

DDTZ v2.5

BY C.B. FALCONER

EDITED BY GEORGE A. HAVACH

Introduction:

DDTZ v2.5 is a complete replacement for DDT, Digital Research's famous Dynamic Debugging Tool, with improved functionality, bug extermination, and full Z80 support. In general, DDTZ is fully compatible with the original utility, but it has extra and extended commands and many fewer quirks. All Z80-specific instructions can be (dis)assembled, though in Intel rather than Zilog format. Furthermore, DDTZ will correctly trace ('T' and 'U' commands) both 8080 and Z80 instructions, depending on which CPU is operating. On startup, the program announces which CPU it is running on.

The program is invoked by typing

```
    ddtz<ret>
or
    ddtz [d:]filespec<ret>
```

In the second form, DDTZ will load the specified file into memory starting at 0100H, unless it's a .HEX file that sets its own load address. Besides reporting the NEXT free address and the PC (program counter) after a successful load, DDTZ also shows the number of memory pages needed for a SAVE. Instead of having to write all this down, just use the 'Q' command at any time to redisplay these three values for the current application.

NOTE: loading more code above the NEXT pointer revises these values.

As in DDT, when a program is loaded above the area holding the 'A' and 'L' (and now 'K') command code, these commands are disabled, and the extra memory is released to the user. Thus, DDTZ can occupy as little as 3K total memory space. Unlike DDT, however, DDTZ will not overwrite itself or the system on program loads (except .HEX files).

At initialization, the stack pointer (SP) points to a return to DDTZ, just like for the CCP. Thus, programs that normally return to the CCP will be returned to DDTZ. The 'B' command reinitializes this condition.

The intercept vector copies the BDOS version number, etc., so an object program does not know that DDTZ is running (except for BIOS-BDOS vector size). Thus, programs that check the version number should execute correctly under DDTZ.

All input parameters can now be entered in any of three formats:

(1) hexadecimal (as in DDT),

- (2) decimal, by adding a leading '#' character,
- (3) ASCII, by enclosing between either single or double quotes; either one or two characters are allowed.

Leading blanks in command lines and parameters are absorbed. Either a comma or a (single) space is a valid delimiter. Either uppercase or lowercase input is accepted.

The default command (for anything not otherwise recognizable) is 'H'. This allows convenient calculation, along with the other features described below. So, to convert a number, just enter it!

As in DDT, the prompt character is '-', and the only error message is the query ('?'), which generally kicks you back to command mode.

New Commands (Over DDT)

NOTE: letters in parenthesis, e.g. "(U)", show the equivalent command for DDTZM version (compatible with MSDOS debug).

@ Sets or shows (with no parameter) the internally stored "base" value. Also used with the 'S' and 'D' commands as an optional parameter (though without the '@') to display memory from an arbitrary base marker (offset). When set to zero (the default), it does not affect any screen displays.

B B)egin: resets the USER stack pointer to its initial value, such that any program that exits by an RET will return to DDTZ. DDTZ provides a default stack space of approximately 24 bytes for user programs.

C C)ompare first_address,last_address,against_address: shows all the byte differences between two memory areas, in the format

```
XXXX aa YYYY bb
```

where XXXX and YYYY are the comparative memory addresses, and aa and bb are the corresponding byte values. Can be used to verify the identity of two files by first loading them into different memory areas with the 'R' command (see below).

K K)eep: stores the modified memory area to disk under the file-name specified by the 'I' command, overwriting the original file from which it was loaded (the user is queried before doing so). By default, the image of memory from 0100H through the "NEXT" value -1 is saved. "K first_address,last_address" overrides this and allows writing ANY memory area to a file. Almost a necessity for CPM 3.0 (no SAVE!).
(W) W)rite on DDTZM

Q Q)uery: redisplays the "NEXT PC SAVE" report at any time.
(X) eX)amine size on DDTZM.

W W)here first_address,last_address,value: searches the specified
(S) memory area for the value (a 16-bit word, not a byte) and shows

the locations of all such. Very useful for finding CALL's or JMP's to a particular address, etc.

Search on DDTZM

- Y** Y)our_option parm1,parm2,address: executes an arbitrary routine at the specified address, with the BC and DE registers set to parm1 and parm2, respectively.
- Z** Displays (but does not alter) the Z80's alternate register set, including the index registers (disabled if running on an 8080). On Z80's, automatically included as the last part of the display by the 'X' command.

Based (Offset) Displays

The 'D' and 'S' commands can use a stored base value (offset), as set by the '@' command. The current @ value may be overridden for a single execution of these commands by adding the base as an extra parameter in the command line. The effect is to add this value to the first/last address and display accordingly. The address listing on the left becomes XXXX:YYYY, where XXXX is the offset address and YYYY is the actual memory address being displayed. For example, if you have a data area located at 42B7H and wish to preserve easy access, just enter "@42b7". Now, "d0,3f" will dump memory starting at 4237H.

Further Changes from DDT

- A** A)ssemble now accepts the full Z80 as well as 8080 instruction set, although it expects them in Intel rather than Zilog format (see notes below under the 'L' command). When in doubt, try poking in the hexcode (with the 'S' command) and then L)isting to see what the (dis)assembler recognizes.
- D** D)isplay or D)ump will accept an optional third parameter to set the base value for a single execution only. Format has been cleaned up.
- H** H)ex_arithmetic on two values also shows their difference in decimal. With only one value, converts to hexadecimal, decimal, and ASCII (low-order byte only).
- I** I)npnt now allows drive specification (d:...) and sets up the complete command line, including both FCB's (at addresses 005CH and 006CH). The tail (stored at 0081H up) is NOT upshifted.
- (N)** N)ame on DDTZM
- L** L)ist now displays the raw hexcode, especially handy when examining non-code areas. Intel (8080 style) mnemonics are used, so some disassembled instructions may look a little strange. E.g., the Z80's 'IN B,(C)' and 'OUT (C),B' become 'INP B' and 'OUTP B', respectively; 'LD (nnnn),BC' becomes 'SBCD nnnn', 'ADD IX, BC' becomes 'DADX B', and 'JP (IX)' becomes 'PCIX'.
- (U)**

U)nassemble on DDTZM

R R)ead now permits loading a file into memory with an offset,
(L) which is added to the default load address of 0100H. When
 reading in a .HEX file with a preset bias, the 'R' command
 will not transfer control to an invalid execution point. Ano-
 ther execution of the 'R' command will reread the input file,
 e.g.:

```
i blah<ret>
r<ret>
...modify the code and generally mess about...
r<ret>
```

The original file is reloaded, and the modifications are re-
 moved.

L)oad on DDTZM.

S S)ubstitute or S)et, like D)isplay, now accepts an optional
(E) second parameter to set the base value for a single execution
 only.

E)nter on DDTZM

T T)rap/trace on termination now shows the complete CPU state.
 Traps and traces no longer lock up when a user RST 7 instruc-
 tion is executed. Tracing of BDOS/BIOS calls is heavily trun-
 cated, avoiding clutter and preventing system crashes.

NOTE: The UNDOCUMENTED Z80 op-codes are not handled. Can crash.

X eX)amine also shows what two-byte values the HL and SP regis-
(R) ters are actually pointing to. On Z80's, displays the alternate
 register set.

R)egisters on DDTZM.

NOTE: Any use of the 'K' or 'R' command resets the system DMA trans-
 fer address to the standard default value of 0080H.

Command Summary

DDTZ command

=====

@ (base)

```
A)ssemble first_address
B)egin {i.e., initialize stack and return}
C)ompare first_address,last_address,against_address
D)ump first_address[,last_address[,base]]
F)ill first_address,last_address,value
G)o_to [address][,trap1[,trap2]]
H)ex_arithmetic value1[,value2]
I)ntput FCBs_command_line
K)eeep [first_address,last_address]
L)ist_code first_address[,last_address]
M)ove first_address,last_address,destination
Q)uery {i.e. display memory parameters for application}
R)ead_file (offset)
```

DDTZM command

=====

```
A
B
C
D
F
G
H
N)ame
W)rite
U)nassemble
M
X)amine
L)oad
```

S)ubstitute_in_memory first_address[,base]	E)nter
T)race_execution [count]	T
U)ntrace_execution [count] (i.e. do count instr)	(NOT AVAILABLE)
W)here_is first_address,last_address,word	S)earch
X)amine/change [register flag]	R)egister
Y)our_option BC:=parm1,DE:=parm2,call_address	Y
Z)80_register_display	Z
(on DDTZM only -->)	Q)uit