==

Z 8 0 D I S

USER

MANUAL

__

A new Z80 disassembler

by

KENNETH GIELOW

28 JULY 85

==

- SECTION 1 -
INTRODUCTORY REMARKS

Z80DIS is an entirely new disassembler for Z80 based CP/M systems. Z80DIS is written in TURBO PASCAL. Z80DIS generates Z80 mnemonics and prepares an assembly language file with many special features for ease of understanding the intent of the code. The program and documantation are Copyright 1985, by Kenneth Gielow, Palo Alto, CA. All rights are reserved.

This user manual is written in 9 sections:

At this time, in order to retain creative control of the growth of the program, I do not intend any general release of the 3000-plus lines of source code for the disassembler.

The following files will be found on this program distribution library package:

```
Z80DIS.COM    - the program you will be installing
Z80DIS.000    - optional overlay files for Z80DIS  (may be
Z80DIS.001    -  one, several, or none depending on version)

ZDINSTAL.COM  - the installation program
ZDINSTAL.MSG  - contains text of ZDINSTAL messages
ZDINSTAL.DTA  - contains terminal characteristics data

Z80DIS.DOC    - the user manual in ready to print format
```

The Z80DIS binary is in three files (see SECTION 3 for use.) The ZDINSTAL files will tailor the interactive displays to your terminal (see SECTION 4 for use.) The DOC file is the text for this user manual.

The distributed version is compiled for a standard somewhat small CP/M system for maximum compatibility with the larger CP/M 3.0 and for users of the ZCPR3 extensions to CP/M. It is an unfortunate quirk of TURBO PASCAL that compilations are for a specific sized machine; TPA memory size has been set for 50K; the top address is C7FF hex. This allows enough heap space for about 1125 label references. See SECTION 4 for an exposition of the files included in this distribution and on interactively customizing the Z80DIS.COM binary file for your terminal configuration.

Please address comments and suggestions to
79 Tulip Lane
Palo Alto, CA 94303

Leave messages for 'Kenneth Gielow' on the following boards:

POTPOURRI RCP/M	(408) 378-7474
METAL MESSAGE SYSTEM #1	(415) 965-4090

- SECTION 2 -
FEATURES OF Z80DIS

A major feature of this program is an extensive memory map listing. This takes on two forms:

The first is an address listing which distinguishes types of label references, marks subroutines and jumps and produces an annotated cross-reference of all labels referenced. The type of each reference is shown by a one or two letter code:

J = Jump	Jr = Jump relative
C = Call	Cr = Call by Restart
Lw = Load word	Lb = Load byte
Sw = Store word	Sb = Store byte
Iw = Immediate word	

The other form of address mapping is an address labeling convention which assists greatly in understanding the assembly code generated. Each label generated on the assembly listing indicates, by format, both the generic type of any references and the singularity of the reference:

Where a CP/M address can be assumed, then that name is used:

0005h=>BDOS, 005Ch=>FCB1, 006Dh=>FCB2, 0080h=>DBUF, etc.

Otherwise, the label takes the form L.hhhh and consists of three parts:

The first character is usually a "J", "C", "D", "I" or "X".
- J = JUMP (JP, JR references only)
- C = CALL (CALL, RST, JP, JR only)
- D = DATA (8- or 16-bit Load, Store or Immediate only)
- I = Immediate 16-bit only
- X = any other combination (a suspicious mixture)

If the first character is a "." (period), the address is not referenced, but is included because it is the beginning of an inaccessible instruction code segment.

The second character indicates the singularity of the label.
- . = multiple references
- # = singular reference

The third through sixth characters are the hex address.

For example, J#02E3 would be used at address 02E3 if one reference was made to that address and that reference was a JUMP.

Another feature of this disassembler is the style of presentation of the assembly code. This code is output in two forms: The **.PRN file and the **.MAC file:

The **.MAC file is your normal assembly language output file ready to be modified and re-assembled.

The **.PRN file is an aid to understanding the intent of the code. The file looks like the output of an assembler with both the instructions and assembled bytes shown. The juxtaposition of the hex bytes and the assembly mnemonics allows the user to recognize errors caused by an incorrectly defined break table.

A blank comment line is inserted after every CALL or CONDITIONAL JUMP. A bar-of-dashes comment line is inserted after every UNCONDITIONAL JUMP. All CALL instructions are indented one space to highlight them. All subroutines are marked with a 5-line header comment which separates them from the other code and allows you to annotate the purpose of the subroutine.

- SECTION 3 -
INSTRUCTIONS FOR EXECUTION

Z80DIS is a batch program with user interactive set-up of all of the control parameters. You start Z80DIS by typing "Z80DIS" with no parameters. The program will prompt you for inputs. In most cases, default entries are shown in DIM intensity already occupying the data field. If you type only the RETURN key, the default will be used.

Z80DIS.COM may be on the drive of your choice and executed via a drive prefix (e.i. B:Z80DIS), but the overlay files, Z80DIS.001 and Z80DIS.002, must be on your logged drive.

There are four phases of execution. The first two are user interactive: Basic parameter set-up and definition of the break table. The second two are automatic: Disassembly pass 1 and disassembly pass 2. These will be dealt with in section 3.1 through 3.4 which follow. Section 3.5 will deal with limitations and run-time errors.

--

SECTION 3.1 -- BASIC PARAMETER DEFINITION: After a brief sign-on message, you will be asked to supply the following information:

```
|==================================================================
| Please enter  INPUT   file name: ______________
|               OUTPUT  file name: ______________
|               LISTING file name: ______________
|   Descriptive TITLE: __________________________________________
|
|              file LOAD  address: ____ HEX
|       disassembly START address: ____ HEX
|       disassembly STOP  address: ____ HEX
|
| Do you wish to run a FULL output (as opposed to XREF only)?
| On which disk do you wish the scratch file to reside? (A-G)
|
| Do You wish to process all Z80 codes (as opposed 8080 subset)?
|
| Are all inputs OK so far?
|==================================================================
```

Each question is presented one-at-a-time from the top. The paragraphs below discuss the purpose, defaults, editing capability and error checking for each answer.

INPUT file name: _____________

Z80DIS wants the name of the file you wish to disassemble. This file must be in the absolute binary format common to CP/M ".COM" files. The file need not be a .COM file and need not load into memory at address 0100 hex. Z80DIS will ask for the load address at a later step.

The program wants a CP/M style file name in the standard form D:FILENAME.EXT. If a disk is not specified, the current drive is used. If an extension is not specified, ".COM" is used. The program will test for existence of the file and will wait until you enter a valid name before proceeding. If you wish a null extension, enter the file name with a final period as "XXXX.".

Until the <CR> terminating the name is typed, you may correct the entry as required by backspacing. You may terminate Z80DIS by the usual ^C.

OUTPUT file name: _____________

Z80DIS wants the name of the file to be used for the disassembled output.

The program provides a default file name which is the same as the file name given for INPUT above with the extension changed to ".MAC". If you enter a <CR> you will get that default name. If you enter a file name without an extension, the default extension ".MAC" will be appended. If you wish a null extension, enter the file name with a final period as "XXXX.".

Until the <CR> terminating the name is typed, you may correct the entry as required by backspacing. You may terminate Z80DIS by the usual ^C.

The program will wait until you enter a valid name before proceeding. The program checks to see that the OUTPUT file name is not the same as the INPUT file name. If there is already a file by the same name, that file is overwritten without comment.

LISTING file name: _____________

Z80DIS wants the name of the file to be used for the listing of the cross-reference and the disassembly.

The program provides a default file name which is the same as the file name given for OUTPUT above with the extension changed to ".PRN". If you enter a <CR> you will get that default name. If you enter a file name without an extension, the default extension ".PRN" will be appended. If you wish a null extension, enter the file name with a final period as "XXXX.".

Until the <CR> terminating the name is typed, you may correct the entry as required by backspacing. You may terminate Z80DIS by the usual ^C.

The program will wait until you enter a valid name before proceeding. The program checks to see that the LISTING file name is not the same as either of the previous file names. If there is already a file by the same name, that file is overwritten without comment.

Descriptive TITLE: ____

You may enter a free-form, 42-character title to be included as identification on your listings and output code. Some form of date, time, cracking attempt number are useful.

Until the <CR> terminating the name is typed, you may correct the entry as required by backspacing. You may terminate Z80DIS by the usual ^C.

----

file LOAD address: ____ HEX

Z80DIS wants the hex memory address at which the first word of the file resides when loaded and ready to run. This is usually 0100 hex for normal .COM files. If you enter only <CR>, you will get the 0100 default.

If you have created the file in some other manner such as a dump of ROM contents to a file, then you will need to give the ROM starting address. Some programs relocate part of the binary image after loading but before that part is executed; this relocation is usually to higher memory. To compensate for relocation, you will have to give an adjusted load address for the first word of the file (higher than 0100 by the relocation amount.)

Until the <CR> terminating the name is typed, you may correct the entry as required by backspacing. You may terminate Z80DIS by the usual ^C.

----

disassembly START address: ____ HEX

Z80DIS wants the hex memory address at which to start the disassembly. This is usually the same as the LOAD address. If you enter only <CR>, you will get the value you entered as LOAD address as default. The START address must be at or higher than the LOAD address.

If you wish only a partial disassembly, you may specify a more limited range of START and STOP addresses.

Editing is the same as for the LOAD address above.

----
disassembly STOP address: ____ HEX

Z80DIS wants the hex memory address at which to stop the disassembly. If you enter only <CR>, you will get the value FFFF hex which means the end of the INPUT file. The STOP address must be higher than the START address.

If you wish only a partial disassembly, you may specify a more limited range of START and STOP addresses.

Until the <CR> terminating the name is typed, you may correct the entry as required by backspacing. You may terminate Z80DIS by the usual ^C.

Do you wish to run a FULL output
      (as opposed to XREF only) ? (Y/N) _

If you answer Y, you will get a full disassembly including both the cross-reference part and the instruction code part. If you answer N, you will only get a cross-reference.

The "N" answer processes faster and is a good place to start to get a feel for the memory layout and to develop a break table.

The "Y" answer requires use of a scratch file to hold the disassembled code and other information prior to the completion of the address reference collection. This file can be rather large (30 bytes per instruction) and should reside on a drive with plenty of space. If you have answered "Y", you will be asked the following question:

On which disk do you wish the scratch file to reside? (A-G) _

This allows you to place the scratch file on a specific drive. If you answer <CR>, then the logged drive is used.

Do you wish to process all Z80 codes
      (as opposed 8080 subset only) ? (Y/N) _

Many of the public domain, and other, programs which run under CP/M are written in pure 8080 code without the Z80 extensions to the instruction set. Such programs should be disassembled with a "N" answer to this question. The NO answer will still generate Z80 mnemonics, but will mark as invalid all instruction bytes corresponding to the unexpected Z80 extensions to the 8080 code set.

You should use the "Y" answer only if you know that the program will run only on a Z80 based CP/M system.

```
Are all inputs OK so far? (Y/N) _
```

This question gives you a chance to re-examine your answers to the other questions before continuing. If you answer "N", the program will return to the INPUT file question. If you answer "Y", you will continue on to setting up your break table.

That completes the set-up phase of the disassembly process. The next step is to define the type of disassembly to be used for each part of the program memory.

--

SECTION 3.2 -- DEFINITION OF THE BREAK TABLE: After you have answered YES to the "Inputs OK" question, Your terminal will now display the following:

```
|================================================================
| >>>  Z80DIS version 1.5
| You may now enter CONTROL BREAK addresses to define the type
|  of disassembly for each section of the code; each control
|  break defines the first address of a section which ends at the
|  beginning of the next section-1 byte.
|
| TO SEE DETAILS OF YOUR COMMAND CHOICES, type H
|
| ?: _
|================================================================
```

If this is your first attempt to disassemble a new program and you have no idea where anything is stored, just skip this phase by entering "Q". The "Q" command will QUIT the break table definition process and treat the whole program memory as instructions. The memory map shown by the cross-reference printout will give you a good estimate of the break table contents for your second attempt to crack the code.

A "break" is defined to be an address in the memory where you wish to either force a break, or you wish to change from one type of disassembly to another. The break table is defined to be a list of the break addresses for a given binary file. The break table may be written to a break file (See SECTION 8.)

This disassembler can operate in six modes:

```
     I = Instructions
     A = Ascii text using DEFM with quoted strings
     B = Byte storage using DEFB with hex byte values
     W = Word storage using DEFW with hex word values
     T = Table of addresses using DEFW with address labels
     S = Space using DEFS and the length
```

Each break address defines the start of that mode. The mode remains in effect until one byte before the next break address or the end of the program memory.

In addition to the six break setting commands, there are eight other commands that may be entered:

```
     C  = Clear all of break table
     FL = Load (append) a break file to the break table
     FS = Store break table as a break file
     L  = List break table for review
     H  = display Help for break table editing commands
     K  = Kill existing break at an address
     P  = Print break table on "LST:" CP/M device
     Q  = Quit break entry process (start the disassembly)
```

THE COMMANDS

Commands may be given in any order with the exception of "Q" which terminates command entry. Each command is presented below to discuss the purpose, editing capability, and error detection.

C

Purpose: Clears entire break table to empty. An empty break table defines the entire code segment to be instructions. As this is a fairly drastic thing to do, you will be asked to respond to the following question ..

```
Do you really want to clear all control breaks? (Y/N) _
```

If you answer N, the command is canceled. If you answer Y, the table is cleared. All other entries but N and Y are ignored.

FS

Purpose: Store break table as a break file (see SECTION 8.)

Response: The program responds by filling out the command line as shown below:

```
FS - Save Control Breaks..      SAVE file name: ___________
```

You must give a legal CP/M file name followed by carriage return <CR>. If you just give a <CR>, the default name of "SAVE.BRK" will be used. If you enter only the name part and not an extension, the extension ".BRK" will be used.

Upon entry of the <CR> the file is written.

FL

Purpose: Load break table from a break file (see SECTION 8.)

Response: The program responds by filling out the command line as shown below:

```
FL - Load Control Breaks..      SAVE file name: ___________
```

You must give a legal CP/M file name followed by carriage return <CR>. If you just give a <CR>, the default name of "SAVE.BRK" will be used. If you enter only the name part and not an extension, the extension ".BRK" will be used.

Upon entry of the <CR> the file is read and merged with the current break table. The following error messages may be given:

```
          < ERROR > that file does not exist
```

The file you have requested is not present; the FL command is canceled.

```
          < ERROR > type "X" is invalid from SAVE file.. IGNORED
```

The break type shown as X was found on your break file; X is not a legal break type. (Legal types are I,A,B,W,T,S) That particular break point is ignored but file processing continues.

```
          < ERROR > Invalid Hex Address "XXXX" from SAVE file
```

The break address shown as XXXX was found on your break file; XXXX is not a valid hexadecimal address. That particular break point is ignored but file processing continues.

L

Purpose: List the break table to your console display device.

Response: The program responds by filling out the command line followed by a listing of the break table as shown below:

```
List Control Breaks

  Typ 0000-0000  Typ 0000-0000  Typ 0000-0000  Typ 0000-0000 ....
```

H

Purpose: Request display of short crib notes on break table commands.

Response: the following screen of information is displayed:

```
|================================================================
|
| command   long name    break code    meaning
|---------/------------/------------/----------------------------
|    C      Clear            -        clear all of break table
|    FL     File Load        -        Load (append) file to table
|    FS     File Store       -        Store table as file
|    L      List             -        List break table for review
|    H      Help             -        Re-list these instructions
|    K      Kill             -        Kill break at address
|    P      Print            -        Print break table on LST:
|    Q      Quit             -        Quit break entry process
|
|    I      Instructions    Ins       set Instruction break
|    A      Ascii           Asc       set Ascii break
|    B      Bytes           Byt       set Byte break
|    W      Words           Wrd       set Word break
|    T      Table of addr   Adt       set address Table break
|    S      Space           Spc       set Space break
|
|================================================================
```

K

Purpose: Remove a break point from the table.

Response: The program will complete the command and position the cursor for entry of a Hexadecimal number:

```
Kill break at .... hex
```

You enter the address followed by <CR>. Any number that you enter will be right justified in the four digits when you push <CR>. If the address is not that of an existing break point, no error message is given and no action is taken. If the address is out of range of the program addresses, the following informative message is given:

```
              < ERROR: address out of range, ignored >
```

P

Purpose: List the break table to your CP/M LST: device.

Response: The program responds by filling out the command line on the display followed by a listing of the break table on the printer as shown below:

```
                       < to the display >

Print Control Breaks

                       < to the printer >

  Typ 0000-0000  Typ 0000-0000  Typ 0000-0000  Typ 0000-0000 ....
```

Q

Purpose: Quit break table processing and proceed with disassembly.

Response: The program will complete the command line in one of two ways. If you have saved your definitions onto a file for future use, or have not made any changes to definitions loaded from a file, you will get the normal message and control will pass to the disassembler pass one:

```
Quit Control Break definition -
```

Alternate Response: If you have not saved your break table, you will get the following warning message:

```
Quit < Informative ERROR >
    attempt to QUIT without saving Break Addresses
      either SAVE to a file    (using FS command),
          or QUIT without save (using Q  command)
```

If you do not wish to save the table, just type "Q" again.

I
A
B
W
T
S

Purpose: These six commands set break table addresses to define transition points in the disassembly mode.

Response: The program will complete the command line and position the cursor for entry of a Hexadecimal number:

```
I at .... hex
```

You enter the address followed by <CR>. Any number that you enter will be right justified in the four digits when you push <CR>. If the address is out of range of the program addresses, one of the following informative message is given:

```
          < ERROR: address below start, set to start >
          < ERROR: address above stop, ignored >
```

Each break address defines the start of that mode. The disassembly mode defined by that break address remains in effect until one byte before the next break address or the end of the program memory. If you enter a break at the same address as an existing break address, the former definition is overwritten without comment.

This disassembler can operate in six modes:

I = Instructions
(Either Z80 full set or 8080 sub set)

A = Ascii text using DEFM with quoted strings
(an Ascii region is output as strings of characters)

B = Byte storage using DEFB with hex byte values
(blocks of single bytes interpreted as numbers)

W = Word storage using DEFW with hex word values
(blocks of double byte words interpreted as numbers)

T = Table of addresses using DEFW with address labels
(blocks of byte pairs interpreted as addresses)

S = Space using DEFS and the length
(blocks of empty space of uninterpreted content)

_   (any other command not listed above)

Purpose: If you can't think of any commands, enter anything.

Response: The program will respond with the following short crib note. as it says, to see more helpful information, enter "H".

```
_ is unacceptable, commands are: C,F,L,H,K,P,Q;
                Break Types are: I,A,B,W,T,S
   To see complete definitions, enter H.
```

Upon entry of the Quit command, the actual disassembly will begin.

--

SECTION 3.3 -- EXECUTION PASS ONE: During this pass the code is cracked according to the break table, the cross reference list structure is linked into the PASCAL heap space in upper memory, the cracked code is copied with context information onto a scratch file called Z80DIS.$$$. During pass one, your console screen will display the following:

```
|================================================================
| Beginning disassembly...
| THIS IS Pass 1
| working at 05F0 Asc
|================================================================
```

The "working at" line is animated and will show the current address and disassembly mode. In the illustrated case the program has just finished processing the file D.COM and the last break type was ascii.

During any execution phase, you may type ^C to abort that phase. If you abort during pass one, the program will treat that as a premature end of file and go on to process what it has through pass two.

--

SECTION 3.4 -- EXECUTION PASS TWO:

During this pass all user output files and listings are produced and the scratch file is erased. During pass two, your console screen will display the following:

```
|=================================================================
| THIS IS Pass 2
| scratch file contains 396 records of 30 bytes each
| Free memory space remaining after XREF table storage
|   assignments is 29144 bytes out of the original 31300
|   bytes (or   6.9 percent used.)
|
| LISTING cross references
|
| LISTING Subroutines
|
| PRODUCING disassembled output files
|   Processing external label equates
|   Working at 05F0 Asc
|=================================================================
```

The display will take you step by step through pass two and show you what the program is doing at each step.

The informative messages about file and memory space will let you see just how close to capacity the program is running.

The "working at" line is animated and will show the current address and disassembly mode. In the illustrated case the program has just finished pass two processing of the file D.COM.

During any execution phase, you may type ^C to abort that phase. If you abort during any of the steps of pass two, the program will go on to process the next step.

When the program is complete, Z80DIS will ring your terminal bell twice and display the sign-off message..

```
|=================================================================
| END of Pass 2
|=================================================================
```

--

SECTION 3.5 -- LIMITATIONS AND RUN-TIME ERRORS: There are several errors which TURBO PASCAL can produce owing to limitations on memory or file size. This section will discuss those errors and the implications.

The following overflow conditions can cause trouble in running a large disassembly:

1) You may overflow the program memory space used to hold the address references.

2) You may overflow Z80DIS.$$$ scratch file space in pass one.

3) You may overflow either **.PRN or **.MAC file space in pass two.

None of these errors is currently trapped and handled in a friendly manner. Each generates a TURBO PASCAL error. TURBO distinguished between "Runtime" errors ans "I/O" errors. The error codes which can be expected are as follows:

RUN-TIME ERRORS

F0 - Overlay file not found

The Z80DIS overlay files Z80DIS.001 and .002 must be on your logged drive. Keep all of Z80DIS on your logged drive and refer to your data files with drive prefixes if necessary (e.g. B:D.COM)

FF - Heap/Stack collision

You have too many labels to references to labels for the memory space allowed. Each label definition takes 9 bytes, each label reference takes 5 bytes, each break table address takes 5 bytes. The allocation for this version of Z80DIS is roughly 24000 bytes. That is about 1125 labels at two references per label.

I/O ERRORS

F0 - Disk Write error

You disk is full. If this occurs in pass one, the disk with the scratch file is full. You should move the scratch file to a disk with more space; this is done during initial setup question and answer session when the program asks for the residence disk of the scratch file.

If this occurs during pass two, then either the **.PRN file or the **.MAC file has overflowed disk space (the .PRN file is considerably larger than the .MAC file.) This will normally not happen in a run for cross-reference only as the .PRN file is much smaller and the .MAC file is not produced. The only cures for this problem are to disassemble by parts or to move these files to a drive with more space (i.e. use a drive prefix when answering the question "OUTPUT file: _____")

F1 - Directory full

One of your disk directories is full. If this occurs during pass one, it is the disk with Z80DIS.$$$. If this occurs during pass two, then either the **.PRN file or the **.MAC file are involved. You must erase a couple of files or use a disk with space remaining in the directory.

No other TURBO errors are expected. Most all type in formats and file existence problems are caught by the program and you are given appropriate helpful suggestions.

- SECTION 4 -
ADAPTATION OF Z80DIS TO YOUR TERMINAL

CP/M is a generic operating system, but most of the terminals used with CP/M have features not anticipated by CP/M. These features include CURSOR ADDRESSing and BRIGHT/DIM display of characters. This program utilizes such features to improve the operator interaction. As the program itself is written in TURBO PASCAL, the TURBO installation support feature has been used to make this adaptation easy.

Therefore, before you use this program, it must be installed to your particular terminal, by providing it with information regarding control characters required for certain functions. This installation is easily performed using the program ZDINSTAL which is described in this section.

The following files are part of this program installation package and must be present during terminal installation (the three ZDINSTAL.* files may then be deleted after installation if no other terminals are to be supported)..

```
     Z80DIS.COM
                    - the program you will be installing
     Z80DIS.000
                    - optional overlay files for Z80DIS  (may be
     Z80DIS.001
                        one, several, or none depending on version)

     ZDINSTAL.COM
                    - the installation program
     ZDINSTAL.MSG
                    - contains text of ZDINSTAL messages
     ZDINSTAL.DTA
                    - contains terminal characteristics data
```

- THE INSTALLATION PROCEDURE -

Start the installation by typing ZDINSTAL at your terminal. Select Screen installation from the main menu.

A numbered menu listing a number of popular terminals will appear, inviting you to choose one by entering its number.

If your terminal is mentioned, just enter the corresponding number, and the installation is complete. Before installation is actually performed, you are asked the question..

```
Do you want to modify the definition before installation? _
```

This allows you to modify one or more of the values being installed as described in the following. If you do not want to modify the terminal definition, just type N, and the installation will complete by asking you the operating frequency of your CPU to establish parameters for timing loops.

If your terminal is not on the menu, however, you must define the required values yourself. The values can most probably be found in the manual supplied with your terminal.

Enter the number corresponding to None of the above and answer the questions one by one as they appear on the screen.

In the following, each command you may install is described in detail. Your terminal may not support all the commands that can be installed. If so, just pass the command not needed by typing RETURN in response to the prompt. If Delete line, Insert line, or Erase to end of line is not installed, these functions will be emulated in software, slowing screen performance somewhat.

Commands may be entered either simply by pressing the appropriate keys or by entering the decimal or hexadecimal ASCII value of the command. If a command requires the two characters 'ESCAPE' and '=', may...

either

Press first the Esc key, then the =. The entry will be echoed with appropriate labels, i.e. <ESC> =.

or

Enter the decimal or hexadecimal values separated by spaces. Hexadecimal values must be preceded by a dollar-sign. Enter e.g. 27 61 or $1B 61 or $1B $3D which are all equivalent.

The two methods cannot be mixed, i.e. once you have entered a non-numeric character, the rest of that command must be defined in that mode, and vise versa.

A hyphen entered as the very first character is used to delete a command, and echoes the text Nothing.

Terminal type: _

Enter the name of the terminal you are about to install. When you complete ZDINSTAL, the values will be stored, and the terminal name will appear on the initial list of terminals. If you later need to re-install Z80DIS to this terminal, you can do that by choosing it from the list.

Send an initialization string to the terminal? _

If you want to initialize your terminal when Z80DIS starts (e.g. to down load commands to programmable function keys), you answer Y for yes to this question. If not, just hit RETURN.

If you answer Y, you may choose between entering the command directly or defining a file name containing the command string.

The latter is a good idea if the initialization string is long, as e.g. a string to program a number of function keys would be.

Send a reset string to the terminal? _

Here, you may define a string to be sent to the terminal when Z80DIS terminates. The description of the initialization command above applies here.

CURSOR LEAD-IN command: _

Cursor Lead-in is a special sequence of characters which tells your terminal that the following characters are an address on the screen on which the cursor should be placed.

When you define this command, you are asked the following supplementary questions..

CURSOR POSITIONING COMMAND to send between line and column: _

Some terminals need a command between the two numbers defining the row and column cursor address.

CURSOR POSITIONING COMMAND to send after line and column: _

Some terminals need a command after the two numbers defining the row and column cursor address.

Column first? _

Most terminals require the address on the format: first ROW, then COLUMN. If this is the case on your terminal, answer N. If your terminal wants COLUMN first, then ROW, then answer Y.

OFFSET to add to LINE _

Enter the number to add to the LINE (ROW) address.

OFFSET to add to COLUMN _

Enter the number to add to the COLUMN address.

Binary address? _

Most terminals need the cursor address sent on binary form. If that is true for your terminal, enter Y. If your terminal expects the cursor address as ASCII digits, enter N. If so, you are asked the supplementary question..

----

2 or 3 ASCII digits? _

Enter the number of digits in the cursor address for your terminal.

CLEAR SCREEN command: _

Enter the command that will clear the entire contents of your screen, both foreground and background, if applicable.

Does CLEAR SCREEN also HOME cursor? _

This is normally the case; if it is not so on your terminal, enter N, and define the cursor HOME command.

DELETE LINE command: _

Enter the command that deletes the entire line at the cursor position.

INSERT LINE command: _

Enter the command that inserts a line at the cursor position.

ERASE TO END OF LINE command: _

Enter the command that erases the line at the cursor position from the cursor position through the right end of the line.

START OF 'LOW VIDEO' command: _

If your terminal supports different video intensities, then define the command that initiates the DIM video here. If this command is defined, the following question is asked . . .

START OF 'NORMAL VIDEO' command: _

Define the command that sets the screen to show characters in 'normal' video.

Number of rows (lines) on your screen: _

Enter the number of horizontal lines on your screen.

Number of columns on your screen: _

Enter the number of column positions on your screen.

Delay after CURSOR ADDRESS (0-255 ms): _
Delay after CLEAR, DELETE, and INSERT (0-255 ms): _
Delay after ERASE TO END OF LINE and HIGHLIGHT On/Off (0-255 ms):

Enter the delay in milliseconds required after the functions specified. RETURN means 0 (no delay).

Is this definition correct? _

If you have made any errors in the definitions, enter N. You will then return to the terminal selection menu. The installation data you have just entered will be included in the installation data file and appear on the terminal selection menu, but installation will not be performed.

When you enter Y in response to this question, you are asked..

Operating frequency of your microprocessor in MHz (for delays): _

As the delays specified earlier are depending on the operating frequency of your CPU, you must define this value.

The installation is finished, installation data is written to Z80DIS, and you return to the outer menu. New installation data is also saved in the installation data file and the new terminal will appear on the terminal selection list when you run ZDINSTAL in the future.

- SECTION 5 -
EXAMPLE OF FIRST ATTEMPT AT DEFAULT DISASSEMBLY

The following is part of the listing produced by running the disassembler against a copy of the public domain program D.COM without a break table. The break table, which specifies which addresses are to be treated as instructions and which are to be treated as various types of data, is defaulted to INS. Thus all bytes are treated as instructions. The break table is shown on the first page of the .PRN file. As shown below, the table consists of "Ins 0100-FFFE".

The CROSS REFERENCE listing, on the second page, is the first place to look. All referenced, and some unreferenced, addresses are shown with each referencing address and type of reference. Data references are clearly separated from instruction references; the labeling codes and the reference legend tell all. On this particular listing several blocks that are clearly data stand out; others may show up as the break table is expanded.

Unreferenced labels, such as those at 0435 through 045C, are generated only for INStruction mode disassembled sections of code. They will be created when there is no jump or call reference to an instruction following an unconditional jump instruction. Unreferenced labels should be viewed with suspicion; they may properly appear in a correct disassembly for only a very few reasons: 1) The programmer has included a standard set of subroutines, but does not use all of them. 2) You have done only a partial disassembly, and some other code not processed does reference the addresses.

Data references may be considered solid if both LOAD and STORE references show up. In this listing, there is clearly a one byte storage cell at D.02E6 as well as a block of both word and byte storage cells at D.0475 through D.0479. Apparently no instruction code exists past D.0C3D.

The REFERENCED SUBROUTINE list, on the page(s) following the cross reference list, is there to provide a place for you to annotate the purpose of each identified subroutine. To get onto this list, an address must be referenced by a CALL or RESTART instruction and must not be referenced as a data storage area. The apparent references to C.0038 as RESTART calls are suspicious as these are seldom used in CP/M programs; these references will turn out to be data mis-interpreted as instructions.

The last place to look is the DISASSEMBLED CODE printout. Here you look for << Illegal Op Code Byte >>. Upon examination, the illegal mix of instructions from 0103 through 0119 consists of a message string. The D program is interesting for the use of the CALL instruction to push in-line messages onto the stack. The mix of instructions from 0130 through 013A is another example of ascii data interpreted as instruction codes.

```
------------------ THE LISTING ------------------------------------------------

Demonstration without break table
the control parameter values used for this run of Z80DIS 1.5

   INPUT   file: D.COM
   OUTPUT  file: D.MAC
   LISTING file: D.PRN

          file LOAD  address 0100 hex
   disassembly START address 0100 hex
   disassembly STOP  address FFFF hex

   FULL DISASSEMBLY flag is set TRUE
        meaning.. Generate both XREF LIST and ASSEMBLY CODE output

   FULL Z80 flag is set to FALSE
        meaning.. Recognize only 8080 subset, but using Z80 mnemonics

   Control Breaks are set as follows:

  Ins 0100-FFFE
```

```
Demonstration without break table
B:D.MAC - Source by Z80DIS 1.5, K.Gielow, Palo Alto, CA.

  Listing of all referenced addresses

  LABELING CODES: Where a CP/M address can be assumed, then name is used.
      otherwise, prefix J=JUMP (Jump references only), C=CALL (Call/Jump only),
         I=Immediate only, D=DATA (Load/Store/Immed.), X= any other combination
      if reference is singular, The second character will be # instead of .

  REFERENCE LEGEND: J=Jump, Jr=Jump relative, C=Call, Cr=Call by Restart,
      Lw=Load word, Lb=Load byte, Sw=Store word, Sb=Store byte, Iw=Immed. word

I#0000   0000 ----I 0120/Iw
BDOS     0005 -C--- 011D/C, 01D6/C, 01E8/C, 01F3/C, 02FE/C, 037C/C,
                    038A/C, 03A1/C, 03AF/C, 03CB/C, 041A/C, 042D/C
I.000B   000B ----I 013C/Iw, 014F/Iw, 016D/Iw, 01A7/Iw, 0280/Iw, 02EB/Iw,
                    0332/Iw, 036E/Iw
C.0038   0038 -C--- 051E/Cr, 0555/Cr
FCB1     005C ---LI 01CB/Lb, 01F0/Iw, 0276/Lb, 0379/Iw, 0387/Iw, 03AC/Iw,
                    03C8/Iw
FCB1.1   005D ---LI 012A/Iw, 015B/Iw, 0195/Iw, 01BD/Lb, 034E/Iw, 035C/Iw,
                    044E/Iw, 045C/Iw
FCB2     006D ----I 014C/Iw, 032F/Iw
DBUF     0080 ----I 03B9/Iw
DBUF.1   0081 ----I 0205/Iw
I#0100   0100 ----I 0399/Iw
C#011A   011A -C--- 0100/C
C#013B   013B -C--- 012D/C
J#015B   015B J---- 0142/J
C#016C   016C -C--- 015E/C
J#0195   0195 J---- 0173/J
C#01A6   01A6 -C--- 0198/C
J#01BD   01BD J---- 01AD/J
J#01CB   01CB J---- 01C5/J
J#01D9   01D9 J---- 01CF/J
J#01E0   01E0 J---- 0265/J
J.01E2   01E2 J---- 023C/J, 0258/J, 02F5/J
J#01E4   01E4 J---- 01DD/J
J#021C   021C J---- 0240/J
J#0229   0229 J---- 0231/J
J#023F   023F J---- 022B/J
J.0243   0243 J---- 0215/J, 0221/J
J#0246   0246 J---- 024C/J
J#0268   0268 J---- 01F7/J
J#0283   0283 J---- 02B1/J
J.0285   0285 J---- 02A4/J, 02B5/J
J#0293   0293 J---- 0299/J
J#02B4   02B4 J---- 0287/J
C#02B8   02B8 -C--- 028E/C
J#02C5   02C5 J---- 02CC/J
I#02D1   02D1 ----I 02C2/Iw
D.02E6   02E6 --SL- 02B8/Lb, 02BE/Sb
J#02E7   02E7 J---- 020E/J
C.02F8   02F8 -C--- 0247/C, 0254/C, 025D/C, 0262/C, 0294/C, 02A0/C,
                    02A9/C, 02AE/C, 02C6/C
C.0305    0305 JC--- 0152/C, 02EE/C, 030C/J, 0371/C
C.0310    0310 JC--- 013F/C, 0170/C, 01AA/C, 0318/J, 0335/C, 0435/C
C.031C    031C -C--- 0148/C, 02E7/C, 0394/C
J#031F    031F J---- 0322/J
C.0327    0327 -C--- 0145/C, 0176/C
J.032A    032A J---- 033C/J, 043C/J
J.033F    033F J---- 0338/J, 0438/J
J.0344    0344 J---- 0349/J, 0449/J
C.034E    034E -C--- 01BA/C, 01C8/C
J.0355    0355 J---- 0358/J, 0458/J
J.035C    035C J---- 0158/J, 0179/J, 0427/J
C.036D    036D -C--- 035F/C, 045F/C
J#039C    039C J---- 03C0/J
J#03E1    03E1 J---- 03B6/J
J#03F4    03F4 J---- 03D2/J
J#0418    0418 J---- 01ED/J
J.0420    0420 J---- 026C/J, 0273/J, 027A/J, 028B/J
C.042A    042A -C--- 017C/C, 03D5/C, 03E1/C, 03F4/C
J.0430    0430 J---- 041D/J, 0424/J
D.0475    0475 --SLI 0124/Sw, 0127/Iw, 0430/Lw
D.0477    0477 --SL- 01B2/Sb, 020A/Lb, 0268/Lb, 0391/Sb, 0420/Lb
D.0478    0478 --SL- 01C2/Sb, 0211/Lb, 026F/Lb
D.0479    0479 --S-I 01B7/Sb, 0219/Iw, 027D/Iw, 031C/Iw, 0327/Iw
J#051C    051C J---- 05C4/J
J#054A    054A J---- 056A/J
J#056D    056D J---- 0560/J
J#05C4    05C4 J---- 0576/J
J#0618    0618 J---- 05EA/J
J#0621    0621 J---- 05DE/J
D#0A0D    0A0D ---L- 02E2/Lb
C#0A3F    0A3F -C--- 055B/C
C#0B05    0B05 -C--- 05DB/C
C.0B16    0B16 -C--- 0573/C, 05CD/C, 05D7/C, 05E7/C
D.0C3D    0C3D ---L- 057E/Lw, 058E/Lw, 0598/Lw, 05A9/Lw
D.0C47    0C47 --SLI 0570/Iw, 0579/Lw, 0593/Lw, 05B6/Lw, 05BA/Sw, 05D0/Iw,
                    05E4/Iw, 05FD/Lw
D.0C49    0C49 --SLI 0563/Lw, 0567/Sw, 056D/Iw, 0589/Lw, 05A4/Lw, 05BD/Lw,
                    05C1/Sw, 05CA/Iw
I#0C4B    0C4B ----I 05C7/Iw
I.0C4D    0C4D ----I 05D3/Iw, 05E1/Iw
D#0C4F    0C4F ---L- 0556/Lw
D.0C51    0C51 --SL- 0586/Sw, 05AE/Lw
D.0C53    0C53 --SL- 05ED/Lb, 05F1/Sb
I#0C54    0C54 ----I 05F7/Iw
I#2044    2044 ----I 04E6/Iw
I#2B21    2B21 ----I 0413/Iw
I#2F31    2F31 ----I 010F/Iw
                                              >> additional lines suppressed
```

```
Demonstration without break table
B:D.MAC - Source by Z80DIS 1.5, K.Gielow, Palo Alto, CA.

  Listing of all referenced SUBROUTINE entry points


BDOS    Subroutine ______________________________________________
         0005 -C--- 011D/C, 01D6/C, 01E8/C, 01F3/C, 02FE/C, 037C/C,
                    038A/C, 03A1/C, 03AF/C, 03CB/C, 041A/C, 042D/C

C.0038  Subroutine ______________________________________________
         0038 -C--- 051E/Cr, 0555/Cr

C#011A  Subroutine ______________________________________________
         011A -C--- 0100/C

C#013B  Subroutine ______________________________________________
         013B -C--- 012D/C

C#016C  Subroutine ______________________________________________
         016C -C--- 015E/C

C#01A6  Subroutine ______________________________________________
         01A6 -C--- 0198/C

C#02B8  Subroutine ______________________________________________
         02B8 -C--- 028E/C

C.02F8  Subroutine ______________________________________________
         02F8 -C--- 0247/C, 0254/C, 025D/C, 0262/C, 0294/C, 02A0/C,
                    02A9/C, 02AE/C, 02C6/C

C.0305  Subroutine ______________________________________________
         0305 JC--- 0152/C, 02EE/C, 030C/J, 0371/C

C.0310  Subroutine ______________________________________________
         0310 JC--- 013F/C, 0170/C, 01AA/C, 0318/J, 0335/C, 0435/C

C.031C  Subroutine ______________________________________________
         031C -C--- 0148/C, 02E7/C, 0394/C

C.0327  Subroutine ______________________________________________
         0327 -C--- 0145/C, 0176/C

C.034E  Subroutine ______________________________________________
         034E -C--- 01BA/C, 01C8/C

C.036D  Subroutine ______________________________________________
         036D -C--- 035F/C, 045F/C

C.042A  Subroutine ______________________________________________
         042A -C--- 017C/C, 03D5/C, 03E1/C, 03F4/C

                                  >> additional lines suppressed
```

```
                ;  Demonstration without break table
                ;  D.MAC - Source by Z80DIS 1.5, K.Gielow, Palo Alto, CA.
                ;
0100                    ORG     0100h
                ;
0000            I#0000: EQU     0000h   ----I
0005            BDOS:   EQU     0005h   -C---
000B            I.000B: EQU     000Bh   ----I
0038            C.0038: EQU     0038h   -C---
005C            FCB1:   EQU     005Ch   ---LI
005D            FCB1.1: EQU     005Dh   ---LI
006D            FCB2:   EQU     006Dh   ----I
0080            DBUF:   EQU     0080h   ----I
0081            DBUF.1: EQU     0081h   ----I
                ;
0100 CD1A01     I#0100:  CALL   C#011A
                ;
0103 44                 LD      B,H
0104 2E43               LD      L,43h   ;  "C"
0106 4F                 LD      C,A
0107 4D                 LD      C,L
0108 20                 DEFB    20h             ; << Illegal Op Code Byte >>
                ;       -----------------
                ;
0109 41                 LD      B,C
010A 53                 LD      D,E
010B 20                 DEFB    20h             ; << Illegal Op Code Byte >>
                ;       -----------------
                ;
010C 4F                 LD      C,A
010D 46                 LD      B,(HL)
010E 20                 DEFB    20h             ; << Illegal Op Code Byte >>
                ;       -----------------
                ;
010F 31312F             LD      SP,I#2F31
0112 32332F             LD      (D#2F33),A
0115 37                 SCF
0116 38                 DEFB    38h             ; << Illegal Op Code Byte >>
                ;       -----------------
                ;
0117 0D                 DEC     C
0118 0A                 LD      A,(BC)
0119 24                 INC     H
                ;
                ;         Subroutine ______________________________
                ;            Inputs  ___________________________
                ;           Outputs  ___________________________
                ;
011A D1         C#011A: POP     DE
011B 0E09               LD      C,09h   ;  9
011D CD0500              CALL   BDOS
                ;
0120 210000             LD      HL,I#0000
0123 39                 ADD     HL,SP
0124 227504             LD      (D.0475),HL
```

```
0127 317504              LD      SP,D.0475
012A 215D00              LD      HL,FCB1.1
012D CD3B01               CALL   C#013B
                ;
0130 41                  LD      B,C
0131 44                  LD      B,H
0132 44                  LD      B,H
0133 20                  DEFB    20h              ; << Illegal Op Code Byte >>
0134 20                  DEFB    20h              ; << Illegal Op Code Byte >>
0135 20                  DEFB    20h              ; << Illegal Op Code Byte >>
0136 20                  DEFB    20h              ; << Illegal Op Code Byte >>
0137 20                  DEFB    20h              ; << Illegal Op Code Byte >>
0138 20                  DEFB    20h              ; << Illegal Op Code Byte >>
0139 20                  DEFB    20h              ; << Illegal Op Code Byte >>
013A 20                  DEFB    20h              ; << Illegal Op Code Byte >>
                ;        -----------------
                ;
                ;
                ;          Subroutine ______________________________
                ;             Inputs  ___________________________
                ;             Outputs ___________________________
                ;
013B D1         C#013B: POP      DE
013C 010B00              LD      BC,I.000B
013F CD1003               CALL   C.0310
                ;
0142 C25B01              JP      NZ,J#015B
                ;
0145 CD2703               CALL   C.0327
                ;
0148 CD1C03               CALL   C.031C
                ;
014B EB                  EX      DE,HL
014C 216D00              LD      HL,FCB2
014F 010B00              LD      BC,I.000B
0152 CD0503               CALL   C.0305
                ;
0155 3EFF                LD      A,0FFh
0157 12                  LD      (DE),A
0158 C35C03              JP      J.035C
                ;
                ;        -----------------
015B 215D00     J#015B: LD       HL,FCB1.1
015E CD6C01               CALL   C#016C
                ;
0161 44                  LD      B,H
0162 45                  LD      B,L
0163 4C                  LD      C,H
0164 20                  DEFB    20h              ; << Illegal Op Code Byte >>
0165 20                  DEFB    20h              ; << Illegal Op Code Byte >>
0166 20                  DEFB    20h              ; << Illegal Op Code Byte >>
0167 20                  DEFB    20h              ; << Illegal Op Code Byte >>

                                          >> additional lines suppressed
```

- SECTION 6 -
EXAMPLE OF XXX.PRN FILE OUTPUT

The following is part of the listing produced by running the disassembler against a copy of the public domain program D.COM using a corrected break table. Here we have analyzed the previous listing and broken the code regions down into INStruction byte regions and WoRD, BYTe, and ASCii regions.

The resulting disassembly looks much better than the first attempt above in SECTION 5. Please refer back to those comments and listings to see the differences that the corrected break table has made. The unreferenced addresses are cleaned up; the references to C.0038 are resolved; the illegal op codes are gone.

The illegal mix of instructions from 0103 through 0119 now is resolved as a message string. The D program is interesting for the use of the CALL instruction to push in-line messages onto the stack. The CALL instruction at 0100 followed by the POP DE instruction at 013B is simply equivalent to LD DE,MESSAGE followed by a JP to 013C.

Note that this listing is un-edited; all comments, etc. are just as produced by the Z80DIS program.

```
---------------- THE LISTING -----------------------------------------------

Demonstration of disassembly of D.COM
the control parameter values used for this run of Z80DIS 1.5

   INPUT   file: D.COM
   OUTPUT  file: D.MAC
   LISTING file: D.PRN

          file LOAD  address 0100 hex
   disassembly START address 0100 hex
   disassembly STOP  address FFFF hex

   FULL DISASSEMBLY flag is set TRUE
        meaning.. Generate both XREF LIST and ASSEMBLY CODE output

   FULL Z80 flag is set to FALSE
        meaning.. Recognize only 8080 subset, but use Z80 mnemonics

   Control Breaks are set as follows:

  Ins 0100-0102  Asc 0103-0119  Ins 011A-012F  Asc 0130-013A  Ins 013B-0160
  Asc 0161-016B  Ins 016C-017E  Asc 017F-0194  Ins 0195-019A  Asc 019B-01A5
  Ins 01A6-02D0  Asc 02D1-02E6  Ins 02E7-0361  Asc 0362-036C  Ins 036D-03D7
  Asc 03D8-03E0  Ins 03E1-03E3  Asc 03E4-03F3  Ins 03F4-03F6  Asc 03F7-0417
  Ins 0418-0461  Asc 0462-0474  Wrd 0475-0476  Byt 0477-0478  Asc 0479-FFFE
```

```
Demonstration of disassembly of D.COM
B:D.MAC - Source by Z80DIS 1.5, K.Gielow, Palo Alto, CA.

  Listing of all referenced addresses

  LABELING CODES: Where a CP/M address can be assumed, then name is used.
      otherwise, prefix J=JUMP (Jump references only), C=CALL (Call/Jump only),
         I=Immediate only, D=DATA (Load/Store/Immed.), X= any other combination
      if reference is singular, The second character will be # instead of .

  REFERENCE LEGEND: J=Jump, Jr=Jump relative, C=Call, Cr=Call by Restart,
      Lw=Load word, Lb=Load byte, Sw=Store word, Sb=Store byte, Iw=Immed. word

I#0000   0000 ----I 0120/Iw
BDOS     0005 -C--- 011D/C, 01D6/C, 01E8/C, 01F3/C, 02FE/C, 037C/C,
                    038A/C, 03A1/C, 03AF/C, 03CB/C, 041A/C, 042D/C
I.000B   000B ----I 013C/Iw, 014F/Iw, 016D/Iw, 01A7/Iw, 0280/Iw, 02EB/Iw,
                    0332/Iw, 036E/Iw
FCB1     005C ---LI 01CB/Lb, 01F0/Iw, 0276/Lb, 0379/Iw, 0387/Iw, 03AC/Iw,
                    03C8/Iw
FCB1.1   005D ---LI 012A/Iw, 015B/Iw, 0195/Iw, 01BD/Lb, 034E/Iw, 035C/Iw,
                    044E/Iw, 045C/Iw
FCB2     006D ----I 014C/Iw, 032F/Iw
DBUF     0080 ----I 03B9/Iw
DBUF.1   0081 ----I 0205/Iw
I#0100   0100 ----I 0399/Iw
C#011A   011A -C--- 0100/C
C#013B   013B -C--- 012D/C
J#015B   015B J---- 0142/J
C#016C   016C -C--- 015E/C
J#0195   0195 J---- 0173/J
C#01A6   01A6 -C--- 0198/C
J#01BD   01BD J---- 01AD/J
J#01CB   01CB J---- 01C5/J
J#01D9   01D9 J---- 01CF/J
J#01E0   01E0 J---- 0265/J
J.01E2   01E2 J---- 023C/J, 0258/J, 02F5/J
J#01E4   01E4 J---- 01DD/J
J#021C   021C J---- 0240/J
J#0229   0229 J---- 0231/J
J#023F   023F J---- 022B/J
J.0243   0243 J---- 0215/J, 0221/J
J#0246   0246 J---- 024C/J
J#0268   0268 J---- 01F7/J
J#0283   0283 J---- 02B1/J
J.0285   0285 J---- 02A4/J, 02B5/J
J#0293   0293 J---- 0299/J
J#02B4   02B4 J---- 0287/J
C#02B8   02B8 -C--- 028E/C
J#02C5   02C5 J---- 02CC/J
I#02D1   02D1 ----I 02C2/Iw
D.02E6   02E6 --SL- 02B8/Lb, 02BE/Sb
J#02E7   02E7 J---- 020E/J
                                        >> additional lines suppressed
```

```
Demonstration of disassembly of D.COM
B:D.MAC - Source by Z80DIS 1.5, K.Gielow, Palo Alto, CA.

  Listing of all referenced SUBROUTINE entry points


BDOS     Subroutine ____________________________________________
          0005 -C--- 011D/C, 01D6/C, 01E8/C, 01F3/C, 02FE/C, 037C/C,
                     038A/C, 03A1/C, 03AF/C, 03CB/C, 041A/C, 042D/C

C#011A   Subroutine ____________________________________________
          011A -C--- 0100/C

C#013B   Subroutine ____________________________________________
          013B -C--- 012D/C

C#016C   Subroutine ____________________________________________
          016C -C--- 015E/C

C#01A6   Subroutine ____________________________________________
          01A6 -C--- 0198/C

C#02B8   Subroutine ____________________________________________
          02B8 -C--- 028E/C

C.02F8   Subroutine ____________________________________________
          02F8 -C--- 0247/C, 0254/C, 025D/C, 0262/C, 0294/C, 02A0/C,
                     02A9/C, 02AE/C, 02C6/C

C.0305   Subroutine ____________________________________________
          0305 JC--- 0152/C, 02EE/C, 030C/J, 0371/C

C.0310   Subroutine ____________________________________________
          0310 JC--- 013F/C, 0170/C, 01AA/C, 0318/J, 0335/C, 0435/C

C.031C   Subroutine ____________________________________________
          031C -C--- 0148/C, 02E7/C, 0394/C

C.0327   Subroutine ____________________________________________
          0327 -C--- 0145/C, 0176/C

C.034E   Subroutine ____________________________________________
          034E -C--- 01BA/C, 01C8/C

C.036D   Subroutine ____________________________________________
          036D -C--- 035F/C, 045F/C

C.042A   Subroutine ____________________________________________
          042A -C--- 017C/C, 03D5/C, 03E1/C, 03F4/C

                                             >> additional lines suppressed
```

```
                ;  Demonstration of disassembly of D.COM
                ;  D.MAC - Source by Z80DIS 1.5, K.Gielow, Palo Alto, CA.
                ;
0100                    ORG     0100h
                ;
0000            I#0000: EQU     0000h   ----I
0005            BDOS:   EQU     0005h   -C---
000B            I.000B: EQU     000Bh   ----I
005C            FCB1:   EQU     005Ch   ---LI
005D            FCB1.1: EQU     005Dh   ---LI
006D            FCB2:   EQU     006Dh   ----I
0080            DBUF:   EQU     0080h   ----I
0081            DBUF.1: EQU     0081h   ----I
                ;
0100 CD1A01     I#0100:  CALL   C#011A
                ;
0103 442E434F           DEFM    'D.COM AS OF 11/23/78',0Dh
0118 0A                 DEFB    0Ah
0119 24                 DEFM    '$'
                ;       -----------------
                ;
                ;
                ;         Subroutine ______________________________
                ;            Inputs  ___________________________
                ;            Outputs ___________________________
                ;
011A D1         C#011A: POP     DE
011B 0E09               LD      C,09h   ;  9
011D CD0500              CALL   BDOS
                ;
0120 210000             LD      HL,I#0000
0123 39                 ADD     HL,SP
0124 227504             LD      (D.0475),HL
0127 317504             LD      SP,D.0475
012A 215D00             LD      HL,FCB1.1
012D CD3B01              CALL   C#013B
                ;
0130 41444420           DEFM    'ADD        '
                ;       -----------------
                ;
                ;
                ;         Subroutine ______________________________
                ;            Inputs  ___________________________
                ;            Outputs ___________________________
                ;
013B D1         C#013B: POP     DE
013C 010B00             LD      BC,I.000B
013F CD1003              CALL   C.0310
                ;
0142 C25B01             JP      NZ,J#015B
                ;
0145 CD2703              CALL   C.0327
                ;
0148 CD1C03              CALL   C.031C
                ;
```

```
014B EB                 EX      DE,HL
014C 216D00             LD      HL,FCB2
014F 010B00             LD      BC,I.000B
0152 CD0503              CALL   C.0305
                ;
0155 3EFF               LD      A,0FFh
0157 12                 LD      (DE),A
0158 C35C03             JP      J.035C
                ;
                ;       -----------------
015B 215D00     J#015B: LD      HL,FCB1.1
015E CD6C01              CALL   C#016C
                ;
0161 44454C20           DEFM    'DEL        '
                ;       -----------------
                ;
                ;
                ;         Subroutine ______________________________
                ;            Inputs  ___________________________
                ;            Outputs ___________________________
                ;
016C D1         C#016C: POP     DE
016D 010B00             LD      BC,I.000B
0170 CD1003              CALL   C.0310
                ;
0173 C29501             JP      NZ,J#0195
                ;
0176 CD2703              CALL   C.0327
                ;
0179 D25C03             JP      NC,J.035C
                ;
017C CD2A04              CALL   C.042A
                ;
017F 2B2B4E41           DEFM    '++NAME NOT IN TABLE++$'
                ;       -----------------
                ;
0195 215D00     J#0195: LD      HL,FCB1.1
0198 CDA601              CALL   C#01A6
                ;
019B 53455420           DEFM    'SET        '
                ;       -----------------
                ;
                ;
                ;         Subroutine ______________________________
                ;            Inputs  ___________________________
                ;            Outputs ___________________________
                ;
01A6 D1         C#01A6: POP     DE
01A7 010B00             LD      BC,I.000B
01AA CD1003              CALL   C.0310
                ;

                                                >> additional lines suppressed
```

- SECTION 7 -
EXAMPLE OF XXX.MAC FILE OUTPUT

The assembly code file is in a format suitable for most assemblers. Various minor changes might have to be made before it will run through some, however (e.g. the colons ":" may have to be removed from labels.)

Fields, such as label and op-code, are separated by <tab>s rather than spaces for file compaction.

```
--------------- A LIST OF THE FILE CONTENTS ---------------------
;  Demonstration of disassembly of D.COM
;  D.MAC - Source by Z80DIS 1.5, K.Gielow, Palo Alto, CA.
;
        ORG     0100h
;
I#0000: EQU     0000h   ----I
BDOS:   EQU     0005h   -C---
I.000B: EQU     000Bh   ----I
FCB1:   EQU     005Ch   ---LI
FCB1.1: EQU     005Dh   ---LI
FCB2:   EQU     006Dh   ----I
DBUF:   EQU     0080h   ----I
DBUF.1: EQU     0081h   ----I
;
I#0100:  CALL   C#011A
;
        DEFM    'D.COM AS OF 11/23/78',0Dh
        DEFB    0Ah
        DEFM    '$'
;       -----------------
;
;
;         Subroutine ______________________________
;            Inputs  ___________________________
;            Outputs ___________________________
;
C#011A: POP     DE
        LD      C,09h   ;  9
         CALL   BDOS
;
        LD      HL,I#0000
        ADD     HL,SP
        LD      (D.0475),HL
        LD      SP,D.0475
        LD      HL,FCB1.1
         CALL   C#013B
;
        DEFM    'ADD        '
;       -----------------
;
;
```

```
;         Subroutine ______________________________
;            Inputs  ___________________________
;            Outputs ___________________________
;
C#013B: POP     DE
        LD      BC,I.000B
         CALL   C.0310
;
        JP      NZ,J#015B
;
         CALL   C.0327
;
         CALL   C.031C
;
        EX      DE,HL
        LD      HL,FCB2
        LD      BC,I.000B
         CALL   C.0305
;
        LD      A,0FFh
        LD      (DE),A
        JP      J.035C
;
;       -----------------
J#015B: LD      HL,FCB1.1
         CALL   C#016C
;
        DEFM    'DEL        '
;       -----------------
;
;
;         Subroutine ______________________________
;            Inputs  ___________________________
;            Outputs ___________________________
;
C#016C: POP     DE
        LD      BC,I.000B
         CALL   C.0310
;
        JP      NZ,J#0195
;
         CALL   C.0327
;
        JP      NC,J.035C
;
         CALL   C.042A
;
        DEFM    '++NAME NOT IN TABLE++$'
;       -----------------
;
J#0195: LD      HL,FCB1.1
         CALL   C#01A6
;
                                  >> additional lines suppressed
```

- SECTION 8 -
CONTENTS AND FORMAT OF *.BRK FILE

The break file is a transcription, to a CP/M file, of the contents of a break table. While the break table is stored internally as a linked list in PASCAL heap space, the break file is stored as a simple CP/M ascii file.

Each break address is a separate record. Records are terminated by the usual <CR> <LF> pair. Only the first five characters of each record are actually processed; any additional characters, if present, would be ignored.

The usual source of a break file is the FS command from within the Z80DIS program, but in fact the file could be separately generated or edited and still be read by the Z80DIS FL command.

Each record consists of two fields of characters: The first field is one upper or lower case character from the set A, B, I, S, T or W. The second field is four characters from the set 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E or F.

The first field specifies the break type (See SECTION 3.)

The second field specifies the hexadecimal address of that break. The span of influence of each break type is from the specified address through the address one byte less that the next highest break address; that is not necessarily the next break address read from the file.

Records produced by the Z80DIS, FS command will always be in ascending address order, but records read by the FL command may be in any order.

```
----- LISTING OF THE BREAK FILE FOR D.COM AS SHOWN HERE ---------
I0100
A0103
I011A
A0130
I013B
A0161
I016C
A017F
I0195
A019B
I01A6
A02D1
I02E7
A0362
I036D
A03D8
I03E1
A03E4
                                >> additional lines suppressed
```

- SECTION 9- POSSIBLE EXTENSIONS TO Z80DIS

Several extensions to the existing program appear to be either desirable or interesting. The inclusion of items on this list is not to be construed as indicating that they will ever be encorporated, but merely as an indication of interest.

This list will grow, and I invite a dialogue on the subject of the possibilities of expanding the utility of the program.

Notice that a disassembler that can show you the flow of a program can be very helpful in understanding your own assembly code as well as in understanding the code of other programmers.

--

1) An extension that I have been experimenting with is the automatic creation, or augmentation, of the break table by the program. This would optionally replace the code generating pass two with a break table generating pass two. The start-up options would become: Full disassembly, Cross reference only, Break table creation.

   The user would either enter his own best guess at the break table or would go with the default. Then the Automatic generation would be invoked and pass one would run to determine the cross references. The break determination pass would then run to examine the interdependency and reference types of the table just as you would. Areas which are clearly data would be isolated and the type of data would be determined by the reference (byte, word or immediate) and examination of the contents to recognize ascii. This additional break information would be added to that supplied by the user and the result would be output to a break file.

   This would not work for some partial disassemblies or for code that is relocated before execution. Spurious references caused by misinterpreting data as instructions can also influence the recognition of the true nature of other parts of the code.

   Still, the resulting table would be of great assistance to the user as a basis for the next disassembly trial. The automatic pass could be repeated with the new table; the changing of the disassembly mode of an area of memory from instruction to data will influence or eliminate spurious reference generation.

2) It could be useful to allow the user to specify one of a number of popular assembler input formats. This would influence the generation of colons following labels, use of <TAB>s to separate fields, hex notation, and others.

   At the present time this does not appear very useful as the code produced can be edited with global changes to correct tabs, labels, etc. In my use of a disassembler, I do not usually intend to fully reassemble a large program. I am looking for places to patch or modify code or to change a command sequence. This is why so much emphasis is placed on showing the blocking and flow of the code via the spacing comments and labeling convention on the **.PRN file. When you are patching code it is imperative that you understand the flow of the code and the usage of each variable. For that purpose, the actual acceptability of the **.MAC file by an existing assembler is moot.

3) A very high level overview of the flow of a program is often useful. The cross reference information could be used to single out, as major components, subroutines with multiple callers, and large blocks of coding with only self-referencing jumps. These would be printed as a crude flowchart for analysis.

--

I intend this list as a catalyst for dialogue. Your inputs are welcome. You may offer items to include, methods of attack, or just comments on the thoughts.

The list, the Z80DIS program, and this manual will be updated from irregularly as time is available for the project. Perhaps some of the extensions would be better handled by additional programs with some common file structure for data sharing within a family of programs.

You may write me by U.S. postal mail or by electronic mail as indicated in SECTION 1 at the beginning of this user manual.

Kenneth Gielow

Table of Contents of the full Z80DIS manual..

How to get the Z80DIS manual

Copies of the Z80DIS USER MANUAL are available by mail. I am asking a nominal charge of $ 20.00 which includes the manual, postage, handling, and an update notification service. The program itself is not available by mail as I do not have extensive, multi-format disk copy capability. The notification service will consist of a letter to you anytime a new version is released so that you may be on the lookout for it.

To order copies of the Z80DIS USER MANUAL by mail, send a request with your name, address, and a check for $ 20.00 to --

Kenneth Gielow
79 Tulip Lane
Palo Alto, CA 94303

This walk through the use of Z80DIS is taken from chapter 5 (page 35) of the larger Z80DIS manual.

- O -

Your first disassembly, a walk-through

The purpose of this section is to walk you through a disassembly using what I would consider the "best" first guess values for all interactive program inputs. I suggest using the widely available public domain program D.COM for your first disassembly so that you may follow along with the example.

Z80DIS is a batch program with user interactive set-up of the control parameters. Disk scratch files are used during the disassembly. I suggest the following file disposition for a two disk drive system: (If you are not using D.COM then substitute your file name as required.)

```
     On drive A - Z80DIS.COM, Z80DIS.000, Z80DIS.001, Z80DIS.002
     On drive B - D.COM
```

During the Z80DIS set up phase direct files as follows -

```
     PRN to your list device (LST:) or to drive B
     Scratch file Z80DIS.$$$ to drive A.
     Initially suppress the MAC file by directing to NONE:
```

If you have a third disk drive, direct the scratch file to that drive.

This optimizes data flow by minimizing disk head action.

The following assumes that you start with the A drive as your selected default drive. Your CP/M input prompt should read:

```
                         A>
```

You start Z80DIS by typing "Z80DIS" with no parameters. The program will prompt you for inputs. In most cases, default entries are shown in reduced intensity already occupying the data field. If you type only the RETURN key, the default will be used.

--

INITIAL PARAMETER SETTING

After a brief sign-on message, you will be asked to supply the following information.

```
|=================================================================
| Please enter  INPUT   file name: ______________
|               OUTPUT  file name: ______________
|               LISTING file name: ______________
|   Descriptive TITLE: ___________________________________________
|
|              file LOAD  address: ____ HEX
|       disassembly START address: ____ HEX
|       disassembly STOP  address: ____ HEX
|
| Do you wish to run a FULL output (as opposed to XREF only)?
| On which disk do you wish the scratch file to reside? (A-G)
|
| Do You wish to process all Z80 codes (as opposed 8080 subset)?
|
| Are all inputs OK so far?
|=================================================================
```

Each question is presented one-at-a-time from the top. Only a single suggested answer will be given for each. You actually have lots of choices possible. For details of the purpose, defaults, editing capability and error checking for each answer please refer to the larger manual.

The following comments apply to all user inputs to Z80DIS:

File Names - The program wants a CP/M style file name in the standard form D:FILENAME.EXT. Each file name, but the very first, has default values selected to simplify your job.

Letter Case - Upper and lower case letters are equivalent and will echo as upper case.

Editing Your Input - All inputs, except single character Y or N answers, must be terminated by a carriage return (shown as <CR>). Until the <CR> terminator is typed, you may correct the entry as required by backspacing.

Terminating Z80DIS - You may terminate Z80DIS by the usual control-C (shown as ^C) at any time. At later stages of execution, only the then current phase will be aborted to provide a graceful recovery of the processing done to that point.

Notation - In the examples to follow, carriage return is sometimes indicated by the notation <cr>.

INPUT file name: ______________

Type the name of your binary file to be disassembled. As I have suggested placing this file on the B drive, you will need a B: drive prefix. The program will default the file extension to

```
               b:d<cr>
```

The program will echo:

```
               B:D.COM
```

OUTPUT file name: ______________

Z80DIS will show the default in reduced intensity as NONE:.MAC. Simply type carriage return (<cr>). The program will echo:

```
               NONE:
```

This will suppress generation of the D.MAC assembly code file.

LISTING file name: ______________

Z80DIS will show the default in reduced intensity as B:D.PRN. I suggest that you change this to direct printer output by typing:

```
               lst:<cr>
```

The program will echo:

```
               LST:
```

Descriptive TITLE: ____

Type the date and time of your disassembly followed by a carriage return. The program will not interpret your input. For the example, I typed:

```
               Demonstration<cr>
```

----
file LOAD address: ____ HEX

Accept the default value (0100 hex) by typing a carriage return.

```
disassembly START address: ____ HEX

     Accept the default value ( 0100 hex) by carriage return.

-----------------------------------------------------------------
disassembly STOP  address: ____ HEX

     Accept the default value by carriage return.

-----------------------------------------------------------------
Do you wish to run a FULL output
      (as opposed to XREF only) ? (Y/N) _

     Type  Y  to select the full output.  No carriage  return  is
needed. Only four characters are accepted for input: y Y n N. All
others are ignored and the program will beep your terminal "bell"
to indicate bad input.

-----------------------------------------------------------------
On which disk do you wish the scratch file to reside? (A-G) _

     Type <cr> to accept the default (A:) drive.

-----------------------------------------------------------------
Do you wish to process all Z80 codes
      (as opposed 8080 subset only) ? (Y/N) _

     As D.COM is an older 8080 CP/M program you should type N. If
your program is known to contain Z80 instructions, type Y.

-----------------------------------------------------------------
Are all inputs OK so far? (Y/N) _

     If all looks OK,  type Y.  A no (N) response will return you
to the first question again.

-----------------------------------------------------------------

                              __
```

BREAK TABLE BUILDING

After you have answered Y (yes) to the "Inputs OK?" question, the screen will clear and your terminal will now display the following:

```
|=================================================================
| >>>  Z80DIS version 2.2
| You may now enter CONTROL BREAK addresses to define the type
|  of disassembly for each section of the code; each control
|  break defines the first address of a section which ends at the
|  beginning of the next section-1 byte.
|
| NOTE: You may select AUTOMATIC CONTROL BREAK assignment by the
|       artificial intelligence expertise of Z80DIS by typing *
|
| TO SEE DETAILS OF YOUR COMMAND CHOICES, type H
|
| ?: _
|=================================================================
```

A break table tells a disassembler how to interpret the binary bytes that it is examining. A disassembler must know whether the bytes are instructions or data in order to proceed.

First type the letter H just to see your choices. Then type the character * to select automatic break table operation. Z80DIS will ask for confirmation: Type Y to confirm.

The * command will generate the break table for you. After automatic assignment returns, you will be shown the break table. You may list the table to your printer if you wish by typing P.

If Z80DIS had trouble reaching a decision about some of the code, anomaly reports will be displayed and you will be given an opportunity to print these reports. I got one message with the version of D.COM that I was running. Note that D.COM replaces itself nearly every time you run it, and your copy will contain different file names in the directory part.

Just for now, print any anomalies if you wish then proceed to exit the break defining process.

To exit the break table building process enter the Q command (Quit). As you have not saved a copy of the changed break table to a file, Z80DIS will ask you to either save the file or type the quit command again. Type Q a second time to start disassembly.

__

DISASSEMBLY PASS ONE

As the disassembly begins the code is cracked according to the break table, the cross reference list structure is linked into the PASCAL heap space in upper memory, the cracked code is copied with context information onto a scratch file called Z80DIS.$$$. During pass one, your console screen will display the following:

```
|=================================================================
| Beginning disassembly...
| THIS IS Pass 1
| working at 05F0 Asc
|=================================================================
```

The "working at" line is animated and will show the current address and disassembly mode. In the illustrated case the program has just finished processing the file D.COM and the last break type was ascii.

--

DISASSEMBLY PASS TWO

During the second pass all user output files and listings are produced and the scratch file is erased. During pass two, your console screen will display the following:

```
|=================================================================
| THIS IS Pass 2
| scratch file contains 396 records of 30 bytes each
| Free memory space remaining after XREF table storage
|   assignments is 29144 bytes out of the original 31300
|   bytes (or   6.9 percent used.)
|
| LISTING cross references
|
| LISTING Subroutines
|
| PRODUCING disassembled output files
|   Processing external label equates
|   Working at 05F0 Asc
|=================================================================
```

The informative messages about file and memory space will let you see just how close to capacity the program is running.

During this pass your printer will start to print the output.

The "working at" line is animated and will show the current address and disassembly mode. In the illustrated case the program has just finished pass two processing of the file D.COM.

When the program is complete, Z80DIS will ring your terminal "bell" twice and display the following sign-off message:

```
|================================================================
| END of Pass 2
|================================================================
```

This section is taken from Appendix A of the Z80DIS user manual (page 43).

- O -

Installation of Z80DIS for your computer

CP/M is a generic operating system, but most of the terminals used with CP/M have features not anticipated by CP/M. These features include CURSOR ADDRESSing and BRIGHT/DIM display of characters. This program utilizes such features, when available, to improve the operator interaction. As the program itself is written in TURBO PASCAL, the TURBO installation support feature has been used to make this adaptation easy.

Therefore, before you use this program, it must be installed to your particular terminal, by providing it with information regarding control characters required for certain functions. This installation is easily performed using the program ZDINSTAL which is described in this section.

The following files are part of this program installation package and must be present during terminal installation (the three ZDINSTAL.* files may then be deleted after installation if no other terminals are to be supported)..

```
     Z80DIS.COM    - Z80DIS main program file
     Z80DIS.000    - overlay file
     Z80DIS.001    - overlay file
     Z80DIS.002    - overlay file

     ZDINSTAL.COM  - the installation program
     ZDINSTAL.MSG  - text of ZDINSTAL messages
     ZDINSTAL.DTA  - terminal characteristics data
```

- THE INSTALLATION PROCEDURE -

Start the installation of Z80DIS by typing "ZDINSTAL" at your terminal. Select Screen installation from the main menu.

A numbered menu listing a collection of popular terminals will appear, inviting you to choose one by entering its number.

If your terminal is mentioned, just enter the corresponding number, and the installation is complete. Before installation is actually performed, you are asked the question..

```
     Do you want to modify the definition before installation? _
```

This allows you to modify one or more of the values being installed as described in the following. If you do not want to modify the terminal definition, just type N, and the installation will complete by asking you the operating frequency of your CPU to establish parameters for timing loops.

If your terminal is not on the menu, however, you must define the required values yourself. The values can most probably be found in the manual supplied with your terminal.

Enter the number corresponding to None of the above and answer the questions one by one as they appear on the screen.

In the following, each command you may install is described in detail. Your terminal may not support all the commands that can be installed. If so, just bypass the command not needed by typing <CR> in response to the prompt. If Delete line, Insert line, or Erase to end of line is not installed, the function will be emulated in software, slowing screen performance somewhat.

Commands may be entered either simply by pressing the appropriate keys or by entering the decimal or hexadecimal ASCII value of the command. If a command requires the two characters <ESC> (escape) and = (equal), you may

either

Press first the Esc key, then the =. The entry will be echoed with appropriate labels, i.e. <ESC> =.

or

Enter the decimal or hexadecimal values separated by spaces. Hexadecimal values must be preceded by a dollar-sign. Enter e.g. 27 61 or $1B 61 or $1B $3D which are all equivalent.

The two methods cannot be mixed in the same definition, i.e. once you have entered a non-numeric character, the rest of that command must be defined in that mode, and vise versa.

A - (hyphen) entered as the very first character is used to delete a command, and the text Nothing to be echoed.

Terminal type: _

Enter the name of the terminal you are about to install. When you complete ZDINSTAL, the values will be stored, and the terminal name will appear on the initial list of terminals. If you later need to re-install Z80DIS to this terminal, you can do that by choosing it from the list.

Send an initialization string to the terminal? _

If you want to initialize your terminal when Z80DIS starts (e.g. to download commands to programmable function keys), you answer Y for yes to this question. If not, just hit <CR>.

If you answer Y, you may choose between entering the command directly or defining a file name containing the command string. The latter is a good idea if the initialization string is long, as a string to program a number of function keys would be.

Send a reset string to the terminal? _

Here, you may define a string to be sent to the terminal when Z80DIS terminates. The above description of the initialization command applies here.

CURSOR LEAD-IN command: _

Cursor Lead-in is a special sequence of characters which tells your terminal that the following characters are an address on the screen on which the cursor should be placed.

When you define this command, you are asked the following supplementary questions:

CURSOR POSITIONING command to send between line and column: _

Some terminals need a command between the two numbers defining the row and column cursor address.

CURSOR POSITIONING command to send after line and column: _

Some terminals need a command after the two numbers defining the row and column cursor address.

Column first? _

Most terminals require the address in the format: first ROW, then COLUMN. If this is the case on your terminal, answer N. If your terminal wants COLUMN first, then ROW, then answer Y.

OFFSET to add to LINE _

Enter the number to add to the LINE (ROW) address.

OFFSET to add to COLUMN _

Enter the number to add to the COLUMN address.

Binary address? _

Most terminals need the cursor address sent in binary form. If that is true for your terminal, enter Y. If your terminal expects the cursor address as ASCII digits, enter N. If so, you are asked the supplementary question..

2 or 3 ASCII digits? _

Enter the number of digits in the cursor address for your terminal.

CLEAR SCREEN command: _

Enter the command that will clear the entire contents of your screen, both foreground and background, if applicable. If you have no command to clear the screen, enter - (hyphen).

Does CLEAR SCREEN also HOME cursor? _

This is normally the case; if it is not so on your terminal, enter N, and define the cursor HOME command.

DELETE LINE command: _

Enter the command that deletes the entire line at the cursor position.

INSERT LINE command: _

Enter the command that inserts a line at the cursor position.

ERASE TO END OF LINE command: _

Enter the command that erases the line at the cursor position from the cursor position through the right end of the line.

START OF 'LOW VIDEO' command: _

If your terminal supports different video intensities, then define the command that initiates the DIM video here. If this command is defined, the following question is asked:

START OF 'NORMAL VIDEO' command: _

Define the command that sets the screen to show characters in 'normal' video.

Number of rows (lines) on your screen: _

Enter the number of horizontal lines on your screen.

Number of columns on your screen: _

Enter the number of column positions on your screen.

Delay after CURSOR ADDRESS (0-255 ms): _
Delay after CLEAR, DELETE, and INSERT (0-255 ms): _
Delay after ERASE TO END OF LINE and HIGHLIGHT On/Off (0-255 ms):

Enter the delay in milliseconds required after the functions specified. <CR> means 0 (no delay).

Is this definition correct? _

If you have made any errors in the definitions, enter N. You will then return to the terminal selection menu. The installation data you have just entered will be included in the installation data file and appear on the terminal selection menu, but instal-lation will not be performed.

When you enter Y in response to this question, you are asked..

Operating frequency of your microprocessor in MHz (for delays): _

As the delays specified earlier are depending on the operating frequency of your CPU, you must define this value. If your processor operates at a fractional speed (i.e. 2.5 Mhz), enter the next larger integer value (3 Mhz for this example.)

The installation is finished, installation data is written to Z80DIS, and you return to the outer menu. New installation data is also saved in the installation data file, and the new terminal will appear on the terminal selection list when you run ZDINSTAL in the future.

Significant Changes in Version 2.2 Released 05/01/87

A. Format Improvements

1) Output format changes for M80 compatibility:
    Singular references now flagged with $ not #
    Inferred references flagged with ?
    No : (colon) on labels on EQU statements
    No address on END card
    Shortened some comments to fit on M80 list page
    Output M80 pseudo-op '.Z80' just before 'ORG' line

2) Comments showing Reference-type on EQU external lines are now preceded by a semi-colon (;).

3) Lower-case 'end' operation changed to upper-case 'END'.

4) Z80DIS now remembers the numeric value on any immediate load of the C register:

```
          LD      C,nn
```

When 'nn' is in the range 0..40 hex, the value is translated and shown on the following BDOS call as a CP/M function.

```
          CALL    BDOS      ; fcnNN= XXXXXXXXXXXXXXX
```

5) DEFS pseudo op now modified to specify filling value found in the region. i.e.

```
          DEFS 1234H,00H
```

When automatic assignment has set the break table, this is the right thing to do because the space area is known to be filled with only a single value throughout. That is not true for a user supplied SPC break but the DEFS is still output as if it was true.

6) Changed format of DEFB print out to try to eliminate the alternation of long and short lines. Also changed automatic break table handling of ASCII data to better resynchronize text lines with the labeled bytes. These are esthetic improvements.

B. Bug Fixes

1) Fixed bug causing occasional truncation of default file name generated for break table save or load.

2) Changed two remaining cases of incorrect address compares to use the correct 16-bit unsigned compare function.

3) Fixed bug when size of SPC blocks get larger than biggest positive integer.

4) Fixed bug causing early termination under certain obscure circumstances involving interactions with zero filled SPC regions.

5) Address table conversion (break type ADT) was modified to change format of output from

```
          DEFW (xxxx)   to
          DEFW xxxx
```

C. Algorithm & Heuristic Improvements

1) Added new controls on recognition of ASCII text. Now check further into string composition for number of common punctuation characters and for number of rare punctuation characters in the string. If excessive, then not ASCII.

2) Improved distinction between strong and weak address references to converge the automatic break assignment with fewer passes over the code.

D. Speed-ups and memory savings

1) Cross Reference Information moved out of real memory to virtual memory on a disk file.

2) Label table records reduced in size by eliminating XREF linkage pointers.

Significant Changes in Version 2.1   Released 01/12/86
----

1) BUG CORRECTION: During automatic break table creation in version 2.0, the codes DD34, DD35, DD36, FD34, FD35, FD36 were not correctly recognized. This resulted in false assumptions about the legality of some Z80 code regions which were then thought to be data regions rather than instruction regions. This has been corrected in version 2.1.

2) IMPROVEMENT IN HANDLING JR REFERENCES: In disassembling 8080 coding, instruction jumps and calls use 3-byte addressing; that is unambiguous and not easily misinterpreted. With Z80 code, ASCII text blocks contain codes that are easily mistaken for Z80 relative jumps; that creates apparent jump references to code within 128 bytes before or after the misinterpreted byte. The ASCII blank (20 hex) is especially bad as it looks exactly like a "JR NZ,xxxx" code. Version 2.1 has been much improved to distrust relative jump references as a basis for declaring a region to be instructions. Version 2.1 examines corroborative evidence to distinguish ASCII data from instruction codes.

3) OTHER IMPROVEMENTS: The entire Expert system comprising the Automatic Break Determination feature was enhanced to yield a more complete analysis of the code structure.

Highlights of Changes in Version 2.0   Released 12/01/85

1) AUTOMATIC BREAK TABLE CREATION: When the program asks for break table inputs, you may now enter '*' which will trigger a detailed analysis of the structure and relationships of the input code.

2) The COM file and overlay files for Z80DIS no longer need to be on your default disk drive.

3) The cross-reference listing now shows the break table region of residency for each label.

4) Overflow of memory or disk space during pass 1 will not abort execution.

5) You may now disassemble direct to your printer by specifying LST: as the list file name.

6) You may now suppress creation of either .PRN or .MAC output files.

7) The default name for the break table save file is now the same name as that of your input file.