The two bytes which represent the address of symbol are not reversed as in the example above because ARG2 is specified as an ARGUMENT-STRING as opposed to a single argument.

Z8E would begin its search at address 100 (ARG1).

G Go

The G command instructs Z8E to begin or resume execution of the user program. The format of the command is:

```
        *G   ARG1  <cr>

             where      ARG1    =    the address of the first
                                     instruction the user wishes
                                     to execute.

             ARG1 may be any argument type
```

Upon receipt of this command Z8E initializes all breakpoints in the user program, restores all user registers, and transfers control to the user program under test at the address specified in ARG1. Execution within the user program will continue until the user program reaches a breakpoint, at which point control will return to Z8E. This is the only way the user is able to return control to Z8E once the GO command is issued.

Z8E breakpoint technique has been designed such that Z8E will not directly initialize a breakpoint at the address specified in ARG1. In actualiity it would be impossible to do so since an attempt would be made to resume execution at this address, a breakpoint would have been set at this address, and control would immediately return to the monitor without this instruction ever having been executed. This limitation has been overcome in Z8E by actually copying the single instruction located at ARG1 to Z8E′s memory, THEN setting the breakpoint at the ARG1 address, and finally executing the "moved" version of the instruction in Z8E′s memory rather than in the user program. Z8E compensates for the that CALL and RELATIVE JUMP instructions are affected by the address at which they are executed. This entire process is totally transparent and it allows the user to debug loops by setting only a single breakpoint within the range of a loop, obviates the need to clear any breakpoints which are located at the address where execution is to resume, and even allows breakpoints at a DJNZ $ instructions!

HINT:
When proceeding from a breakpoint it is simplest to use the form of the GO command: *G P <cr> which informs Z8E to resume execution at the address specified by the user′s current program counter.

H Display Symbol Table

The H command allows the user to view the symbol table on the console. The format of the command is:

```
*H   [ARG1]  <cr>

          where ARG1 must be a symbol name
```

If ARG1 is omitted Z8E will display the entire symbol table starting with the first symbol in the table. If ARG1 is present Z8E will begin the display with that symbol. Z8E displays a block of 32 symbols then waits for user input. If the user enters a carriage return the the next block of 32 symbols is displayed. If the user enteres any other character the command terminates.

If a symbol name entered as ARG1 cannot be found in the symbol table Z8E prints a question mark.

I Input file

The I command allows the user to load files into the TPA after the debug session has started. The format of the command is:

```
*I   ARG1[,ARG2]  <cr>
```

ARG1 is a single unambiguous file name conforming to standard CP/M syntax rules:

- optional drive name followed by a colon
- mandatory primary file name
- optional secondary file name preceded by a period

ARG2 is an optional load address. If ARG2 is not specified the named file is loaded at the start of the TPA (address 100 hex). If ARG2 is given the file will be loaded at this address. Z8E will NOT relocate individual addresses within the file to reflect the new load address. ARG2 may be of any type.

NOTE: If no arguments are entered then Z8E will redisplay the starting address, ending address, and the number of 256 byte pages of the last file loaded.

If Z8E detects a error in the file name specification the message "Syntax error" is printed on the console and the command terminates.

If Z8E is unable to locate the file on the specified drive the message "File not found" is printed on the console and the command terminates.

Z8E contains no facilties for converting .HEX (Intel Hex format) object files to loadable memory image. All files, regardless of type, are loaded into memory in exactly the same form as they appear on disk. To debug a .HEX file the user should first load the file with the CP/M LOAD command and save the file with the CP/M SAVE command which produces an absolute memory image loadable by Z8E. All .COM files are of course already in loadable form and no LOADing and SAVEing is required.

If the file will not fit into the TPA, Z8E will print the message:

```
Out of memory - Continue?
```

If the user answers "Y", Z8E will resume loading the file at address 100 hex if ARG2 was not entered, or at the address specified as ARG2. If the user types any other response, the loading process terminates and Z8E returns to the command level. However, the user may resume loading the file at a later time by issuing the I command and specifying the file name "." (a single period). The user may choose to specify a new starting load address following the period; if ARG2 is omitted then the load address defaults back to 100 hex, the start of the TPA. If the user has done any subsequent disk I/O (such as loading a new file of disassembling to disk) in between the time loading was suspended and then restarted, Z8E will treat the file name "." as a syntax error.

The user may occasionally need to overlay a section of code in a program which already resides in memory with input from a file on disk, for example in modifying a BIOS in preparation for MOVCPM. While this is possible with loaders which process .HEX object files, it is not feasible with Z8E. The user can circumvent this limitation by loading the file from disk into an unused section of memory and then using Z8E′s move command to move only the data needed to accomplish the overlay.

J Animated Full Screen Debugger

The J command provides the user with the ability to "see" inside the Z80 as it executes a program. The Z8E animated debugger allows the user to view registers, memory, and instructions while the Z80 is simultaneously executing code. In addtion the J command provides the user with the ability to interactively single-step through a program using the full screen facilities of the command. The format of the J command is:

```
*J   [/] [*] [ARG1] [ARG2]
```

USE OF THE J COMMAND FOR SINGLE STEPPING IS DESCRIBED AT THE END OF THIS SECTION. THIS SECTION DESCRIBES THE NON-INTERACTIVE VERSION OF THE J COMMAND DURING WHICH THE USER TURNS OVER COMPLETE CONTROL OF THE EXECUTION OF THE PROGRAM UNDER TEST TO Z8E.

ARG1 is the starting address of the display and may be of any valid argument type. For example, the user may specify *J P to resume execution at the point where it had previouly been stopped.

The slash and star control subroutine tracing as follows:

"/" Slash informs Z8E not to trace any subroutines at all.

"*" Asterisk informs Z8E not to trace any subroutine calls to addresses located in the range 0 to FF. This feature is intended to screen out calls to location 5 (BDOS) in order to prevent Z8E's and the user's stack from becoming hopelessly entangled.

ARG2 represents an optional timeout paramter which affects the speed at which instructions are executed. This number may be in the range 0 - 255, with 10 (decimal) as the default if no value is entered. A timeout value of 10 yields approximately a one half second delay between the execution of sequential instructions. A value of 0 represents NO time delay and is in actuality the fastest rate a which the J command can run.

Once the J command commences, Z8E takes over the Z80 and furnishes the user with a "peephole" into the CPU. Z8E executes one instruction at a time in the user program pausing after each one to dynamically update the screen display. The J command divides the screen into three areas: register map, disassembled code, and memory window. The register map displays all registers on the top two lines of the screen along with the contents of the F register which is shown in mnemonic form. Z8E also disassembles 18 instructions based on the current PC value and displays them on the screen; finally, using the parameters entered in the W command, Z8E snapshots a block of memory and displays it as a window on the screen.

Execution of the user program continues until any non-numeric key on the keyboard is pressed which ends the command. If a numeric key is pressed, then Z8E responds by changing the timeout parameter on the fly. The user may use the keys 0 - 9 as a throttle to govern the execution speed. Zero being the fastest; nine being the slowest.

The command also terminates whenever a user defined breakpoint is reached. That is, if the user had set a breakpoint via the B command and this address is reached the J command ends and Z8E prompts the user for the next command. If the breakpoint had a pass count associated with it, the pass count must reach zero before the J command will terminate.

USING THE J COMMAND FOR SINGLE STEPPING

Z8E permits the user to single-step through a program while allowing a continuous full-screen view of the registers, code being executed, and the contents of a block of memory as specified by the K command. In order to invoke the full screen single-step the user enters the following command:

```
*J  [/]  [*]
```

/ instructs Z8E not to trace any subroutines at all

* instructs Z8E not to trace any subroutines location below address 100H and is specifically designed to allow the user the option of not becoming tangled in BDOS and BIOS.

Note that this version of the J command is differentiated from the non-interactive version by the absence of any argument indicating a execution address.

This version allows the user to execute one instruction in his program and then regain control at the Z8E command level. Z8E will execute the instruction pointed to by the user's current PC. After the instruction is executed an ARROW (=>) points to the next instruction to be executed.

The / and * options are only valid if the next instruction to be executed is a CALL. If the program counter is pointing at any other instruction then the / and * have no effect.

K Set Memory Window Parameters for Use With the J Command

The K command sets the starting address and block size of the memory window display during the J command. The format of the command is:

```
*K  ARG1 [,ARG2]
```

ARG1 represents the starting address of the memory block.

ARG2 is an optional size paramter, if omitted the block size defaults to the maximum.

The maximum block size is 144 decimal which is 90 hex. The starting address of the memory block can be anywhere in memory; it does not have to be within the confines of the user program.

M Move Memory

The M command allows the user to move blocks of date from any address in memory to any other address in memory. The format of the command is:

```
*M   ARG1  ARG2  ARG3
```

where ARG1 = the starting address of the source data block

ARG2 = the ending address of the source data block

ARG3 = the starting address of the destination data block

arguments may be of any type

Z8E automatically decides whether a head-to-head or tail-to-tail move is required based on the three arguments entered. If a head-to-head move is needed then the first byte of the source data block will be written to the first byte position of the destination data block; the second byte of the source data block will be written to the second byte position of the destination data block, and so on until the ending address of the destination data block is reached.

On the other hand, if a tail-to-tail move is necessary Z8E will move the last byte of the source data block to the last byte position of the destination data block, followed by the second to last byte of the source data block to the second to last byte position of the destination data block, and so on until the starting address of the destination block is reached.

A tail to tail move would be necessary in the following example to prevent the overwriting of the destination data block:

```
*M  1000 100F 1008
```

N Output to I/O Ports Without Pre-Read

This command allows the user to output data to an I/O port without first reading the port (as occurs in the Q command). The format of the command is:

```
*N   [ARG1]
```

where ARG1 is the port number to which the data will be written.

If ARG1 is omitted then Z8E uses the last port address which had been input by a previous N or Q command.

Z8E will prompt the user by displaying the current port number on the left hand side of the console and postioning the cursor two spaces the the right. At this point the user can enter the data to be sent to the port in the form of an ARGUMENT-STRING. The ARGUMENT-STRING allows the user to mix various argument types such as hex data and ASCII literal strings. Of course the user can elect to merely output single bytes if desired. The N command is particularly useful when programming various Z80 peripheral chips such as the DMA and SIO chips which expect initialization bytes to arrive in a stream without intervening reads.

```
*N 80
80  'T',00,12#,998
```

This ARGUMENT-STRING would be evaluated into the 5 bytes: 54 00 0C 09 98. These five bytes would be sent to port 80 via an OTIR instruction. No delay occurs between successive bytes.

After the data has been entered and after it has been sent to the I/O port Z8E reprompts the user by displaying the same port number. This gives the user to oppportunity to send addtional data to the same port. However, by not entering data the user can change the current port address by entering any of the following:

CARRIAGE RETURN The next sequential port number in ascending order becomes the current port address.

UP ARROW The next sequential port number in descending order becomes the current port address.

=ARG Any argument appearing immediately after the equal sign (no intervening spaces) is evaluated as an 8 bit number, and if found to be valid then it becomes the new current port address.

PERIOD Terminate command

The user can also monitor an I/O port with the N command by enclosing the port number on the command line in parentheses. Monitor mode via the N command is identical to that of the Q command (see Q command).

O Output Current Breakpoints to Console

The O command allows the user to view all breakpoints currently in effect. The format of the command is:

```
        *O

                no arguments are required
```

If Z8E finds a symbol name corresponding to the absolute hex address of a breakpoint address in the symbol table (if a symbol table exists) then the symbol name as well as the memory address is displayed. If no symbol corresponding to the address is found only the hex address is displayed.

If any pass counts are currently in effect they are displayed next to the breakpoint address with which they are associated.

P Examine/Modify PSW (Flag Register)

The P command provides a convenient method of examining and optionally modifying the F(lag) register in the user register set. The format of the command is:

```
*P
```

no arguments are required on the command line

Upon receipt of the P command Z8E displays the mnemonics corresponding to the current state of the four user-modifiable bits (sign, carry, zero, parity) in Flag register:

MNEMONIC	MEANING	BIT STATUS
P	positive	reset
M	minus	set
NC	no carry	reset
C	carry	set
PO	parity odd	reset
PE	parity even	set
NZ	not zero	reset
Z	zero	set

Z8E prints the mnemonic corresponding to the current state of each of the four flag bits. Z8E then issues a carriage return/line feed and pauses for user input. The user may modify any of the four flag bits by typing the appropriate mnemonic followed by a carriage return. The user may enter multiple mnemonics by separating each one with a space.

If no mnemonics are entered, no flags bits are altered and the command terminates.

If an invalid flag bit mnemonic is entered Z8E prints a question mark.

Q Query I/O Ports

The Q command allows the user flexible access to I/O ports by providing the ability to perform single byte input, continuous input (monitor mode), and single or multi-byte output following a pre-read of the port. The format of the command is:

```
*Q   [(] [ARG1] [)]
```

where ARG1 is an 8 bit port address in the range 0 - 255

ARG1 may be any symbol type, however if a 16 bit value is specified only the low order byte is significant

If no argument is given Z8E will use the most recent port number as entered by the user via an N or Q command.

If ARG1 is enclosed in parentheses Z8E will enter MONITOR MODE.

Upon receipt of ARG1 Z8E will read the specified I/O port and display the byte read as both 8 bit hexadecimal value and it's ASCII equivalent. Command options once a byte has been read from the I/O port are as follows:

SINGLE BYTE INPUT

By entering a SPACE immediately following the displayed contents of the I/O port the user can instruct Z8E to continue reading from the same I/O port:

```
*Q EE
EE   24  $  <space>
EE   24  $
```

By entering a CARRIAGE RETURN following the displayed contents of the I/O port the user can instruct Z8E to read the next port number (ascending order):

```
*Q EE
EE   24  $  <cr>
EF   C1  A
```

By entering a caret "^" following the displayed contents of the I/O port the user can instruct

Z8E to read the previous port number (descending order):

```
*Q EE
EE   24  $   ^   (up arrow entered by user)
ED   06  ~
```

By entering an equal sign "=" followed by a valid argument, the user can switch to reading a new port address:

```
*Q EE
EE   24  ´$´  =90
90   BF  ´?´
```

CONTINUOUS INPUT (MONITOR MODE)

Z8E provides the user with the ability to monitor an input port. Z8E will continously read the selected input port and display the contents on the screen. Z8E displays the byte in both hex and binary. This feature is useful in the testing of I/O ports. Depressing any key on the keyboard exits monitor mode.

MULTI-BYTE OUTPUT

Following the read of an I/O port the user may elect to output data. The user may enter an ARGUMENT-STRING which will be sent to the port on a byte by byte basis with no intervening reads between outputs as shown below:

```
*Q 50
50  44  ´D´ 23,9,´B2E´,00,F723,81
            ^                    ^

            string  as entered by
            user appears  between
            arrows
```

The data as entered by the user in this example would first be converted to the 9 bytes shown below:

```
23 09 42 32 45 00 F7 23 81
```

These 9 bytes would then be sent to port 50 one byte after another without any intervening reads or status checks.

R Examine/Modify Register Contents

The R command allows the user to examine and optionally modify registers and register pairs in the user register set. The format of the command is:

```
*R   ARG1  <cr> or space
```

where ARG1 is any of the 22 register mnemonics listed below:

```
A    B    C    D    E    H    L
AF   BC   DE   HL   IX   IY   SP
AF'  BC'  DE'  HL'  I    R    P    PC
```

(the program counter may be specified as either P or PC)

To examine a register the user enters a mnemonic from the above list followed by a carriage return or a space. Z8E will display the current contents of the register on the same line. At this point the user has the option of entering an argument of any type if the contents of the register or register pair are to be changed. The replacement value may be terminated by either a carriage return or a space. If no value is entered Z8E issues a carriage return/line feed and waits for the next register mnemonic to be entered.

If the user specifies a 16 bit value as the new contents of an 8 bit register only the low order byte of the value is used.

The command terminates when a carriage return or space is entered when Z8E is waiting for a register mnemonic.

S Single Step

The S command allows the user to execute a program instruction by instruction. The S command provides for full tracing of the user program. The format of the command is:

```
*S   [/] [ARG1]  <cr>
```

where ARG1 is the number of instructions to execute in the user program, if no argument is given Z8E defaults to 1

ARG1 may be of any type

The slash "/" allows the user control over the tracing of subroutines. If a slash is included before the count (if a count is entered), or if the slash is the only character on the command line then subroutines will not be traced. A slash affects only CALL instructions which lie within the range of ARG1. In the most typical case no ARG1 is present and the single step count defaults to 1. If the current PC, 1000 in this example, is pointing to a call instruction then the command:

```
*S /
```

```
1000  CD 56 30  RASRTN: CALL  ANYSUB
1003  FE 04             CP    4
1005  CA 17 10          JP    Z,AHEAD
```

will cause the entire subroutine ANYSUB to be executed and control will return to the user at address 1003.

If ARG1 is omitted Z8E will transfer control to the user program and one instruction, the one pointed to by the current contents of the user's program counter, will be executed. Following the execution of the instruction (or group of instructions if ARG2 was greater than 1) Z8E regains control and automatically displays the current contents of all the user registers.

The user may optionally indicate that more than one instruction is to be executed by entering a value greater than 1 for ARG1. Z8E will transfer control to the user program and regain control only when the specified number of instructions have been executed. This feature is useful in debugging small loops; in that the user can set ARG1 equal to the number of instructions in the range of the loop. Z8E will display the register contents after each instruction of the loop is executed and return control to the user after every iteration of the loop.

The single step command always causes the execution of the instruction pointed to by the current contents of the user's program counter. This is the instruction that appears in disassembled form as part of the output of the "X" command (display machine state). Bear in mind that ARG1 is not the address at which single stepping is to begin; it is a count to the number of instructions to execute. If the user desires to single step at an address other than the one contained in the program counter, then the PC register must be modified via the R command before the single step command is issued to Z8E.

Allowing the convenience of entering "S" <cr> to execute one instruction has the side effect of not allowing the user to abort the command in between the time the "S" is typed and the <cr> is entered by simply omitting an argument and typing <cr>. If you change your mind and want to cancel the command, type in an invalid argument as ARG1. This will cause a question mark to be displayed; however, no instruction will be executed.

During block tracing (ARG1 greater than 1) the command may be terminated by hitting any key on the keyboard.

The S command does not relocate instructions before execution as does the G command (see G command). Hence, it is not possible to single step through each iteration of a DJNZ $ instruction.

U Write Symbol Table to Disk

The U command allows the user to write the current symbol table to a disk file. The format of the command is:

```
*U   ARG1
```

ARG1 is the name of the file to which the symbol table is to be written.

This command is useful to save any symbol names entered by the user via the A command. The entire symbol table is written to disk using the format of a .SYM file (see appendix A). The table can be subsequently loaded at the next invokation of Z8E.

Note that since the file is stored as a .SYM formatted file the user should use a file name extension that begin with the letter "S". This is due to the fact that the next time Z8E loads this symbol file it will examine the the first character of the file name extension. If the first character is an "S" the format is assumed to be .SYM and the symbol table is built accordingly; the appearance of any other letter is taken to indicate a .PRN file.

If a file with the name ARG1 already exists on disk it will be deleted.

V Verify two memory blocks

The V command allows the user to compare two blocks of memory. Z8E will display all differences between the two. The format of the command is:

```
*V    ARG1   ARG2   ARG3

      where  ARG1  =  the starting address of memory block 1

             ARG2  =  the ending address of memory block 1

             ARG3  =  the starting address of memory block 2
```

Z8E compares memory block 1 to memory block 2 byte by byte. If a mismatch occurs Z8E will display the address in each block at which the mismatch was found, as well as the byte contained at each address. The comparison continues until the ending address is reached.

The user may halt the command at any time by depressing any key on the keyboard.

W write memory to disk

The W command allows the user to write the contents of memory to a disk file. The format of the command is:

```
*W  arg1 [arg2  arg3]
```

ARG1 is the name of a file to which writing will take place.

ARG2 and ARG3 are the optional starting and ending addresses of the portion of memory to be written to the disk. If the addresses omitted then the memory block to be written is defined by the starting and ending addresses of the last file loaded. These addresses can be redisplayed by entering the I command with no arguments.

Z8E always deletes any file on disk whose name is the same as ARG1. If no file by this name exists then Z8E will automatically create it.

Z8E will echo the starting memory address and continually update the ending memory address as the writing to disk takes place.

X display machine state

The X command displays the current contents of all user registers. The format of the command is:

```
*X
```

no arguments are required

Z8E displays displays all registers, except the I register and the R register, on two lines of the console. In addition, the instruction pointed to by the user´s program counter is disassembled and displayed on the second line. Think of this as the "on deck" instruction: the instruction that will be executed upon the receipt of the next G (GO) or S (SINGLE STEP) command.

To inspect the I or R registers use the R command.

Y fill memory

The Y command fills a user specified block of memory with a user specified pattern of bytes, the length of which is limited only by the length of Z8E′s input buffer which is 80 bytes long:

```
*Y  ARG1  ARG2  ARG3  <cr>
```

where ARG1 = the starting address of the block to fill

ARG2 = the ending address of the block to fill

ARG3 = is the data pattern to be written to memory. ARG3 is evaluated by Z8E as type ARGUMENT-STRING which may be of any length in the range of 1 through the number of bytes remaining in the input buffer once ARG1 and ARG2 have been input.

The Y command gives the user the capability to initialize memory to any data pattern. The capability of entering multi-byte strings as the data pattern with which to fill memory allows the user to store repeating patterns of data in memory with a single command. For example if the user entered the command:

```
*Y 1000 127C ′abcd′,16,77
```

Z8E would begin writing the 6 byte pattern (61 62 63 64 16 77) entered as ARG3 starting at address 1000. This pattern would repeat at address 1006, 100C, 1012, etc.

The command ends after a byte is written to the ARG2 address even if this byte does not represent the last byte in the ARG3 block. In the above example the command would end when a byte is written to address 127C even if that byte is not 77.

Z disassemble command

The Z command allows the user to disassemble a block of data. Z8E performs disassembly, which is the translation of binary memory data into source code format, using the full Z80 instruction set and Zilog mnemonics. The resultant source code may be directed to the console or to the console and a disk file simultaneously. Z8E also allows the user to disassemble interactively when ARG2 is equal to 1. The format of the command is:

```
*Z  ARG1 [ARG2 ARG3]  <cr>
```

where ARG1 = the start address at which disassembly is to begin

ARG2 = is optional and represents the upper limit of the disassembly process (see details below)

ARG3 = is an optional file name specification for disassembly to disk

ARG1 may be of any argument type.

ARG2 is treated in one of two ways depending on its value:

1) If ARG2 evaluates to a number between 1 and 255 (decimal) Z8E will disassemble in "block mode" and ARG2 becomes a count of the number of instructions per block to disassemble. As will be explained below, Z8E pauses after each block is disassembled and allows the user to continue or to terminate the command.

   If ARG2 is omitted altogether a default block size of 1 is used.

   Whenever ARG2 equals 1, either explicitly or by default, Z8E allows interactive disassembly which allows the user to choose the output format of the data. Interactive disassembly is discussed below.

2) If ARG2 evaluates to a number greater than 255 it is assumed to be an ending address. In this case disassembly will proceed from starting address (ARG1) to ending address (ARG2) and no user intervention is required.

ARG3, if present, is assumed to be the name of a disk file into which the disassembled output will be written. Z8E searches the specified disk for the named file. If the file is found, then all disassembled output will be written to it, overwriting any data that existed there. If the file does not exist the file will be created using the name specified in ARG3.

NOTE: If ARG3 is present ARG2 must be explicitly specified, otherwise Z8E will mistakenly treat the file name as ARG2.

Z8E outputs to the console using the following format:

```
     ADDRESS        OBJECT CODE       LABEL:   OPCODE    OPERAND
```

Z8E writes to disk using the following formart:

```
                                      LABEL:   OPCODE    OPERAND
```

Z8E disassembles memory block by block in the user specified block size. After each block is output Z8E pauses for user input. A carriage return input by the user terminates the command, while any other character causes the next block to be disassembled (unless interactive mode is in effect). Perhaps the most convenient way to disassemble is to specify a count of one, either explicity or by omitting ARG2, and to use the space bar as an on/off switch. Holding down the space bar produces output, releasing the space bar ends output.

Z8E′s disassembler is especially powerful when used in conjunction with the symbol facility. By building a symbol table with both .PRN and .SYM files, and/or creating user defined symbol names via the A command, the user can virtually recreate an assembler output listing (minus the comments) with Z8E inserting labels and symbolic operands wherever possible.

If Z8E cannot match an operand in the disassembled instruction to a corresponding symbol in the symbol table, or if no symbol table exists, Z8E uses the hexadecimal value.

If multiple symbols in the symbol table are equal to the same 16 bit value or address, Z8E disassembles using the first symbol name encountered in the search of the symbol table which is equated to the 16 bit operand specified in the instruction being disassembled. This will unavoidably produce an occasional misnamed operand when more than one symbol name is equated to the same 16 bit value.

Z8E does not substitute symbol names in those Z80 instructions which reference 8 bit immediate data (ie. LD A,24H). Eight bit immediate data is disassembled as a quoted ASCII character if it′s absolute value is in the range 20 hex to 7E hex; otherwise, it is disassembled as a hex byte.

Output by Z8E to a disk file is instantly assemblable by most any assembler which accepts Zilog mnemonics without any modifications other than adding an END statement at the end of the file.

When disassembling a block of memory (starting address to ending address) the disassembly process may be halted at any time by depressing any key on the keyboard.

Interactive disassembly allows the user to specify the format of the source code produced by disassembly on a line by line basis. Interactive mode, which is always in effect whenever ARG2 is equal to 1, causes Z8E to pause after each instruction is disassembled. This pause for input permits the user to enter one of the following commands to choose the desired output format:

CHARACTER	OUTPUT FORMAT	EXAMPLE
A	ASCII DEFB	DEFB ´Q´
B	HEX DEFB	DEFB 23H
C	CODE	EX DE,HL
D	HEX DEFW	DEFW 02FCH or DEFW LABEL
;	add COMMENT	;This is a Comment
carriage return	(terminate command)	
any other character	PROCEED TO THE NEXT INSTRUCTION	

ASCII DEFB:
The contents of memory at the current disassembly address is converted to a quoted ASCII character. Values less than hexadecimal 20 (ASCII space) or greater than hexadecimal 7E (ASCII tilde) cannot be disassembled into this format.

HEX DEFB:
The 8 bit contents of memory at the current disassembly address are converted to a hex byte.

CODE:
This is the normal default for disassembly. As Z8E moves on to a new address it will always display the contents of memory as a Z80 instruction. The "C" is only needed to redisplay the contents of memory as an instruction had one of the other characters (A, B, or D) already have been entered.

HEX DEFW:
The contents of the two bytes of memory starting at the loaction of the current disassembly address are output as a define word directive. The byte pointed to directly by the current disassembly address becomes the low order byte of the operand. The byte at disassembly address plus one becomes the high order byte.

NOTE:

> If Z8E had just disassembled a multi-byte Z80 instruction and the user entered any of the characters listed above (A, B, C, or D) only the first byte, or first two for "D", of the instruction would be converted to the requested format. The remaining bytes of the instruction would be treated as a new Z80 instruction once the user proceeded to the next disassembly address.

ADDING COMMENTS
Z8E allows the user to add one comment per line of disassembled code. If MAXLEN is set to 6 then comments may be up to 29 characters in length. If MAXLEN is set to 14 then comments may be up to 16 characters in length.

If during disassembly, Z8E encounters data which cannot be disassembled into a valid Z80 instruction it will display the data as DEFB´s.

APPENDIX A

FILE FORMAT FOR SYMBOL TABLES

Z8E is currently set up to be able to read any of the listing files which appear below:

1.	Microsoft	MACRO-80	V3.37	.PRN Files	May 8, 1980
2.	Microsoft	MACRO-80	V3.44	.PRN Files	Dec 9, 1981
3.	Microsoft	LINK-80	V3.44	.SYM Files	Dec 9, 1981
4.	SLR Systems	Z80ASM	V1.07	.LST Files	
5.	SLR Systems	SLRNK	V1.07	.SYM Files	

The unique characteristics of each are:

MACRO-80 V3.37

Z8E searches for the 8 byte string "Symbols:" in the file. Once this string is found, Z8E expects an ASCII carriage return character and an ASCII line feed character to be the next two bytes in the file. The symbol table listing should begin in the next character position in the file.

Each line of the symbol table listing contains four symbol names and an associated address.

If the character following the symbol's hex value is an apostophe, the symbol is considered to be program relative. If the user specified a bias in the command line the bias will be added to the symbol's value.

If the character following the symbol's hex value is an "I" (meaning that the symbol is globally defined) then the character following the "I" is examined. If this character is an apostrophe it is considered to be program relative and the bias, if specified is added to the value.

If the character following the hex symbol value or the "I" is any character besides an apostrophe, the symbol is considered absolute and the bias will not be added.

The file should be terminated with the CP/M end-of-file character (control Z which is equivalent to a hex 1A).

If the string "Symbols" is never found, Z8E prints the message: Symbol Table not Found

APPENDIX A

MACRO-80 V3.44

Z8E searches for the 8 byte string "Symbols:" in the file. Once this string is found, Z8E expects an ASCII carriage return character and an ASCII line feed character to be the next two bytes in the file. The symbol table listing should begin in the next character position in the file.

In this release of MACRO-80 the format of the symbol table is completely opposite of V3.37. That is, the hex value appears before the symbol name. In addition, these hex value/symbol name combination appear three per line.

The character appearing after the hex value is interpreted as described for version 3.37.

If the string "Symbols" is never found, Z8E prints the message: Symbol Table not Found

LINK-80 V3.44

LINK-80 can optionally produce a link map (.SYM file) which lists all globally defined symbols if the user specifies the "Y" option the L80 command line. Z8E treats all symbols names loaded from a LINK-80 .SYM file as absolute (non-relocatable) addresses. Nevertheless, if the user specifies a bias, it will be added to every symbol value read in from the .SYM file.

Z8E expects the first symbol value in a .SYM file to begin in the first byte position in the file. Each symbol value consists of four hexadecimal bytes in ASCII followed by a tab character. Immediately after the tab character is the symbol name which may be between one and six alphanumeric characters in length. The symbol name is followed by a tab and the sequence repeats. Every fourth symbol value/symbol name pair should be followed by a carriage return and line feed.

The file should be terminated with the CP/M end-of-file character (control Z which is equivalent to a hex 1A).

Z80ASM

Z80ASM may be configured to produce either 80 or 132 column output.

Z8E searches for the 8 byte string "Symbol Table:" in the file. This string need not be at the beginning of the file; Z8E will scan the entire file looking for it. Once this string is found, Z8E expects an ASCII carriage return character and an ASCII line feed character to be the next two bytes in the file. The symbol table listing should begin in the next character position in the file.

In a Z80ASM .LST file the hex value appears before the symbol name. Hex value/symbol name combinations appear three per line. Z80ASM symbol names may contain up to 16 characters. Z8E will accept the first 14 characters of a symbol name if MAXLEN is set to 14 or the first 6 characters if MAXLEN is set to 6.

If the string "Symbol Table:" is never found, Z8E prints the message:

Symbol Table Not Found

SLRNK

SLRNK can optionally produce a link map (.SYM File) similar to the one produced by Link-80. Z8E treats all symbols loaded from a SLRNK .SYM file as absolute symbols. However, as in the case of Link-80 .SYM files, Z8E will add a relocation bias to each symbol if one is specified.

Each symbol value in a SLRNK .SYM file consists of four hexadecimal bytes followed by a space followed by the symbol name. The symbol name is followed by two ASCII tab characters.

Use SLRNK´s /M option to produce a link map.

NOTE:

While reading in a MACRO-80 .PRN file, or a Z80ASM .LST file, Z8E is capable of reading an entire assembly listing file looking for the "Symbols:" string or "Symbol Table:" string. These strings need not be located at the beginning of the file. However, the loading of the symbol table will be speeded up considerably if the symbol table is the only data in the file. This is accomplished quite easily in both MACRO-80 by turning off the listing during an assembly through the use of the .XLIST directive. The listing can then be turned back on just prior to the END directive via a .LIST directive to ensure that the symbol table is written to disk.

If you are using Z80ASM use the /S option to instruct Z80ASM to produce a symbol file.

Z8E is able to process symbol tables which occupy multiple pages in any of the file types mentioned above. Headings which precede each page are automatically ignored by Z8E.

APPENDIX B - ZILOG MNEMONICS

```
0049                  NN     EQU   49H        ;8 BIT OPERAND
123F                  NNNN   EQU   123FH      ;16 BIT   OPERAND
0036                  INDEX  EQU   36H        ;INDEX REGISTER INDEX

010B    8E                   ADC   A,(HL)
010C    DD 8E 36             ADC   A,(IX+INDEX)
010F    FD 8E 36             ADC   A,(IY+INDEX)
0112    8F                   ADC   A,A
0113    88                   ADC   A,B
0114    89                   ADC   A,C
0115    8A                   ADC   A,D
0116    8B                   ADC   A,E
0117    8C                   ADC   A,H
0118    8D                   ADC   A,L
0119    CE 49                ADC   A,NN
011B    ED 4A                ADC   HL,BC
011D    ED 5A                ADC   HL,DE
011F    ED 6A                ADC   HL,HL
0121    ED 7A                ADC   HL,SP


0123    86                   ADD   A,(HL)
0124    DD 86 36             ADD   A,(IX+INDEX)
0127    FD 86 36             ADD   A,(IY+INDEX)
012A    87                   ADD   A,A
012B    80                   ADD   A,B
012C    81                   ADD   A,C
012D    82                   ADD   A,D
012E    83                   ADD   A,E
012F    84                   ADD   A,H
0130    85                   ADD   A,L
0131    C6 49                ADD   A,NN
0133    09                   ADD   HL,BC
0134    19                   ADD   HL,DE
0135    29                   ADD   HL,HL
0136    39                   ADD   HL,SP
0137    DD 09                ADD   IX,BC
0139    DD 19                ADD   IX,DE
013B    DD 29                ADD   IX,IX
013D    DD 39                ADD   IX,SP
013F    FD 09                ADD   IY,BC
0141    FD 19                ADD   IY,DE
0143    FD 29                ADD   IY,IY
0145    FD 39                ADD   IY,SP


0147    A6                   AND   (HL)
0148    DD A6 36             AND   (IX+INDEX)
014B    FD A6 36             AND   (IY+INDEX)
014E    A7                   AND   A
014F    A0                   AND   B
0150    A1                   AND   C
0151    A2                   AND   D
```

```
0152     A3                      AND    E
0153     A4                      AND    H
0154     A5                      AND    L
0155     E6 49                   AND    NN


0157     CB 46                   BIT    0,(HL)
0159     DD CB 36 46             BIT    0,(IX+INDEX)
015D     FD CB 36 46             BIT    0,(IY+INDEX)
0161     CB 47                   BIT    0,A
0163     CB 40                   BIT    0,B
0165     CB 41                   BIT    0,C
0167     CB 42                   BIT    0,D
0169     CB 43                   BIT    0,E
016B     CB 44                   BIT    0,H
016D     CB 45                   BIT    0,L


016F     CB 4E                   BIT    1,(HL)
0171     DD CB 36 4E             BIT    1,(IX+INDEX)
0175     FD CB 36 4E             BIT    1,(IY+INDEX)
0179     CB 4F                   BIT    1,A
017B     CB 48                   BIT    1,B
017D     CB 49                   BIT    1,C
017F     CB 4A                   BIT    1,D
0181     CB 4B                   BIT    1,E
0183     CB 4C                   BIT    1,H
0185     CB 4D                   BIT    1,L


0187     CB 56                   BIT    2,(HL)
0189     DD CB 36 56             BIT    2,(IX+INDEX)
018D     FD CB 36 56             BIT    2,(IY+INDEX)
0191     CB 57                   BIT    2,A
0193     CB 50                   BIT    2,B
0195     CB 51                   BIT    2,C
0197     CB 52                   BIT    2,D
0199     CB 53                   BIT    2,E
019B     CB 54                   BIT    2,H
019D     CB 55                   BIT    2,L


019F     CB 5E                   BIT    3,(HL)
01A1     DD CB 36 5E             BIT    3,(IX+INDEX)
01A5     FD CB 36 5E             BIT    3,(IY+INDEX)
01A9     CB 5F                   BIT    3,A
01AB     CB 58                   BIT    3,B
01AD     CB 59                   BIT    3,C
01AF     CB 5A                   BIT    3,D
01B1     CB 5B                   BIT    3,E
01B3     CB 5C                   BIT    3,H
01B5     CB 5D                   BIT    3,L


01B7     CB 66                   BIT    4,(HL)
```

```
01B9    DD CB 36 66          BIT   4,(IX+INDEX)
01BD    FD CB 36 66          BIT   4,(IY+INDEX)
01C1    CB 67                BIT   4,A
01C3    CB 60                BIT   4,B
01C5    CB 61                BIT   4,C
01C7    CB 62                BIT   4,D
01C9    CB 63                BIT   4,E
01CB    CB 64                BIT   4,H
01CD    CB 65                BIT   4,L

01CF    CB 6E                BIT   5,(HL)
01D1    DD CB 36 6E          BIT   5,(IX+INDEX)
01D5    FD CB 36 6E          BIT   5,(IY+INDEX)
01D9    CB 6F                BIT   5,A
01DB    CB 68                BIT   5,B
01DD    CB 69                BIT   5,C
01DF    CB 6A                BIT   5,D
01E1    CB 6B                BIT   5,E
01E3    CB 6C                BIT   5,H
01E5    CB 6D                BIT   5,L

01E7    CB 76                BIT   6,(HL)
01E9    DD CB 36 76          BIT   6,(IX+INDEX)
01ED    FD CB 36 76          BIT   6,(IY+INDEX)
01F1    CB 77                BIT   6,A
01F3    CB 70                BIT   6,B
01F5    CB 71                BIT   6,C
01F7    CB 72                BIT   6,D
01F9    CB 73                BIT   6,E
01FB    CB 74                BIT   6,H
01FD    CB 75                BIT   6,L

01FF    CB 7E                BIT   7,(HL)
0201    DD CB 36 7E          BIT   7,(IX+INDEX)
0205    FD CB 36 7E          BIT   7,(IY+INDEX)
0209    CB 7F                BIT   7,A
020B    CB 78                BIT   7,B
020D    CB 79                BIT   7,C
020F    CB 7A                BIT   7,D
0211    CB 7B                BIT   7,E
0213    CB 7C                BIT   7,H
0215    CB 7D                BIT   7,L

0217    DC 123F              CALL   C,NNNN
021A    FC 123F              CALL   M,NNNN
021D    D4 123F              CALL   NC,NNNN
0220    CD 123F              CALL   NNNN
0223    C4 123F              CALL   NZ,NNNN
0226    F4 123F              CALL   P,NNNN
0229    EC 123F              CALL   PE,NNNN
022C    E4 123F              CALL   PO,NNNN
```

```
022F    CC 123F           CALL   Z,NNNN


0232    3F                CCF


0233    BE                CP   (HL)
0234    DD BE 36          CP   (IX+INDEX)
0237    FD BE 36          CP   (IY+INDEX)
023A    BF                CP   A
023B    B8                CP   B
023C    B9                CP   C
023D    BA                CP   D
023E    BB                CP   E
023F    BC                CP   H
0240    BD                CP   L
0241    FE 49             CP   NN


0243    ED A9             CPD
0245    ED B9             CPDR
0247    ED A1             CPI
0249    ED B1             CPIR


024B    2F                CPL


024C    27                DAA


024D    35                DEC   (HL)
024E    DD 35 36          DEC   (IX+INDEX)
0251    FD 35 36          DEC   (IY+INDEX)
0254    3D                DEC   A
0255    05                DEC   B
0256    0B                DEC   BC
0257    0D                DEC   C
0258    15                DEC   D
0259    1B                DEC   DE
025A    1D                DEC   E
025B    25                DEC   H
025C    2B                DEC   HL
025D    DD 2B             DEC   IX
025F    FD 2B             DEC   IY
0261    2D                DEC   L
0262    3B                DEC   SP


0263    F3                DI


0264    10 04             DJNZ   $+6
```

```
0266    FB                   EI

0267    E3                   EX    (SP),HL
0268    DD E3                EX    (SP),IX
026A    FD E3                EX    (SP),IY
026C    08                   EX    AF,AF'
026D    EB                   EX    DE,HL
026E    D9                   EXX

026F    76                   HALT

0270    ED 46                IM    0
0272    ED 56                IM    1
0274    ED 5E                IM    2

0276    ED 78                IN    A,(C)
0278    DB 49                IN    A,(NN)
027A    ED 40                IN    B,(C)
027C    ED 48                IN    C,(C)
027E    ED 50                IN    D,(C)
0280    ED 58                IN    E,(C)
0284    ED 60                IN    H,(C)
0286    ED 68                IN    L,(C)

0288    34                   INC    (HL)
0289    DD 34 36             INC    (IX+INDEX)
028C    FD 34 36             INC    (IY+INDEX)
028F    3C                   INC    A
0290    04                   INC    B
0291    03                   INC    BC
0292    0C                   INC    C
0293    14                   INC    D
0294    13                   INC    DE
0295    1C                   INC    E
0296    24                   INC    H
0297    23                   INC    HL
0298    DD 23                INC    IX
029A    FD 23                INC    IY
029C    2C                   INC    L
029D    33                   INC    SP

029E    ED AA                IND
02A0    ED BA                INDR
02A2    ED A2                INI
02A4    ED B2                INIR

02A6    E9                   JP    (HL)
02A7    DD E9                JP    (IX)
```

```
02A9     FD E9               JP    (IY)
02AB     DA 123F             JP    C,NNNN
02AE     FA 123F             JP    M,NNNN
02B1     D2 123F             JP    NC,NNNN
02B4     C3 123F             JP    NNNN
02B7     C2 123F             JP    NZ,NNNN
02BA     F2 123F             JP    P,NNNN
02BD     EA 123F             JP    PE,NNNN
02C0     E2 123F             JP    PO,NNNN
02C3     CA 123F             JP    Z,NNNN


02C6     38 04               JR    C,$+6
02C8     18 04               JR    $+6
02CA     30 04               JR    NC,$+6
02CC     20 04               JR    NZ,$+6
02CE     28 04               JR    Z,$+6


02D0     02                  LD    (BC),A
02D1     12                  LD    (DE),A
02D2     77                  LD    (HL),A
02D3     70                  LD    (HL),B
02D4     71                  LD    (HL),C
02D5     72                  LD    (HL),D
02D6     73                  LD    (HL),E
02D7     74                  LD    (HL),H
02D8     75                  LD    (HL),L
02D9     36 49               LD    (HL),NN


02DB     DD 77 36            LD    (IX+INDEX),A
02DE     DD 70 36            LD    (IX+INDEX),B
02E1     DD 71 36            LD    (IX+INDEX),C
02E4     DD 72 36            LD    (IX+INDEX),D
02E7     DD 73 36            LD    (IX+INDEX),E
02EA     DD 74 36            LD    (IX+INDEX),H
02ED     DD 75 36            LD    (IX+INDEX),L
02F0     DD 36 36 49         LD    (IX+INDEX),NN


02F4     FD 77 36            LD    (IY+INDEX),A
02F7     FD 70 36            LD    (IY+INDEX),B
02FA     FD 71 36            LD    (IY+INDEX),C
02FD     FD 72 36            LD    (IY+INDEX),D
0300     FD 73 36            LD    (IY+INDEX),E
0303     FD 74 36            LD    (IY+INDEX),H
0306     FD 75 36            LD    (IY+INDEX),L
0309     FD 36 36 49         LD    (IY+INDEX),NN


030D     32 123F             LD    (NNNN),A
0310     ED 43 123F          LD    (NNNN),BC
0314     ED 53 123F          LD    (NNNN),DE
0318     22 123F             LD    (NNNN),HL
```

```
031B    DD 22 123F        LD    (NNNN),IX
031F    FD 22 123F        LD    (NNNN),IY
0323    ED 73 123F        LD    (NNNN),SP


0327    0A                LD    A,(BC)
0328    1A                LD    A,(DE)
0329    7E                LD    A,(HL)
032A    DD 7E 36          LD    A,(IX+INDEX)
032D    FD 7E 36          LD    A,(IY+INDEX)
0330    3A 123F           LD    A,(NNNN)
0333    7F                LD    A,A
0334    78                LD    A,B
0335    79                LD    A,C
0336    7A                LD    A,D
0337    7B                LD    A,E
0338    7C                LD    A,H
0339    ED 57             LD    A,I
033B    7D                LD    A,L
033C    3E 49             LD    A,NN
033E    ED 5F             LD    A,R


0340    46                LD    B,(HL)
0341    DD 46 36          LD    B,(IX+INDEX)
0344    FD 46 36          LD    B,(IY+INDEX)
0347    47                LD    B,A
0348    40                LD    B,B
0349    41                LD    B,C
034A    42                LD    B,D
034B    43                LD    B,E
034C    44                LD    B,H
034D    45                LD    B,L
034E    06 49             LD    B,NN


0350    ED 4B 123F        LD    BC,(NNNN)
0354    01 123F           LD    BC,NNNN


0357    4E                LD    C,(HL)
0358    DD 4E 36          LD    C,(IX+INDEX)
035B    FD 4E 36          LD    C,(IY+INDEX)
035E    4F                LD    C,A
035F    48                LD    C,B
0360    49                LD    C,C
0361    4A                LD    C,D
0362    4B                LD    C,E
0363    4C                LD    C,H
0364    4D                LD    C,L
0365    0E 49             LD    C,NN


0367    56                LD    D,(HL)
```

```
0368    DD 56 36            LD   D,(IX+INDEX)
036B    FD 56 36            LD   D,(IY+INDEX)
036E    57                  LD   D,A
036F    50                  LD   D,B
0370    51                  LD   D,C
0371    52                  LD   D,D
0372    53                  LD   D,E
0373    54                  LD   D,H
0374    55                  LD   D,L
0375    16 49               LD   D,NN

0377    ED 5B 123F          LD   DE,(NNNN)
037B    11 123F             LD   DE,NNNN

037E    5E                  LD   E,(HL)
037F    DD 5E 36            LD   E,(IX+INDEX)
0382    FD 5E 36            LD   E,(IY+INDEX)
0385    5F                  LD   E,A
0386    58                  LD   E,B
0387    59                  LD   E,C
0388    5A                  LD   E,D
0389    5B                  LD   E,E
038A    5C                  LD   E,H
038B    5D                  LD   E,L
038C    1E 49               LD   E,NN

038E    66                  LD   H,(HL)
038F    DD 66 36            LD   H,(IX+INDEX)
0392    FD 66 36            LD   H,(IY+INDEX)
0395    67                  LD   H,A
0396    60                  LD   H,B
0397    61                  LD   H,C
0398    62                  LD   H,D
0399    63                  LD   H,E
039A    64                  LD   H,H
039B    65                  LD   H,L
039C    26 49               LD   H,NN

039E    2A 123F             LD   HL,(NNNN)
03A1    21 123F             LD   HL,NNNN

03A4    ED 47               LD   I,A

03A6    DD 2A 123F          LD   IX,(NNNN)
03AA    DD 21 123F          LD   IX,NNNN

03AE    FD 2A 123F          LD   IY,(NNNN)
03B2    FD 21 123F          LD   IY,NNNN
```

```
03B6    6E              LD   L,(HL)
03B7    DD 6E 36        LD   L,(IX+INDEX)
03BA    FD 6E 36        LD   L,(IY+INDEX)
03BD    6F              LD   L,A
03BE    68              LD   L,B
03BF    69              LD   L,C
03C0    6A              LD   L,D
03C1    6B              LD   L,E
03C2    6C              LD   L,H
03C3    6D              LD   L,L
03C4    2E 49           LD   L,NN

03C6    ED 4F           LD   R,A

03C8    ED 7B 123F      LD   SP,(NNNN)
03CC    F9              LD   SP,HL
03CD    DD F9           LD   SP,IX
03CF    FD F9           LD   SP,IY
03D1    31 123F         LD   SP,NNNN

03D4    ED A8           LDD
03D6    ED B8           LDDR
03D8    ED A0           LDI
03DA    ED B0           LDIR

03DC    ED 44           NEG

03DE    00              NOP

03DF    B6              OR   (HL)
03E0    DD B6 36        OR   (IX+INDEX)
03E3    FD B6 36        OR   (IY+INDEX)
03E6    B7              OR   A
03E7    B0              OR   B
03E8    B1              OR   C
03E9    B2              OR   D
03EA    B3              OR   E
03EB    B4              OR   H
03EC    B5              OR   L
03ED    F6 49           OR   NN

03EF    ED BB           OTDR
03F1    ED B3           OTIR
```

```
03F3     ED 79                OUT    (C),A
03F5     ED 41                OUT    (C),B


03F7     ED 49                OUT    (C),C
03F9     ED 51                OUT    (C),D
03FB     ED 59                OUT    (C),E
03FD     ED 61                OUT    (C),H
03FF     ED 69                OUT    (C),L
0401     D3 49                OUT    (NN),A

0403     ED AB                OUTD
0405     ED A3                OUTI

0407     F1                   POP    AF
0408     C1                   POP    BC
0409     D1                   POP    DE
040A     E1                   POP    HL
040B     DD E1                POP    IX
040D     FD E1                POP    IY

040F     F5                   PUSH    AF
0410     C5                   PUSH    BC
0411     D5                   PUSH    DE
0412     E5                   PUSH    HL
0413     DD E5                PUSH    IX
0415     FD E5                PUSH    IY

0417     CB 86                RES    0,(HL)
0419     DD CB 36 86          RES    0,(IX+INDEX)
041D     FD CB 36 86          RES    0,(IY+INDEX)
0421     CB 87                RES    0,A
0423     CB 80                RES    0,B
0425     CB 81                RES    0,C
0427     CB 82                RES    0,D
0429     CB 83                RES    0,E
042B     CB 84                RES    0,H
042D     CB 85                RES    0,L

042F     CB 8E                RES    1,(HL)
0431     DD CB 36 8E          RES    1,(IX+INDEX)
0435     FD CB 36 8E          RES    1,(IY+INDEX)
0439     CB 8F                RES    1,A
043B     CB 88                RES    1,B
043D     CB 89                RES    1,C
043F     CB 8A                RES    1,D
0441     CB 8B                RES    1,E
0443     CB 8C                RES    1,H
0445     CB 8D                RES    1,L
```

```
0447      CB 96               RES    2,(HL)
0449      DD CB 36 96         RES    2,(IX+INDEX)
044D      FD CB 36 96         RES    2,(IY+INDEX)
0451      CB 97               RES    2,A
0453      CB 90               RES    2,B
0455      CB 91               RES    2,C
0457      CB 92               RES    2,D
0459      CB 93               RES    2,E
045B      CB 94               RES    2,H
045D      CB 95               RES    2,L

045F      CB 9E               RES    3,(HL)
0461      DD CB 36 9E         RES    3,(IX+INDEX)
0465      FD CB 36 9E         RES    3,(IY+INDEX)
0469      CB 9F               RES    3,A
046B      CB 98               RES    3,B
046D      CB 99               RES    3,C
046F      CB 9A               RES    3,D
0471      CB 9B               RES    3,E
0473      CB 9C               RES    3,H
0475      CB 9D               RES    3,L

0477      CB A6               RES    4,(HL)
0479      DD CB 36 A6         RES    4,(IX+INDEX)
047D      FD CB 36 A6         RES    4,(IY+INDEX)
0481      CB A7               RES    4,A
0483      CB A0               RES    4,B
0485      CB A1               RES    4,C
0487      CB A2               RES    4,D
0489      CB A3               RES    4,E
048B      CB A4               RES    4,H
048D      CB A5               RES    4,L

048F      CB AE               RES    5,(HL)
0491      DD CB 36 AE         RES    5,(IX+INDEX)
0495      FD CB 36 AE         RES    5,(IY+INDEX)
0499      CB AF               RES    5,A
049B      CB A8               RES    5,B
049D      CB A9               RES    5,C
049F      CB AA               RES    5,D
04A1      CB AB               RES    5,E
04A3      CB AC               RES    5,H
04A5      CB AD               RES    5,L

04A7      CB B6               RES    6,(HL)
04A9      DD CB 36 B6         RES    6,(IX+INDEX)
04AD      FD CB 36 B6         RES    6,(IY+INDEX)
04B1      CB B7               RES    6,A
04B3      CB B0               RES    6,B
```

```
04B5    CB B1               RES   6,C
04B7    CB B2               RES   6,D
04B9    CB B3               RES   6,E
04BB    CB B4               RES   6,H
04BD    CB B5               RES   6,L


04BF    CB BE               RES   7,(HL)
04C1    DD CB 36 BE         RES   7,(IX+INDEX)
04C5    FD CB 36 BE         RES   7,(IY+INDEX)
04C9    CB BF               RES   7,A
04CB    CB B8               RES   7,B
04CD    CB B9               RES   7,C
04CF    CB BA               RES   7,D
04D1    CB BB               RES   7,E
04D3    CB BC               RES   7,H
04D5    CB BD               RES   7,L

04D7    C9                  RET
04D8    D8                  RET   C
04D9    F8                  RET   M
04DA    D0                  RET   NC
04DB    C0                  RET   NZ
04DC    F0                  RET   P
04DD    E8                  RET   PE
04DE    E0                  RET   PO
04DF    C8                  RET   Z

04E0    ED 4D               RETI
04E2    ED 45               RETN

04E4    CB 16               RL   (HL)
04E6    DD CB 36 16         RL   (IX+INDEX)
04EA    FD CB 36 16         RL   (IY+INDEX)
04EE    CB 17               RL   A
04F0    CB 10               RL   B
04F2    CB 11               RL   C
04F4    CB 12               RL   D
04F6    CB 13               RL   E
04F8    CB 14               RL   H
04FA    CB 15               RL   L

04FC    17                  RLA

04FD    CB 06               RLC   (HL)
04FF    DD CB 36 06         RLC   (IX+INDEX)
0503    FD CB 36 06         RLC   (IY+INDEX)
0507    CB 07               RLC   A
0509    CB 00               RLC   B
```

```
050B    CB 01           RLC   C
050D    CB 02           RLC   D
050F    CB 03           RLC   E
0511    CB 04           RLC   H
0513    CB 05           RLC   L


0515    07              RLCA


0516    ED 6F           RLD


0518    CB 1E           RR   (HL)
051A    DD CB 36 1E     RR   (IX+INDEX)
051E    FD CB 36 1E     RR   (IY+INDEX)
0522    CB 1F           RR   A
0524    CB 18           RR   B
0526    CB 19           RR   C
0528    CB 1A           RR   D
052A    CB 1B           RR   E
052C    CB 1C           RR   H
052E    CB 1D           RR   L


0530    1F              RRA


0531    CB 0E           RRC   (HL)
0533    DD CB 36 0E     RRC   (IX+INDEX)
0537    FD CB 36 0E     RRC   (IY+INDEX)
053B    CB 0F           RRC   A
053D    CB 08           RRC   B
053F    CB 09           RRC   C
0541    CB 0A           RRC   D
0543    CB 0B           RRC   E
0545    CB 0C           RRC   H
0547    CB 0D           RRC   L


0549    0F              RRCA


054A    ED 67           RRD


054C    C7              RST   0
054D    CF              RST   08H
054E    D7              RST   10H
054F    DF              RST   18H
0550    E7              RST   20H
0551    EF              RST   28H
0552    F7              RST   30H
0553    FF              RST   38H
```

```
0554    9E                    SBC   A,(HL)
0555    DD 9E 36              SBC   A,(IX+INDEX)
0558    FD 9E 36              SBC   A,(IY+INDEX)
055B    9F                    SBC   A,A
055C    98                    SBC   A,B
055D    99                    SBC   A,C
055E    9A                    SBC   A,D
055F    9B                    SBC   A,E
0560    9C                    SBC   A,H
0561    9D                    SBC   A,L
0562    DE 49                 SBC   A,NN
0564    ED 42                 SBC   HL,BC
0566    ED 52                 SBC   HL,DE
0568    ED 62                 SBC   HL,HL
056A    ED 72                 SBC   HL,SP


056C    37                    SCF


056D    CB C6                 SET   0,(HL)
056F    DD CB 36 C6           SET   0,(IX+INDEX)
0573    FD CB 36 C6           SET   0,(IY+INDEX)
0577    CB C7                 SET   0,A
0579    CB C0                 SET   0,B
057B    CB C1                 SET   0,C
057D    CB C2                 SET   0,D
057F    CB C3                 SET   0,E
0581    CB C4                 SET   0,H
0583    CB C5                 SET   0,L


0585    CB CE                 SET   1,(HL)
0587    DD CB 36 CE           SET   1,(IX+INDEX)
058B    FD CB 36 CE           SET   1,(IY+INDEX)
058F    CB CF                 SET   1,A
0591    CB C8                 SET   1,B
0593    CB C9                 SET   1,C
0595    CB CA                 SET   1,D
0597    CB CB                 SET   1,E
0599    CB CC                 SET   1,H
059B    CB CD                 SET   1,L


059D    CB D6                 SET   2,(HL)
059F    DD CB 36 D6           SET   2,(IX+INDEX)
05A3    FD CB 36 D6           SET   2,(IY+INDEX)
05A7    CB D7                 SET   2,A
05A9    CB D0                 SET   2,B
05AB    CB D1                 SET   2,C
05AD    CB D2                 SET   2,D
05AF    CB D3                 SET   2,E
05B1    CB D4                 SET   2,H
05B3    CB D5                 SET   2,L
```

```
05B5    CB DE              SET   3,(HL)
05B7    DD CB 36 DE        SET   3,(IX+INDEX)
05BB    FD CB 36 DE        SET   3,(IY+INDEX)
05BF    CB DF              SET   3,A
05C1    CB D8              SET   3,B
05C3    CB D9              SET   3,C
05C5    CB DA              SET   3,D
05C7    CB DB              SET   3,E
05C9    CB DC              SET   3,H
05CB    CB DD              SET   3,L

05CD    CB E6              SET   4,(HL)
05CF    DD CB 36 E6        SET   4,(IX+INDEX)
05D3    FD CB 36 E6        SET   4,(IY+INDEX)
05D7    CB E7              SET   4,A
05D9    CB E0              SET   4,B
05DB    CB E1              SET   4,C
05DD    CB E2              SET   4,D
05DF    CB E3              SET   4,E
05E1    CB E4              SET   4,H
05E3    CB E5              SET   4,L

05E5    CB EE              SET   5,(HL)
05E7    DD CB 36 EE        SET   5,(IX+INDEX)
05EB    FD CB 36 EE        SET   5,(IY+INDEX)
05EF    CB EF              SET   5,A
05F1    CB E8              SET   5,B
05F3    CB E9              SET   5,C
05F5    CB EA              SET   5,D
05F7    CB EB              SET   5,E
05F9    CB EC              SET   5,H
05FB    CB ED              SET   5,L

05FD    CB F6              SET   6,(HL)
05FF    DD CB 36 F6        SET   6,(IX+INDEX)
0603    FD CB 36 F6        SET   6,(IY+INDEX)
0607    CB F7              SET   6,A
0609    CB F0              SET   6,B
060B    CB F1              SET   6,C
060D    CB F2              SET   6,D
060F    CB F3              SET   6,E
0611    CB F4              SET   6,H
0613    CB F5              SET   6,L

0615    CB FE              SET   7,(HL)
0617    DD CB 36 FE        SET   7,(IX+INDEX)
061B    FD CB 36 FE        SET   7,(IY+INDEX)
061F    CB FF              SET   7,A
0621    CB F8              SET   7,B
```

```
0623    CB F9                  SET    7,C
0625    CB FA                  SET    7,D
0627    CB FB                  SET    7,E
0629    CB FC                  SET    7,H
062B    CB FD                  SET    7,L


062D    CB 26                  SLA    (HL)
062F    DD CB 36 26            SLA    (IX+INDEX)
0633    FD CB 36 26            SLA    (IY+INDEX)
0637    CB 27                  SLA    A
0639    CB 20                  SLA    B
063B    CB 21                  SLA    C
063D    CB 22                  SLA    D
063F    CB 23                  SLA    E
0641    CB 24                  SLA    H
0643    CB 25                  SLA    L


0645    CB 2E                  SRA    (HL)
0647    DD CB 36 2E            SRA    (IX+INDEX)
064B    FD CB 36 2E            SRA    (IY+INDEX)
064F    CB 2F                  SRA    A
0651    CB 28                  SRA    B
0653    CB 29                  SRA    C
0655    CB 2A                  SRA    D
0657    CB 2B                  SRA    E
0659    CB 2C                  SRA    H
065B    CB 2D                  SRA    L


065D    CB 3E                  SRL    (HL)
065F    DD CB 36 3E            SRL    (IX+INDEX)
0663    FD CB 36 3E            SRL    (IY+INDEX)
0667    CB 3F                  SRL    A
0669    CB 38                  SRL    B
066B    CB 39                  SRL    C
066D    CB 3A                  SRL    D
066F    CB 3B                  SRL    E
0671    CB 3C                  SRL    H
0673    CB 3D                  SRL    L


0675    96                     SUB    (HL)
0676    DD 96 36               SUB    (IX+INDEX)
0679    FD 96 36               SUB    (IY+INDEX)
067C    97                     SUB    A
067D    90                     SUB    B
067E    91                     SUB    C
067F    92                     SUB    D
0680    93                     SUB    E
0681    94                     SUB    H
0682    95                     SUB    L
0683    D6 49                  SUB    NN
```

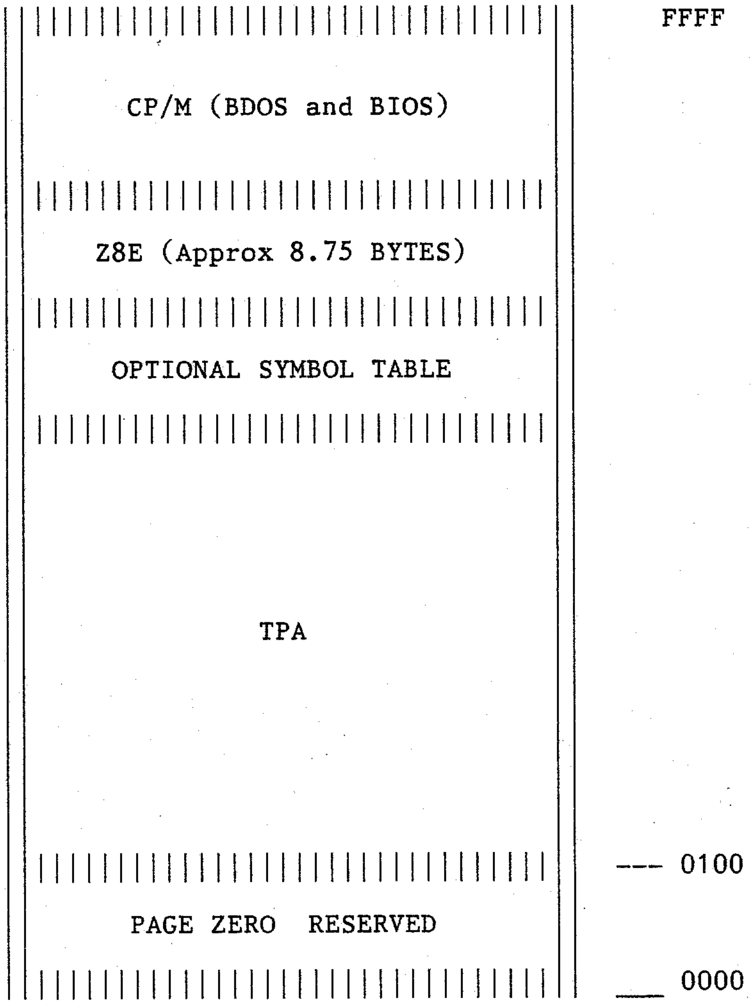
APPENDIX B - ZILOG MNEMONICS

```
0685     AE                   XOR    (HL)
0686     DD AE 36             XOR    (IX+INDEX)
0689     FD AE 36             XOR    (IY+INDEX)
068C     AF                   XOR    A
068D     A8                   XOR    B
068E     A9                   XOR    C
068F     AA                   XOR    D
0690     AB                   XOR    E
0691     AC                   XOR    H
0692     AD                   XOR    L
0693     EE 49                XOR    NN
```

APPENDIX B - ZILOG MNEMONICS

APPENDIX B - ZILOG MNEMONICS

APPENDIX C - SYSTEM MEMORY MAP



APPENDIX B - ZILOG MNEMONICS

COMMAND SUMMARY REFERENCE

CMD	Description	Arguments		
A	Inline Assembly	StartAddr		
B	Set Breakpoint	Addr1[,Pass Count]	[Addr2..AddrN]	
C	Clear Breakpoint	Addr1	[Addr2..AddrN]	
D	Dump Memory	[StartAddr]	[End/Count]	
E	Examine Memory	StartAddr		
F	Find	StartAddr	MatchData	
G	Go	ExecutionAddr		
H	Display Symbol Table	[FirstSymbol]		
I	Input File	FileName	[,Load Address]	
J	Full Screen/Animated Debug	[/] [*] [Addr]	[Timeout]	
K	Set Memory Window	StartAddr	[Size]	
M	Move Memory	SourceStart	SourceEnd	DestStart
N	Output to Port NO Pre-Read	[(] PortAddr [)]		
O	Output Current Breakpoints			
P	Exam/Modify PSW (Flag Reg)			
Q	Query I/O Port	[(] PortAddr [)]		
R	Examine/Modify Registers	RegSpecifier		
S	Single-Step	[/] [Count]		
U	Write Symbol Table To Disk	FileName		
V	Verify Memory	SourceStart	SourceEnd	DestStart
W	Write to Disk	FileName	[StartAddr]	[EndAddr]
X	Examine Machine State			
Y	Fill Memory	FromAddr	ToAddr	Data
Z	Disassemble	StartAddr	End/Count	FileName

[]	Denotes Optional Argument
[/]	Do Not Trace Subroutine
[*]	Do Not Trace BDOS Call
[(] [)]	I/O Port Monitor Mode