ZDM/ZDMZ/ZDMH -- Z-System Tools

Z80/HD64180 DEBUGGER and MONITOR

USER'S GUIDE

by

Robert Doolittle

ZDM/ZDMZ/ZDMH is Copyright 1985 RD SOFTWARE. No part of this document may be reproduced in any way or by any means without prior written permission of publisher. Address requests to Echelon, Inc., 101 First Street, Los Altos, CA 94022.

TABLE OF CONTENTS

Author's Note

These programs have been thoroughly tested and are believed to be correct. If you find something not to your liking let us know. We welcome your comments, criticisms or questions. Please call or write if you experience any problems.

Robert Doolittle
Echelon Z-Team Member
Telephone 213/454-8270
1290 Monument Street
Pacific Palisades, CA 90272

I. INTRODUCTION

ZDM, ZDMZ and ZDMH are Z80 and HD64180 machine language debuggers and monitors designed to run under Z-System (and CP/M*) operating system environment. They recognize and debug 8080, Z80 or HD64180 code although they only run on Z80 and HD64180 machines. The command types and command structure are nearly identical to those of the DDT module provided by Digital Research as part of their CP/M operating system. The major difference is that Z80/HD64180 code can be debugged and ten new commands with variations have been added. ZDM/ZDMZ/ZDMH supports all the DDT commands except the in-line assembly command A. Also, only one breakpoint is implemented.

ZDMZ/ZDMH support Zilog Z80 mnemonics whenever instruction mnemonics are displayed. Additionally, ZDMH supports the enhanced Hitachi instruction set of the HD64180. ZDM uses extended Intel 8080 mnemonics similar to TDL (Xitan) which retains all standard 8080 mnemonics. Please refer to the Appendices for a definition of the mnemonics used by ZDM. Refer also to Section III Installation Procedures before attempting to run ZDM, ZDMZ or ZDMH.

Throughout the remainder of this manual all references to ZDM apply equally to ZDMZ/ZDMH except where otherwise explicitly noted.

ZDM is invoked by typing one of the following three forms at the console:

```
ZDM
ZDM filename
ZDM filename.filetype
```

where "filename" is the name of the file to be loaded. ZDM will then sign-on and relocate itself to overlay the CCP and reside directly below ZRDOS (or BDOS). The jump to ZRDOS at location 5 is altered to address the base location of ZDM which, in turn, contains a jump to ZRDOS or BDOS. Note that ZDM provides an additional page of available transient memory compared to DDT for a given size Z or CP/M system. Like DDT, transient programs loaded for debugging can overwrite the disassembler module. In this case the L command is disabled and the instruction field for the X and T commands is replaced by the corresponding hexadecimal bytes of the instruction.

ZDM has an additional feature to prevent overlaying the ZDM nucleus itself. If this is about to happen, ZDM aborts the load and print an "OUT OF MEMORY" message. It then returns to command level so that those portions which were loaded may be examined.

---

*CP/M is registered trademark of Digital Research. Z-System is trademark of Echelon.

The second and third forms of the console command line result in the named file being loaded after ZDM is entered.

After the sign-on message and program loading, if specified in the command line, ZDM will respond with the prompt character "-" and wait for input commands. Each command consists of either one or two characters, as defined in Section II, which determines the command type. These characters may also be followed by additional parameters. No delimiter should be used between the command type characters and the first parameter except as described for the G command. Subsequent parameters are delimited by a comma or a single space. In all cases, if the command expects a final parameter and this parameter is omitted, ZDM will assume it is zero.

All command lines are terminated by a carriage return. All keyboard input to ZDM and output from ZDM is in hexadecimal. ZDM will accept either upper or lower case letters. A single character (?) is printed if an error occurs. To exit ZDM and return to ZCPR3 or CP/M command level either a Control-C or a G0 (jump to loation 0) may be executed.

II. ZDM COMMANDS

Details of each command are given in this Section.

1. The D (Display) Command. The D command permits the operator to view the contents of memory in hexadecimal and ASCII formats. The forms are:

```
D
Ds
Ds,f
```

In the first case, memory is displayed from the current display address (initially 100H) and continues for the number of lines specified at initialization. Subsequent display addresses are initialized to the value of the program counter following an X, T, U, or G command.

The second form of the D command is similar to the first except that the display address is first set to address s. The third form displays from address s through f. In all cases a subsequent issue of the first form will start with the display address following the last address displayed, resulting in a continuing display. Long typeouts can be aborted with the rubout key.

2. The DI (Disable Interrupt) Command. This command takes the single form:

```
DI
```

The default condition, whenever the target program is entered via the G, T, or U command, is that interrupts are enabled. (Interrupts are always disabled when returning to ZDM). The DI command overrides this default condition. The DI command will remain in effect until a subsequent EI is issued.

3. The EI (Enable Interrupt) Command. This command restores the default interrupt condition. See the DI command description. It takes the single form:

```
EI
```

4. The F (Fill) Command. The F command takes the form:

```
Fs,f,c
```

where s is the starting address, f is the final address, and c is a hexadecimal byte constant. If c is omitted then it is assumed to be zero. This command fills the block of memory from s to f

inclusive with the constant c. If f is less than s an error message (?) will occur.

5. The G (Go) Commands. Program execution is started using the G command with one optional breakpoint address. The G command takes four possible forms:

```
G
Gs
Gs,b
G,b
```

The first form starts execution of the target program at the current value of the program counter and in the current machine state with no breakpoints set. The only way for ZDM to regain control is through a RST 7 execution. The second form is similar to the first except that the program counter is first set to s before execution begins. Third form is the same as the second except that a breakpoint is set at address b. Program execution is stopped and control is returned to ZDM. The instruction at address b is not executed when the breakpoint is encountered. The fourth form starts execution from the current program counter and machine state and sets breakpoint at address b.

Upon entering a breakpoint, ZDM types *d where d is the stop address. The machine state can be examined at this point using the X or &X command.

6. The H (Hex Math) Command. The H command takes the form:

```
Ha,b
```

where a and b are hexadecimal constants from 1 to 4 digits. The sum a+b and the difference a-b are displayed in hexadecimal in the form:

```
a+b a-b
```

7. The I (Input) Command. The I command allows the operator to insert a file name into the Z-System default file control block at 5CH. The default FCB can be used by the program under test as if it had been passed by the Console Command Processor. This command must also be used prior to the R command when reading additional HEX or COM files. The forms of the I command are:

```
Ifilename
Ifilename.filetype
```

If the filetype is anything except HEX then ZDM will assume it is a COM file and the R command will read it into memory starting at 100H. (See the R command for further details).

8. The L (List) Command. The L command is used to list assembly language mnemonics. The three forms of the command are:

```
L
Ls
Ls,f
```

The first form lists the number of lines specified in the initialization and starting at the current address of the program counter. The second form lists the same number of lines but starts at the address s. The third form starts at the address s and continues for f lines. All three forms can be continued with a subsequent L command similar to the D command. Also, like the D command, the starting address if not specified is always initialized to the program counter following an X, T, U or G command. Long typeouts can be aborted with the rubout key.

9. The M (Move) Command. The M command will move a block of memory from one location to another. The form of the M command is:

```
Ms,f,d
```

where s is the start address of the move, f is the final address and d is the destination address. If f is less than s, an error (?) will occur.

10. The QI (Query Input) Command. The QI command allows the operator to read an input port and display the value at the port address. The form of this command is:

```
QIa
```

where a is a one byte port address in hexadecimal. The value is printed immediately following execution of this command. Note that if a is omitted, it is assumed to be zero.

11. The QO (Query Output) Command. The QO command allows the operator to output a specified byte to a specified port address. The form of this command is:

```
QOa,b
```

where a is the port address and b is the byte to be output. If either a or b is omitted, it is assumed to be zero.

12. The R (Read) Command. The R command is used in conjunction with the I command to read COM and HEX files from disk into memory. There are two forms of this command:

```
R
Rb
```

where b is an optional offset address which is added to each program or data address as it is loaded. If b is omitted then it is assumed to be zero. Note that if the file name in the FCB from a previous I command is not a HEX type then ZDM assumes it is COM and will load it at 100H or 100H+b if the parameter b is included. If the file cannot be opened or an error occurs in reading, ZDM responds with the error indicator (?). Otherwise at completion of the load a message is issued:

```
NEXT  PC
nnnn pppp
```

where nnnn is the next address following the program just loaded and pppp is the first address of the program just loaded. For HEX files pppp is taken from the last record of the HEX file and will be zero unless an END statement followed by the start address has been included in the source program prior to assembly.

13. The S (Set) Command. The S command allows memory locations to be examined and optionally altered. The form of the command is:

```
Ss
```

where s is the hexadecimal starting address for examination or alteration of memory. ZDM will print the address followed by the byte stored at that address. A carriage return will advance to the next address, displaying the next address and the next byte. If a new byte value is typed followed by a carriage return, this new value will be stored at that address and ZDM will advance automatically to the next address. To terminate the command a period is typed rather than a byte value. The command will also terminate if an invalid hexadecimal value is entered.

14. The T (Trace) Command. The T command permits single step instruction tracing of program execution for 1 to 65535 steps. The forms of this command are:

```
T
Tn
```

where n is an optional step number. The first form assumes an implied n equal one. The CPU state is displayed and the next program step is executed. The termination address is displayed as *hhhh where hhhh is the next address to be executed. The format for the CPU state display is otherwise identical to that of the X command.

Program tracing is discontinued at the interface to Z-System and resumes again after return from Z-System to the target program. Long tracing with the Tn command can be stopped with the rubout key. ZDM will continue tracing from this break if another T or Tn command is issued.

15. The U (Untrace) Command. The U command is identical to the T command except that the CPU state is not displayed. The forms of the command are:

```
U
Un
```

All conditions of the T command apply to the U command. The last CPU state is displayed following the execution of a U or Un command.

16. The X (Examine) Command. (See also the & Command). The X command permits selective display and alteration of the current CPU state at any time. The forms are:

```
X
Xr
```

where r is any of the Z80 registers or flags.

C	Carry Flag	(0/1)
Z	Zero Flag	(0/1)
M	Minus Flag	(0/1)
E	Even Parity Flag	(0/1)
I	Interdigit Carry	(0/1)
A	Accumulator	(0-FF)
B	BC register pair	(0-FFFF)
D	DE register pair	(0-FFFF)
H	HL register pair	(0-FFFF)
S	Stack Pointer	(0-FFFF)
P	Program Counter	(0-FFFF)
X	X-index register	(0-FFFF)
Y	Y-index register	(0-FFFF)

In the first case the CPU register state is displayed in the format

```
CfZfMfEfIf A=bb B=dddd D=dddd H=dddd P=dddd S=dddd
X=dddd Y=dddd instruction
```

where f is a 0 or 1 flag value, bb is a byte value and dddd is a double byte corresponding to a register pair. The instruction field contains the disassembled instruction at the location addressed by the program counter.

The second form of the X command permits display and optional alteration of register or flag values specified by r. If a carriage return is typed following an Xr command then the

command is terminated with no changes taking place. Otherwise ZDM accepts input for register or flag changes. If a hexadecimal number in the proper range is typed then that flag or register is correspondingly altered.

17. The & (Alternate Register) Command. The & command takes one of three forms:

```
&
&X
&Xr
```

The first form unconditionally exchanges all CPU registers and flags to the Z80 alternate register set.

The second and third forms of this command are identical to the X and Xr commands except that the operations take place on the alternate register or flag set. Upon termination of these latter two forms the CPU state prior to command execution is restored. The display associated with the &X command replaces the X and Y registers by the vector interrupt register value V. This register value may also be altered by an &XV command followed by the byte value to be stored in the vector interrupt register. The A, B, D and H registers and register pairs are labeled by prime (') symbols whenever the alternate set is being displayed or altered. Note that primes are not used for the flag register except during alteration.

18. The B (Block Search) Command. The B command permits the user to search memory for all occurrences of a byte string. Strings are limited to ten bytes. A second form of this command is initiated by BT rather than B. This second form will accept an ASCII or text string. The form of this command is:

```
Bs,f or BTs,f
```

where s is the start address and f is the final address of the memory block to be searched. Following the carriage return you will be prompted to enter the string. The B form expects the string as a series of HEX bytes. The delimiter may be a space or a comma. The BT form expects a single ASCII string. The input

string is terminated by a carriage return following the last entry. The start address of each occurrence of the string from s to f will be displayed.

19. The V (Verify) Command. The V command will verify if two blocks of memory are identical. The form of this command is:

Vs,f,b

where s is the start address and f is the final address of one block and b is the start address of the other block. If the

match fails, the address is printed out followed by the byte at the corresponding address in the second block.

20. The P (Print) Command. The P command is a toggle which does not expect any arguments. The effect is to send all output to the LIST device as well as to the console. It is turned off by a subsequent P command. Whenever the P toggle is on, a 'P' will be displayed as part of the X or T display.

21. The J (Jump) Command. The J command is a toggle which does not expect any arguments. It only affects subsequent T or U commands. If J has been executed then the T command will display only conditional and unconditional CALLS, JUMPS, RETURNS, RESTARTS, PCHL (IX or IY) and relative JUMPS. The Tn form of the T command is usually used where n represents the actual number of instructions to be traced. As usual this command can be aborted with the rubout key. Whenever the J toggle is on, a 'J' will be displayed as part of the X or T display.

III. INSTALLATION PROCEDURES

The user should first make a copy of ZDM and keep the original master as a back-up. Do not write on the master. Using the copy type "ZDM". ZDM will respond with the question "What is your terminal width (in hex)". A carriage return in response to this question will default to a width of 80(50H). Otherwise, type in the character width of your terminal in hex followed by a carriage return. Next, the number of lines desired for the D and L command displays are requested. A carriage return at this point defaults to 21(15H) lines. (This is the recommended size for 24 line terminals.) Otherwise, type in the number of lines desired, in hex, followed by a carriage return.

Finally, ZDM will ask the question "Is this correct? (Y or N)". Do not respond with "Y" at this time. Following an "N" response ZDM will sign-on, print the prompt character "-", and await a command. You should now test the D and L commands to determine if these displays are sized properly for your terminal. If not, return to Z-System command level by typing G0 or Control-C and repeat the above procedure. When you are satisfied that the displays are properly sized, then respond with "Y" when asked "Is this correct? (Y or N)". After a "Y" response ZDM will automatically create your custom installed file. If you are using ZDM then the installed file name will be ZDI.COM. If you are using ZDMZ or ZDMH then the installed file name is either ZDIZ.COM or ZDIH.COM. You may rename these to whatever names you desire.

ZDM is now properly configured for your terminal. Subsequent invocations of ZDM will proceed directly to the sign-on message. If, at some later time, you wish to change these display parameters you will have to start once again with a copy of the master uninstalled file.

APPENDIX A

Some users have experienced difficulties with ZDM/ZDMZ when used with interrupt driven systems. (ZDMH has interrupts enabled as default condition. You can still disable interrupts with the DI command when entering the target program but they will be enabled again when ZDMH regains control.) The following patches are recommended when running ZDM/ZDMZ on such systems.

Using ZDMH or other debugger, load an image of ZDM/ZDMZ into memory starting at 100h and change the following bytes (addresses apply to version 3.2):

ZDM ADDRESS	ZDMZ ADDRESS	FROM	TO
0C61H	OC31H	0F3H	00H
0C7AH	0C4AH	0F3H	00H
0DC1H	OD91H	0F3H	00H

Return to Z-System or CP/M without disturbing the memory image and save 22 (or 16h) pages with the SAVE command.

Another frequent user request has been to change the ZDM RESTART address, curently RST 7 at 38h, to a different RESTART. The following patches will accomodate this change. (RST 0 cannot be used under Z operating system or under CP/M.)

ZDM ADDRESS	ZDMZ ADDRESS	ZDMH ADDRESS	FROM	TO
0D9DH	0D6DH	0DEDH	38H	new RESTART address
0DA5H	0D75H	0DF5H	39H	new RESTART address + 1
1007H	0FD7H	1057H	0FFH	new RESTART opcode

APPENDIX B

The disassembler module of ZDM uses a mnemonic set which is similar to the Technical Design Laboratories (TDL) mnemonics. All Intel 8080 mnemonics are preserved. The Z80 peculiar instructions differ from the Zilog mnemonics as shown in the accompanying table. The ZDM mnemonic set is nearly identical to that released by Digital Research as Z80.LIB to be used with their CP/M Macro Assembler "MAC". The following conventions are used in the table.

```
    r - any register or memory
   rr - any register pair or stack pointer
   nn - 8 bit immediate data (0 to 255)
    d - 8 bit signed displacement (-128 to 127)
 nnnn - 16 bit address or immediate data (0 to 65535)
    b - bit number (0 to 7, 7 is most significant)
 addr - 16 bit address within PC+127 through PC-128
```

In cases involving a displacement, d, this parameter is always last one in operand field.

ZDM	ZILOG	ZDM	ZILOG
LDX r,d	LD r,(IX+d)	LDIR	LDIR
LDY r,d	LD r,(IY+d)	LDD	LDD
STX r,d	LD (IX+d),r	LDDR	LDDR
STY r,d	LD (IY+d),r	CCI	CPI
MVIX nn,d	LD (IX+d),nn	CCIR	CPIR
MVIY nn,d	LD (IY+d),nn	CCD	CPD
LDAI	LD A,I	CCDR	CPDR
LDAR	LD A,R	ADDX d	ADD (IX+d)
STAI	LD I,A	ADDY d	ADD (IY+d)
STAR	LD R,A	ADCX d	ADC (IX+d)
LXIX nnnn	LD IX,nnnn	ADCY d	ADC (IY+d)
LXIY nnnn	LD IY,nnnn	SUBX d	SUB (IX+d)
LBCD nnnn	LD BC,(nnnn)	SUBY d	SUB (IY+d)
LDED nnnn	LD DE,(nnnn)	SBBX d	SBC (IX+d)
LSPD nnnn	LD SP,(nnnn)	SBBY d	SBC (IY+d)
LIXD nnnn	LD IX,(nnnn)	ANAX d	AND (IX+d)
LIYD nnnn	LD IY,(nnnn)	ANAY d	AND (IY+d)
SBCD nnnn	LD (nnnn),BC	XRAX d	XOR (IX+d)
SDED nnnn	LD (nnnn),DE	XRAY d	XOR (IY+d)
SSPD nnnn	LD (nnnn),SP	ORAX d	OR (IX+d)
SIXD nnnn	LD (nnnn),IX	ORAY d	OR (IY+d)
SIYD nnnn	LD (nnnn),IY	CMPX d	CP (IX+d)
SPIX	LD SP,IX	CMPY d	CP (IY+d)
SPIY	LD SP,IY	INRX d	INC (IX+d)
PUSHIX	PUSH IX	INRY d	INC (IY+d)
PUSHIY	PUSH IY	DCRX d	DEC (IX+d)
POPIX	POP IX	DCRY d	DEC (IY+d)
POPIY	POP IY	NEG	NEG

ZDM	ZILOG	ZDM	ZILOG
EXAF	EX AF,AF'	IM0	IM 0
EXX	EXX	IM1	IM 1
XTIX	EX (SP),IX	IM2	IM 2
XTIY	EX (SP),IY	DADC rr	ADC HL,rr
LDI	LDI	DSBC rr	SBC HL,rr
DADX rr	ADD IX,rr	OUTI	OUTI
DADY rr	ADD IY,rr	OUTIR	OTIR
INXIX	INC IX	IND	IND
INXIY	INC IY	INDR	INDR
DCXIX	DEC IX	OUTD	OUTD
DCXIY	DEC IY	OUTDR	OTDR
BIT b,r	BIT b,r	RLCR r	RLC r
SET b,r	SET b,r	RLCX d	RLC (IX+d)
RES b,r	RES b,r	RLCY d	RLC (IY+d)
BITX b,d	BIT b,(IX+d)	RALR r	RL r
BITY b,d	BIT b,(IY+d)	RALX d	RL (IX+d)
SETX b,d	SET b,(IX+d)	RALY d	RL (IY+d)
SETY b,d	SET b,(IY+d)	RRCR r	RRC r
RESX b,d	RES b,(IX+d)	RRCX d	RRC (IX+d)
RESY b,d	RES b,(IY+d)	RRCY d	RRC (IY+d)
JR addr	JR addr	RARR r	RR r
JRC addr	JR C,addr	RARX d	RR (IX+d)
JRNC addr	JR NC,addr	RARY d	RR (IY+d)
JRZ addr	JR Z,addr	SLAR r	SLA r
JRNZ addr	JR NZ,addr	SLAX d	SLA (IX+d)
DJNZ addr	DJNZ,addr	SLAY d	SLA (IY+d)
PCIX	JP (IX)	SRAR r	SRA r
PCIY	JP (IY)	SRAX d	SRA (IX+d)
RETI	RETI	SRAY d	SRA (IY+d)
RETN	RETN	SRLR r	SRL r
INP r	IN r,(C)	SRLX d	SRL (IX+d)
OUTP r	OUT (C),r	SRLY d	SRL (IY+d)
INI	INI	RLD	RLD
INIR	INIR	RRD	RRD

* *
* *
* *

APPENDIX C

HITACHI HD64180 MNEMONICS

Object Code	Source Statement		Operation
ED3805	IN0	A,(nn)	Load register with input from port (nn).
ED0005	IN0	B,(nn)	
ED0805	IN0	C,(nn)	
ED1005	IN0	D,(nn)	
ED1805	IN0	E,(nn)	
ED2005	IN0	H,(nn)	
ED2805	IN0	L,(nn)	
ED4C	MLT	BC	Unsigned multiplication of each half of the specified register pair with the 16-bit result going to the specified register pair.
ED5C	MLT	DE	
ED6C	MLT	HL	
ED7C	MLT	SP	
ED8B	OTDM		Load output port (C) with location (HL), decrement HL, B, and C.
ED9B	OTDMR		Load output port (C) with location (HL), decrement HL, B, and C. Repeat until B=0.
ED83	OTIM		Load output port (C) with location (HL), increment HL and C. Decrement B.
ED93	OTIMR		Load output port (C) with location (HL), increment HL and C. Decrement B. Repeat until B=0.
ED3905	OUT0	(nn),A	Load output port (nn) from register.
ED0105	OUT0	(nn),B	
ED0905	OUT0	(nn),C	
ED1105	OUT0	(nn),D	
ED1905	OUT0	(nn),E	
ED2105	OUT0	(nn),H	
ED2905	OUT0	(nn),L	
ED76	SLP		Enter sleep mode.
ED3C	TST	A	Non-destructive AND with accumulator and specified operand.
ED04	TST	B	
ED0C	TST	C	
ED14	TST	D	
ED1C	TST	E	
ED24	TST	H	
ED2C	TST	L	
ED6405	TST	nn	
ED34	TST	(HL)	
ED7405	TSTIO	nn	Non-destructive AND of (nn) and the contents of port (C).

APPENDIX D

ZDM/ZDMZ/ZDMH COMMAND SUMMARY

Function	Form	Definition
Display	D[s,f]	display screen of memory in hex and ASCII
Disable Interrupt	DI	disable interupts, normal default
Enable Interrupt	EI	enable interupts, default if entering from G, T, and U
Fill	Fs,f,c	fill range of memory with declared byte value
Go	G[s,b]	execute program with optional breakpoint
Hex Math	Ha,b	obtain sum and difference of two hex numbers
Input	Ifilename	set up file control block to receive file name
List	L[s,f]	list to screen assembly language mnemonics
Move	Ms,f,d	move data from one area of memory to another
Query Input	QIa	display input byte from indicated port a
Query Output	QOa,b	output byte b to indicated port a
Read	R[b]	read in file set up with I command, optional offset
Set	Ss	examine and optionally alter memory
Trace	T[n]	sinple step program execution, up to 65535 steps
Untrace	U[n]	similar to T, but CPU state not displayed
Examine	X[r]	examine CPU register values
Alternate Register	&[X][r]	examine Z80 alternate register values
Block Search	B[T]s,f	find ASCII or hex string in declared memory range
Verify	Vs,f,b	verify if two blocks of memory are identical
Print	P	send all screen output also to printer
Jump	J	display only branch statements: calls, jumps, returns, etc.

Legend: items in [ ]'s are optional; s=start address; f=final address; c=hex byte value; a=hex value or port address; b=hex value or offset, breakpoint or block start address; d=destination address; n=step number; r=register letter, a for accumulator, b for bc pair, s for sp, etc.