The Alternate Source MONitor (TASMON4)

Original:
Bruce Hansen

Model 4 Adaptation:
Paul Snively

by The Alternate Source
704 North Pennsylvania Avenue
Lansing, MI 48906
(517) 482-8270

TABLE OF CONTENTS

TASMON - MONITOR PROGRAM FOR THE TRS-80

With TASMON (The Alternate Source's Monitor), memory may be examined/modified and machine language programs executed. Machine language programs may be run in real time, single step or slow motion. Your Z-80 registers may be examined/modified. They are continuously displayed in the upper right part of the screen. Three different memory dumps can be displayed on the left side of the screen while executing any TASMON command on the right side of the screen. Memory can be disassembled and routed to disk as an Editor/Assembler source file with labels generated for pertinent addresses. Machine language disk files can be read in and written out.

INSTALLING TASMON4

There is a program on the distribution disk called INSTALL/CMD. You will have to have this program on one of your disks in order to put TASMON4 on your working TRSDOS 6.x backup. After you have used the INSTALL program once, you can put it away. All other copies of TASMON4 can be made by backing up the working master that you make with INSTALL.

Running INSTALL/CMD is easy. Just make sure that the file SYS13/OBJ is on a copy of TRSDOS 6.x in drive 0 and type INSTALL. As long as INSTALL is in a drive and SYS13/OBJ is on a TRSDOS 6.x disk in drive 0, all should go well, and after accessing drive 0 for a while, INSTALL should say "TASMON overlay installed." If something should go wrong, INSTALL will give an error message and abort back to TRSDOS Ready. Correct the error and try again. REMEMBER - FURTHER BACKUPS CAN BE MADE FROM THE INSTALLED TRSDOS 6.x DISK!!! It is also important to note that after INSTALL has run, SYS13/OBJ will no longer appear in your TRSDOS 6.x directory. That's ok. INSTALL has actually put the code from SYS13/OBJ into SYS13/SYS and killed SYS13/OBJ.

TASMON FORMATS AND CONVENTIONS

All numbers displayed by TASMON are hex unless otherwise noted.

In all examples given below, user inputs are underlined.

The version of TASMON on the distributed diskette loads in memory from E000-FFFF with an entry point of E000.

There is also a short machine language program entitled "TEST/CMD" which is used in the sessions discussed at the end of this manual.

For the most part, TASMON uses single letter commands and the ENTER key is not needed. The BREAK key may be depressed at any time to exit a command (except when writing/loading files to/from disk.) The LEFT ARROW does not backspace the cursor (unless entering a file name) so hit the BREAK key and re-enter the command if a mistake is made.

When a four or two digit number is being input, any and all leading zeros must be entered.

TASMON uses the DOS keyboard, video and printer routines. If a key is held down, after a pause it will start repeating. The video display supports upper/lower case. TASMON displays the down-arrow key on the video display as a backslash which we have chosen to represent with an exclamation point (!). So, when you see an exclamation point, think "down-arrow."

When TASMON is entered, the user's stack pointer is set at 03A0. The user may change it as needed provided it does not interfere with TASMON.

The register display appears on the the right side of the screen in this format:

```
CALL    01C9
IX  52A8   IY  F29A
AF' E23A   BC' 1530
DE' 06FA   HL' BC09
AF  0044   BC  028A
DE  D980   HL  3DF5
SP  03A0   PC  8000
-Z---P--   (HL)  FF
```

The top line is the Zilog disassembled mnemonic of the instruction pointed to by the PC register (8000 in this case). The four digit hex number following each register pair is the current value of that register pair. The last line of the display is the status of the Z-80 flags (F register). A "-" indicates that the bit is cleared. The "FF" following "(HL)" is the value found at the current address of HL. In this case "FF" is at memory location 3DF5. All user inputs are done on the sixteen lines below the register display.

MEMORY USAGE

TASMON uses approximately 8K of memory. Some SVC's are used to decrease program size. No RAM outside of TASMON is used with the exception of the symbol table when disassembling to disk, the user's screen memory when using the KEEP SCREEN command, and the DOS overlay area. One of the most commonly made errors with TASMON is using it to disassemble to disk and not reserving enough room in RAM for the symbol table. If you want to disassemble to disk, the best way is to make sure that TASMON resides from E000-FFFF and to set HIGH$ to DFFF. Before entering TASMON, type:

MEMORY (HIGH=X'DFFF')

at TRSDOS Ready, which will set HIGH$ to DFFF. Then enter TASMON by typing:

TASMON4

at TRSDOS Ready. Finally, if TASMON4 isn't at E000-FFFF (it is, if you loaded it straight from our distribution disk) then relocate it with the X command (described in detail in the command summary.)

LOADING AND REENTERING TASMON

For executing the program from disk, type the following from DOS:

TASMON4

The Z-80 registers and the caret user prompt will be displayed.

If TASMON is exited for some reason there are two ways to reenter the monitor:

1) Go to BASIC and type:

DEFUSR=&HE000 <ENTER> (assuming that TASMON is at E000)
X=USR(0) <ENTER>

where "E000" is the hex starting address of where TASMON was located in memory.

2) With a DOS system enter DEBUG and type:

G E000

where "E000" is the hex starting address of where TASMON was located in memory.

When TASMON is reentered in this manner the user's registers are not changed. However, any breakpoints that were set are cleared.

EXITING TASMON

The "E" or EXIT command is used to leave TASMON. To execute this command enter:

E <ENTER>

The ENTER key needs to be pressed as a safety precaution to prevent exiting the monitor unexpectedly.

TASMON COMMANDS

The following is a list of TASMON's commands and the format with which they are entered.

REPLACE REGISTERS

The REPLACE command changes any of the Z-80 registers. To use the command press "R", the first letter of the register pair to be changed (the second letter is also required for IX and IY), and the new four digit register value. An apostrophe is typed after the register pair name if the secondary set is to be changed. The display will appear as follows:

R HL 09AF	set HL to 09AF
R AF' 2044	set AF' to 2044

MODIFY MEMORY

The Modify memory command allows the user to change the contents of RAM. To execute the command press "M", an "A" or "H" (for modification to be in ASCII or hex mode) followed by the address to modify. If the ENTER key is depressed for the address to modify, the current PC address is used for the starting modification address.

The ASCII modification mode accepts single character ASCII values and places them in the addresses being modified. The only control code recognized is the carriage return. The hex mode puts two digit hex values into an address. The current contents of the modification address will be displayed in the format:

ADDRESS ASCII HEX

Where "ADDRESS" is the memory address being modified, "ASCII" is the ASCII value of the byte at "ADDRESS" (only displayed if the value is between 20-7F) and "HEX" is the hex value of the byte.

To change the contents of ADDRESS, type in a new two digit hex value or one ASCII character depending on which modification mode was selected. The up arrow will leave the current address unmodified and move to the previous byte. The down arrow will move down to the next byte. To exit, hit the BREAK key. A typical display is:

M H 707F C 43 AE   Hex modify mode, address is 707F, ASCII value of byte at that address is "C", the hex value is 43 and the byte was changed to AE.

7080 82 <UP ARROW>	Move up one byte
707F AE <BREAK>	Exit modify mode

MEMORY DUMPS

There are three memory dumps in TASMON:

1) Hexadecimal dump
2) ASCII dump
3) Disassembled dump

All three dumps are initiated by pressing the appropriate command key followed by a four digit hex value where the dump is to start. Pressing the ENTER key instead of the four digit starting value causes the dump to begin at the current PC register. The screen will clear and 15 lines of the dump will be displayed.

Pressing the SPACE BAR at this point causes the next 23 lines to be displayed. The DOWN ARROW is used to display the next line. Pressing the "-" key causes the display to move back in memory 23 lines (B8H bytes with ASCII and Hex dumps, 23 instructions with disassembler). Holding down any of these command keys will cause them to repeat. Pressing the BREAK key, as always, returns control to command mode.

HEX DUMPS

The hex dump will display the hexadecimal values of memory starting with the address entered. For example:

<u>H</u> <u>5200</u>

will cause this type of display to appear on 23 lines:

```
5200 45 AF 20 0F C3 DD 00 ED
5208 34 A8 FF FF 99 83 FA 00
```

The 5200 is the address and 45 is located there, 5201 holds AF, etc.

ASCII DUMPS

The ASCII dump will display 20-7F values as the appropriate ASCII character. All other values are displayed in hex. For example:

<u>A</u> <u>F00C</u>

will cause this type of display on 23 lines:

```
F00C  T  H  E 00 03  B  O  Y
```

DISASSEMBLED DUMP

The Disassembled dump will display memory in Zilog mnemonics. This dump makes reading programs much easier. For example:

D 0000

will cause the screen to clear and fifteen lines like the following to appear:

```
0000 FE         DI
0001 AF         XOR      A
0002 C37406     JP       0674
```

Relative jump addresses are displayed giving their destination address (like an absolute jump) instead of a relative offset.

Illegal instructions are disassembled with "DEFB h" as the instruction where "h" is the offending byte. For example:

```
8000 CB         DEFB     CB
8001 3007       JR       NC,800A
```

DISASSEMBLED LISTING TO PRINTER

The "P" or PRINT command is used to route disassembly to the printer. To run this command press "P", the starting address of the dump and the ending address of the dump. The disassembly is also echoed on the screen. Pressing the BREAK key at any time will cause printing to stop. If this command is executed and the printer is not ready, control returns to TASMON with nothing printed. For example:

P 0000 00F0   Disassemble to the printer starting at 0000 and ending at 00F0.

DUMP SCREEN CONTENTS TO PRINTER

Pressing the "*" key while TASMON is waiting for keyboard input (except when a file name is being entered) will cause the current screen display to be sent to the printer. If the printer is not ready at the time the "*" is pressed, nothing is printed and control returns to TASMON. Graphics characters are printed as periods.

SUM/SUBTRACT HEX VALUES

This command will either sum or subtract two four digit hex values. Press "S" and two values followed by a "+" for sum or a "-" for subtract. The second value is added to or subtracted from the first. For example:

```
S 0100 8023 + 8123
S EC00 0100 - EB00
```

FIND CONSECUTIVE BYTES IN MEMORY

The find command will locate positions in memory where from 1 to 4 user specified two digit hex digits occur.

To run the command press "F", the starting address of the search and from 1 to 4 two digit search bytes. If less than 4 bytes are input, the ENTER key must be depressed to start the search. Pressing "F" followed by ENTER will find the next occurence of the last search key entered.

The address where the bytes were found is printed after the command line. If no value is printed, there were no more occurences of the search key in memory. For example:

```
F 0000 AF 54 <ENTER> 4176
```

Find where AF 54 resides in memory starting at 0000. First occurence was at 4176.

```
F <ENTER> 87FE
```

Find next occurence of AF 54. Found to be at 87FE.

```
F <ENTER>
```

No value was printed so there were no more occurences.

NOTE: The FIND command will always locate at least one occurence of the search key since the search key is stored in TASMON.

ZERO A BLOCK OF MEMORY

The "Z" or ZERO MEMORY command is used to set a block of memory to some value. To execute ZERO MEMORY, press the "Z" key, a starting address, an ending address and a two digit hex value to be written into the block. For example:

`Z F000 F050 54` will set memory from F000 through F050 to 54.

`Z F000 F050 00` will set memory from F000 through F050 to 00.

SKIP OR BACK UP ONE INSTRUCTION

To move the user's PC register to the next instruction without executing the current instruction press the RIGHT ARROW key. To move back to the previous instruction press the LEFT ARROW key.

These commands allow an instruction to be easily repeated or skipped. For example:

If the user's PC register holds 8000 and the following code is in memory:

```
7FFD 2110F0  LD      HL,F010
8000 110000  LD      DE,0000
8003 C38392  JP      9283
```

Pressing the LEFT ARROW would move the PC register back one instruction or to 7FFD. Pressing the RIGHT ARROW would skip the instruction at 8000 and move PC to 8003.

USER ROUTINES

This command is undefined by TASMON. It allows the user to define a routine to be executed by pressing the "U" key. If the "U" key is pressed without a user routine present nothing happens. To put a user routine in place, TASMON must be changed via the MODIFY MEMORY command so it will jump to the routine. The first step is to find where in memory to modify. TASMON checks for commands with the following type of code:

```
     CP      'U'
     JP      Z,ADDRESS
```

To patch in a user routine the address at "ADDRESS" must be changed to the entry address of the user's routine. To find where to modify enter the following, replacing the "E000" with the starting address of your working copy of TASMON:

```
F E000 FE 55 CA <ENTER> E085
```

The FIND command just found the first occurence of the menu select routine for the "U" key. The E000 address should be substituted with the starting address of TASMON (E000 in this case). The FE 55 is a "CP 'U'" Z-80 instruction, and the CA is the first byte of the "JP Z,ADDRESS" instruction.

To patch the user routine in place, MODIFY MEMORY in hex at three plus the address returned by FIND (this is the jump address). Now type in the entry address of the user routine in Z-80 format (LSB first, MSB last).

The patched version of TASMON can be written to disk. Refer to the WRITE command discussed below for instructions on how to do so.

To return from the user routine to TASMON simply do a Z-80 "RET" instruction (assuming the stack pointer has not changed).

The USER function will be supported by various routines in the future.

CLEAR SCREEN

The clear screen command will clear the video display and redisplay the Z-80 registers. To execute this command press the SHIFT-CLEAR key.

RELOCATE AND MOVE MEMORY

The RELOCATE command allows a machine language program to be moved from one location to another. All necessary jumps and loads within the range of relocation are changed. This command can be used to move TASMON from one location to another. RELOCATE can move many other machine language programs to new execution addresses.

To RELOCATE memory, press an "X" followed by the starting point of the move, the ending point of the move and the starting address of where the code is to be moved to. RELOCATION takes about 3 seconds per 4K of memory moved.

Suppose a program was loaded in memory from 8000 to 9FFF and we want to move it to E000 to FFFF. The command flow would go like this:

```
X 8000 9FFF E000 RELOCATE from 8000 to 9FFF and move it to E000
```

NOTE: The RELOCATE command will function correctly if code is overlapped. However, it will not allow TASMON to be overlapped while relocating.

For example, if a program resides from 8000 to 9FFF and is relocated to a new starting address of 9000, the relocated code will reside from 9000-AFFF. The relocated version overlaps the origin memory block of 8000 to 9FFF. This type of relocation will work with all programs except TASMON.

A problem can occur when relocating. For example, suppose the following code was in memory:

```
8000 210080    LD     BC,8000H
8003 CD6000    CALL   STALL
```

Suppose we relocated memory from 8000 through 80FF to E000. The code at E000 would appear as follows:

```
E000 2100E0    LD     BC,0E000H
E003 CD6000    CALL   STALL
```

If 8000 was a pointer to a text message, the change from 8000 to E000 would be correct, but in this case the 8000 was a stall value since the call to STALL is a delay routine. The change from 8000 to E000 in effect doubles this stall.

There are other occurences of this type. Another is when a register pair is loaded with, for example, the number of bytes to read from a disk file. If this number is changed the results could be disastrous.

Even with these two potential problems, RELOCATE does function with most programs.

MOVE A BLOCK OF MEMORY

To MOVE a block of memory from one location to another use the "Y" command. The command parameters are the same as for the RELOCATE (starting address, ending address and new starting address) command. This command simply copies memory from one location to another. The move routine is "smart" enough to allow code to overlap. For example:

```
Y 6000 6035 5000
```

Move memory from 6000 through to 6035 to 5000.

INPUT/OUTPUT

The author of TASMON chose to make the program's disk I/O file oriented rather than sector oriented as most other monitors. This allows a disk file to be loaded into RAM and then written back out.

VIEW A FILE

The VIEW command is similiar to the LOAD command in that it returns the starting, ending and transfer addresses of a disk file, except the VIEW command does not load the file into memory.

To execute the VIEW command press "V". A file name must be entered. For example:

```
V
CHESS/CMD
7000 8FA3 7535
```

The file "CHESS/CMD" was VIEWed from disk. The starting, ending and transfer addresses were found to be 7000, 8FA3 and 7535 respectively. Memory from 7000 to 8FA3 was not modified however.

NOTE: it is good practice to VIEW a file before LOADing it to verify the module will not load over TASMON.

LOAD A FILE

To LOAD a CMD file from disk press the "L" key, the ENTER key or a load offset, and a filename. For example:

```
L <ENTER>   Load the file TEST/CMD into memory from disk
TEST/CMD
```

After a module is loaded, the starting, ending and transfer addresses are displayed in that order. A typical load display would be:

```
L <ENTER>
MYPROG/CMD <ENTER>
F000 F035 F010
```

The starting address of the program is F000, the ending address is F035 and the transfer address is F010.

WRITE A FILE

To write a file out press "W" (for WRITE) followed by the starting, ending and entry addresses. Lastly, the filename is entered. When entering a file name the SHIFT BACKSPACE does not function. The BACKSPACE must be repeatedly pressed or held down to get to the beginning of the line.

If the above block move example was to be written to disk the following would be keyed in:

```
W 6350 6C08 6BFB
FILE/CMD
```

Write to disk starting at 6350, ending at 6C08 with an entry of 6BFB. Use the file name "FILE/CMD"

DISASSEMBLE TO DISK

The OUTPUT command will disassemble to disk as an Editor/Assembler source file. The code sent to disk is also echoed on the screen. To execute this command press the "O" key (for OUTPUT) and the starting, ending and transfer addresses of the dump. A filename is also entered.

A symbol table is generated by TASMON to ease the reading of the dump. The symbols are created for all 16 bit addresses between the starting and ending addresses specified. This table starts at the high memory pointer and builds downward in memory. If there is a program running in high memory make sure this pointer is set to such a value that the program will be protected. If TASMON is moved to high memory there will be about 100 bytes free for the symbol table. There are basically two ways to increase the symbol table size:

1) Move TASMON lower in memory.
2) Set HIGH$ to the address immediately below TASMON.

Of the two, the second is usually the preferred technique.

The symbol table uses two bytes per label. If large amounts of memory are being disassembled, there could be a pause of several seconds while the symbol table is being generated.

The starting address given will be used as the address of the ORG pseudo-op. The ending address is simply where output will halt. The transfer address is the address placed on the END pseudo-op.

Any text messages dumped to disk will be sent as Z-80 instructions. Therefore, some work may be required by the user to generate the proper source code in this case.

The source is written out with line numbers of 00000. Therefore, the first command executed from Editor/Assembler after the source has been loaded in would be RENUMBER (i.e. N 100,10).

The command format goes as follows:

O F000 F035 F010 TEST/ASM	Output to disk starting at F000, ending at F035, and entry address of F010. Use the file name "TEST/ASM"

The symbols TASMON generates are simply the address in question preceeded by a "Z". For example, a typical label would be:

```
Z0046H CALL    Z002BH
```

Bad symbols can be generated in some instances where text messages and stall or counter values are used. For example, if the following code was in memory:

```
8000 21
8001 00
8002 1F
8003 10
8004 FD
```

The bytes at 8000 and 8001 could be the last two bytes of a text message. The instruction at 8002 is a RRA. The instruction at 8003 is a DJNZ and the offset at 8004 refers back to 8002. However, when this code is disassembled out it would appear as follows:

```
        LD      HL,1F00H
        DJNZ    Z8002H
```

The symbol "Z8002H" is never defined since the instruction at 8002 was incorrectly disassembled as the most significant byte of the "LD HL,nn" instruction at 8000. The solution for this problem is to change the symbol "Z8002H" to the address "8002H". The source code will still appear incorrect but reassembling the source will give correct results.

NOTE: if a disk error ever occurs with TASMON, an error message and the TRSDOS error code (in hex) is printed. Refer to appendix A at the end of this manual for a list of error messages.

BREAKPOINTS

TASMON gives the user control over 9 breakpoints. A breakpoint allows a machine language program to be stopped at a predetermined spot and transfer control back to TASMON. For example, if a breakpoint was set at 8000 and the user's program executed the instruction at this address, control would be returned to TASMON.

Breakpoints are labeled 1-9. A three byte breakpoint (CALL nn) is used to intercept the user's program.

One unique feature of TASMON is that the number of times a breakpoint is executed before halting may be set for each breakpoint.

SET AND DISPLAY BREAKPOINTS

To set a breakpoint press "B" followed by the breakpoint number (1-9) and a four digit value. Breakpoints may be placed anywhere in memory. TASMON sets a breakpoint to 0000 in order to clear it.

To display the breakpoints press "B" and hit ENTER. Three rows of three sets of 4 and 2 digit hex values will be printed. These correspond to the values and number of executions for breakpoints 1, 2, 3, etc. For example:

B 8 809E sets breakpoint 8 to 809E

B <ENTER> displays all breakpoints

```
41F3 01 0000 01 0000 01
7802 01 0000 01 0000 01
0000 01 809E 18 0000 01
```

Breakpoint 1 is set at 41F3 and the execution number is 1,
Breakpoint 4 is set at 7802 and the execution number is 1,

Breakpoint 8 is set at 809E and the execution number is 18H, and all others are cleared.

NOTE: Care should be taken so that breakpoints do not overlap. For example, breakpoints must differ in address by at least three to function correctly. Suppose a breakpoint is set at 8000 and another at 8001. They will not function correctly since the three byte breakpoints will overlap (8000-8002 and 8001-8003):

	Brkpnt 1	Brkpnt 2
8000	CALL	
8001	lsb	CALL
8002	msb	lsb
8003		msb

NUMBER OF EXECUTIONS BEFORE BREAK

The "N" or "Number of executions before break" command allows setting the number of times a breakpoint is executed before the breakpoint is acknowledged. The default value is 01. This means execution will halt if the breakpoint is executed 1 time.

The formats of the command are:

N n h   Set the number of executions for breakpoint n to "h" (a value from 00-FF where 00 is 256 decimal).

N I   Set the number of executions for all breakpoints to 01 (or the normal number of executions).

N <ENTER> Set all breakpoints back to their set values. This value will be 01 unless changed by the "N n h" command.

The number of executions value is used only by the TRACE and GO commands (both discussed below), not by the single steppers. The value is decremented each time the breakpoint is executed. When this value reaches zero, execution halts and all execution numbers are reset to their original values (01 unless changed by the "N n h" command). The "N <ENTER>" command will also reset the values.

Most users probably will not use this command. If the execution number is left at 01, breakpoints will function as with any other monitor program.

CLEAR BREAKPOINTS

To clear a single breakpoint press "C" followed by the breakpoint number (1-9). To clear all breakpoints type "C" followed by ENTER. For example:

C 1   will clear breakpoint 1.

C <ENTER>   will clear all breakpoints.

INSTRUCTION STEP COMMANDS

There are two types of step commands in TASMON, manual and automatic. Each will start at the location pointed to by the user's PC register and return control to TASMON and display the registers. The PC register should contain the execute address of the user's program.

SINGLE STEP

There are two types of single steppers in TASMON:

1) step next instruction with CALLs executed in full.
2) step next instruction with CALLs stepped through.

The first type of single stepper will execute one instruction with CALLs executed in one step. To execute this command hit the DOWN ARROW key. The user's registers will be redisplayed upon return to TASMON. If a breakpoint is set within a CALL executed with this stepper, the CALL will be executed only up to the point of the break.

The second type of single stepper will execute one instruction with CALLs stepped through one instruction at a time. This command is executed by pressing the "I" key.

SINGLE STEPPING RESTARTS

The Z-80 "RST" command is a special single byte CALL. RESTARTS may be "stepped through" or "executed in full." The DOWN ARROW and "I" keys are still used to step restarts, except the "J" or JUMP THROUGH RESTARTS command is used to determine how they are handled. If pressing the "J" key displays a DOWN ARROW, restarts will be executed in full. If pressing the "J" key displays an "I", restarts will be stepped through. For example:

J I	Step through restarts mode is on
J !	Execute restarts in full mode is on

The status of restart stepping has no effect on how CALLs are handled. For example, CALLs can be stepped through while restarts are executed in full.

TRACE COMMANDS

The Trace command will continuously single step the user's program and redisplay his registers. To invoke this command press the "T" key. Next, enter the type of stepping desired. A

DOWN ARROW is used to execute CALLs in full and an "I" for step through CALLs. For example:

T I starts TRACE with calls stepped through

The step rate can be varied from about 2 seconds per instruction to 15 instructions per second by pressing the 0-7 keys while TRACE is executing (7 is the fastest step rate). Every time TRACE is entered the step rate is reset to one instruction per second.

Trace execution is halted by one of four ways:

1) One of the 9 user breakpoints is hit and the execution number is decremented to zero.
2) The BREAK key is depressed (control returns to command mode).
3) The SPACE BAR is depressed (execution pauses until the SPACE BAR is depressed again).
4) A "RET" instruction was executed while the "RETURN BREAKPOINT" option was on.

At times the user starts stepping through a CALL. When all the information needed is found, all the user wants to do is get out of the call. The "RETURN BREAKPOINT" option is a way of getting out of the CALL quickly. By pressing the "R" key while tracing, the "RETURN BREAKPOINT" option is turned on. When this option is on, the next Z-80 "RET" or "RET cc" where the condition was met will halt TRACE execution. This option is like putting a "floating" breakpoint on "RET" instructions. The only way to turn this option off is to exit and reenter TRACE.

GO COMMAND

The GO command will start the user's program at full speed.

The only way to halt the user's program is for a breakpoint to be executed until the execution number is decremented to zero.

To use the GO command, press a "G" followed by either a hex value where execution is to start or the ENTER key (execution starts at the user's PC register). For example:

G 8000 will start execution at 8000
G <ENTER> will start execution at the PC register.

To continue on from a breakpoint with GO do either of the following:

1) Single step over the instruction where the break occured.
2) Clear the breakpoint where the break occured then use the GO command to continue on.

One of these two steps is required since GOing at a breakpoint address simply returns control to TASMON with none of the user's program executed. Single stepping over the instruction at the break address then allows the GO command to continue on normally until the next breakpoint is executed.

NOTE: More than one instruction may need to be single stepped since a breakpoint uses three bytes. If GO execution is resumed in the middle of a breakpoint results can be unpredictable.

If the number of executions for a breakpoint is set greater than one, the GO command will execute part of the user's program at full speed and single step part of it (single step enough of it to make sure execution does not resume in the middle of a breakpoint). The BREAK key may be depressed to halt execution while single stepping if desired.

NOTE: TASMON does not allow an illegal Z-80 opcode to be single stepped or traced. Bad code is disassembled as "DEFB h." To run this type of code, a breakpoint must be set after the instruction and the GO command used to step it if so desired.

KEEP SCREEN

TASMON uses columns 56 to 72 for its displays. However, some user programs may also use these locations. The "K" or "KEEP SCREEN" command may be used to save the screen before TASMON affects it. When the "KEEP SCREEN" option is enabled, the user's last screen will be redisplayed before single stepping, tracing or GOing. On return from one of the stepping commands the screen will be resaved. There are four formats of the "K" command as follows:

1)	K start address	save screen at "start address"
2)	K <ENTER>	display user's screen
3)	K Y	turn KEEP SCREEN on
4)	K N	turn KEEP SCREEN off

The first option, "K start address", is used to initialize the KEEP SCREEN command. The four digit value "start address" is the starting address of a 2048 byte buffer in memory where the user's screen is to be saved. When the location is entered the screen memory is set to a clear screen of 2048 spaces (20H). The ASCII option of the MODIFY MEMORY command may be used to set the screen to some initial condition.

The second option, "K <ENTER>", will bring the user's saved screen back to the video display and leave it there as long as the ENTER key is held down. This option allows for a quick review of the user's display.

The third option, "K Y", is used to turn the KEEP SCREEN option on. Whenever a program is stepped with this option on, the user's screen will be redisplayed and saved continuously. TASMON will not affect the user's screen at all.

The forth option, "K N", is used to turn the KEEP SCREEN option off. The current saved screen is not changed by turning the command off.

The screen buffer may be cleared by the "K start address" option or the ZERO MEMORY command. Example inputs are:

K DCOO	Set user's screen buffer at DCOO-FFFF and clear the buffer (make it all spaces).
K <ENTER>	Display the current saved screen.
K Y	Turn the KEEP SCREEN command on.
K N	Turn the KEEP SCREEN command off.

GENERAL COMMENTS

A commented listing of the source code is available from The Alternate Source for $30. The address is:

The Alternate Source
704 N. Pennsylvania Ave.
Lansing, Mi. 48906
(517) 482-8270

SINGLE STEPPING THROUGH BASIC

A powerful feature of TASMON is that the BASIC interpreter written by Microsoft may be single stepped.

This allows a BASIC program to be entered from the keyboard and RUN. TASMON will step through the routines of the BASIC interpreter to perform these tasks.

The first step is to get BASIC and TASMON co-resident in memory. This can be done a few ways:

Disk users can load TASMON from DOS and load BASIC by typing:
L <ENTER>
BASIC/CMD.BASIC
AAAA BBBB CCCC

Where AAAA, BBBB, and CCCC are the starting, ending, and execution addresses that TASMON reports for BASIC/CMD.

The next step is to set a breakpoint at 5920. This the address of the BASIC command mode. If this breakpoint is not set, any error from BASIC such as a SYNTAX or MISSING OPERAND error will cause TASMON to be exited.

If TASMON is ever exited in this manner simply re-enter the monitor by typing:

```
>DEF USR = &HE000 <ENTER>
>X = USR(0) <ENTER>
```

The state of the Z-80 registers will remain unchanged. Therefore, stepping can continue from where TASMON was exited.

RESTARTS must be set to "execute in full" mode. Press the "J" key until this mode is enabled. The "execute in full" mode is on when an "!" is displayed after the "J" pressed by the user.

Now BASIC can be either single stepped via the DOWN ARROW key or "I" key or TRACE mode.

If TRACE mode is selected with CALLs executed in full and the "7" speed option is selected (fastest TRACE step rate), BASIC will operate about 5000 times slower than normal.

If CALLs are stepped through, keyboard characters must be held down until the keyboard driver routine used by BASIC scans through them. After this there is a significant stall to eliminate keybounce. For these reasons CALLs executed in full is recommended for stepping BASIC.

The re-entry address of BASIC is 5920. Modify the PC register to this address before stepping BASIC as follow:

```
R PC 5920
```

After TASMON has been patched in, BASIC will function normally.

Some BASIC commands will not function correctly. For example, none of the disk input/output commands will function correctly.

The breakpoint at 5920 will be executed each time the ENTER key is pressed. This may be an irritation, but the breakpoint is required or stepping BASIC will not function correctly.

When the breakpoint at 5920 is executed, simply continue tracing or single stepping by pressing the appropriate command key(s). For example, to continue with TRACE mode type:

```
T !
```

Pressing the BREAK key will exit BASIC and return to TASMON. To continue stepping BASIC, simply continue tracing or single stepping by pressing the appropriate command keys.

Refer to SESSION 4 for more information and an example of single stepping a BASIC program.

SAMPLE SESSIONS

The following sample sessions are examples using TASMON's commands.

SESSION 1 - RELOCATE TASMON

The distributed version of TASMON loads from E000-FFFF with an entry point of E000. however, some owners of TASMON will probably want TASMON to run at low memory or 3000-4FFF. To do this enter the following commands:

From DOS enter TASMON by typing:

```
TASMON4
```

The Z-80 registers and user prompt will be displayed.

Next use the RELOCATE command to move the program to memory starting at 3000. The format is:

`X E000 FFFF 3000` which relocates memory from E000-FFFF to memory starting at 3000.

Now TASMON resides at E000-FFFF and at 3000-4FFF. To save the low memory version to disk use the WRITE command. The format is:

```
W 3000 4FFF 3000
LTASMON/CMD
```

Which dumps memory from 3000-4FFF with a transfer address of 3000 to disk with the file name "LTASMON/CMD".

Whenever "LTASMON" is typed in from DOS the low memory version of the program will be executed.

SESSION 2 - STEP AND TRACE A /CMD FILE

The short program used in this example appears as follows:

```
00100         DI                    ;ELIMINATE INTERRUPTS
00110         LD      C,28          ;HOME CURSOR CHAR
00120         LD      A,2           ;DISPLAY A CHAR CODE
00130         RST     28H           ;DO SVC
00140         LD      C,31          ;CLEAR TO EOF CHAR
00150         LD      A,2           ;DISPLAY CODE
00160         RST     28H           ;DO SVC
00170         LD      E,'K'         ;LOAD DE WITH *KI NAME
00180         LD      D,'I'
00190         LD      A,82          ;SET UP SVC #82
00200         RST     28H           ;DO SVC
00210         EX      DE,HL         ;GET RESULT IN DE
00220         LD      HL,BUFF       ;POINT TO BUFFER
00230         LD      A,99          ;CONVERT TO ASCII
00240         RST     28H           ;DO SVC
00250         LD      HL,KIMSG      ;POINT TO *KI MESSAGE
00260         CALL    DISPL         ;DISPLAY IT
00270         LD      E,'D'         ;LOAD DE WITH *DO NAME
00280         LD      D,'O'
00290         LD      A,82          ;SET UP SVC #82
00300         RST     28H           ;DO SVC
00310         EX      DE,HL         ;GET RESULT IN DE
00320         LD      HL,BUFF2      ;POINT TO BUFFER
00330         LD      A,99          ;CONVERT TO ASCII
00340         RST     28H           ;DO SVC
00350         LD      HL,DOMSG      ;POINT TO *DO MESSAGE
00360         CALL    DISPL         ;DISPLAY IT
00370         LD      E,'P'         ;LOAD DE WITH *PR NAME
00380         LD      D,'R'
00390         LD      A,82          ;SET UP FOR SVC #82
00400         RST     28H           ;DO SVC
00410         EX      DE,HL         ;GET RESULT IN DE
00420         LD      HL,BUFF3      ;POINT TO BUFFER
00430         LD      A,99          ;CONVERT TO ASCII
00440         RST     28H           ;DO SVC
00450         LD      HL,PRMSG      ;POINT TO *PR MESSAGE
00460         CALL    DISPL         ;DISPLAY IT
00470         LD      A,22          ;EXIT TO TRSDOS
00480         RST     28H           ;DO SVC - BYE, BYE!
00490 DISPL   LD      A,(HL)        ;GET THE CHAR AT (HL)
00500         INC     HL            ;BUMP POINTER
00510         OR      A             ;ZERO BYTE?
00520         RET     Z             ;RETURN IF SO
00530         LD      C,A           ;PUT CHAR IN C
00540         LD      A,2           ;DISPLAY A CHARACTER
00550         RST     28H           ;DO SVC
00560         JR      DISPL         ;GO BACK TO LOOP
00570 KIMSG   DEFM    'The keyboard DCB resides at: '
00580 BUFF    DEFM    'xxxxH'
```

```
00590        DEFW     000DH
00600 DOMSG  DEFM     'The video DCB resides at: '
00610 BUFF2  DEFM     'xxxxH'
00620        DEFW     000DH
00630 PRMSG  DEFM     'The printer DCB resides at: '
00640 BUFF3  DEFM     'xxxxH'
00650        DEFW     000DH
00660        END      3000H
```

The purpose of this program is to find the Device Control Blocks for the major devices (keyboard, video, and printer) and display them on the screen. For this session TASMON is assumed to be in memory from E000-FFFF and the short program given above is saved on disk under the file name "TEST/CMD". The distributed copy of TASMON has both of these files on the master diskette with the indicated load addresses.

The first step is to load the file into memory. The LOAD command is used for this by keying in:

```
L <ENTER>
TEST/CMD
3000 30BB 3000
```

The file "TEST/CMD" loaded from 3000-30BB with an entry point of 3000.

The first time through the program we will simply single step it.

The first step is to load the PC register with the starting address of the program or 3000. Use the REPLACE command to do this:

```
R PC 3000
```

To aid in viewing the program, disassemble the program to the screen. This is done by entering:

```
D 3000
```

The first 23 instructions of the program will be displayed on the left side of the screen. Now hit the BREAK key to get back to command mode. The disassembled code and the Z-80 registers will be displayed. The screen should appear as follows:

```
3000 F3       DI                     DI
3001 0E1C     LD      C,1C           IX   4C41   IY   094C
3003 3E02     LD      A,02           AF'  4B43   BC'  4353
3005 EF       RST     28             DE'  AA52   HL'  0B0A
3006 0E1F     LD      C,1F           AF   00FF   BC   4C44
3008 3E02     LD      A,02           DE   4C48   HL   A070
300A EF       RST     28H            SP   41E4   PC   5F00
300B 1649     LD      D,49           SZ1H1PNC   (HL)   4C
300D 1E4B     LD      E,4B           -
```

```
300F 3E52        LD       A,52
3011 EF          RST      28
3012 EB          EX       DE,HL
3013 217130      LD       HL,3071
3016 3E63        LD       A,63
3018 EF          RST      28H
3019 215430      LD       HL,3054
301C CD4A30      CALL     304A
301F 1E44        LD       E,44
3021 164F        LD       D,4F
3023 3E52        LD       A,52
3025 EF          RST      28H
3026 EB          EX       DE,HL
3027 219230      LD       HL,3092
```

Notice that the labels used in the source code have been changed to actual addresses, and the text message appears as Z-80 instructions.

Hit the BREAK key to exit the DISASSEMBLE mode and reenter TASMON's command mode.

To single step the instruction at the PC register or 3000 (which is a DI) hit the DOWN ARROW or "I" key. The PC equals 3001 and the instruction at 3001 (or PC) is LD C,1C.

Single step this instruction. The C register will hold 1C or an ASCII "HOME CURSOR". PC will now be 3003. The next instruction is "LD A,02". Single step PC again. A equals 02 or the SVC code to display a character. PC now holds 3005. The instruction there is a RST 28H. This instruction is the SVC, or Supervisor Call. Remember the "J" key? Press it until you see a backslash. This is to make sure that the RST will be executed in full. Single stepping through it could be disastrous for reasons that are explained in the Technical Appendix. In the meantime, execute this instruction in full.

The next instruction is LD C,1F which is the ASCII code to clear to the end of the frame (screen).

The next instruction is LD A,02. Again, this is the SVC code to display a character on the screen. Step this instruction also.

The next instruction is the RST 28H. Yet again, this is the actual Supervisor Call instruction. Execute it by hitting either "I" or "!".

The screen should have cleared. TASMON's register display will still be intact, since TASMON updates the register display whenever necessary.

The current instruction should be "LD E,4B". You can verify this by looking above TASMON's register display. The LD E,4B should appear there. Single step this instruction.

The next instruction is LD D,49. Single step this instruction also.

The next instruction is LD A,52. This is the code for the SVC to find the DCB of the device named in DE (which at the moment is device *KI, or the keyboard. 49H is a "K" and 4BH is an "I".

The next instruction is the SVC to find the DCB. Execute it in full.

HL should now contain the DCB address. The next instruction, EX DE,HL should move the address to the DE register pair so that we can convert the hex value to an ASCII string with the next SVC. Step the EX DE,HL.

Now we see LD HL,3071 which points HL to the buffer where we want to store the result of the ASCII conversion of DE. Step the instruction.

Next we have LD A,63 which is the code for the hex to ASCII convert SVC. Step this.

Here is the SVC. Execute it in full.

Just to see if it worked, type:

A 3071 <ENTER>

You should see the string of characters that represents the number that was in the HL register pair after the third SVC (the one that found the address in the first place.) If so, all is well.

Hit <BREAK> to get back to TASMON's command mode. The next instruction is LD HL,3054. This points HL to the beginning of a message that preceedes the actual value of the address. Step this instruction.

Next we have a CALL 304A. For the moment, let's take a close look at this subroutine by pressing the "I" key to single s .

Notice that the PC is now at 304A, which is the address that was CALLed. The instruction is LD A,(HL). Remember that HL is pointing to a message. Step this instruction.

Now we see an INC HL instruction. This simply moves HL to the next character in the string. Step this one, too.

Now we have an OR A. ORing A against itself like this is an easy way to see if A contains a zero. Step this instruction.

Look at the register display. The line of dashes and letters has been changing throughout this session, but only now is it really important, because the instruction here is RET Z. Is there a Z in the line of dashes? No, there isn't. That's because the contents of A (the first character of the message, in other words) is not equal to zero. Step this instruction.

It didn't return, did it? The current instruction is LD C,A. This is necessary because the SVC to display a character needs the character to be in the C register. Step this instruction.

The next one is the (by now) familiar LD A,02 instruction, which is the command to display a character. Step this instruction.

Now we have the SVC. Execute this in full. An uppercase "T" should appear in the upper left hand corner of the screen.

The next instruction is a JR 304A. Step this.

We're back at the beginning of the subroutine! We already know what the subroutine looks like, so let's trace. Type T ! to start tracing.

We want the tracing to stop as soon as the subroutine is done, so press the "R" key. This tells TASMON that as soon as it is about to execute a RET instruction to stop tracing and go back to command mode.

The tracing can be sped up by pressing a key from 1 to 7 if the trace is too slow for your taste.

The tracing will stop with the current instruction being the RET Z that we saw earlier. The fact that TASMON has stopped tracing means that the condition has been met, and the RET is about to be executed. Step through the RET.

The next instruction is LD E,44. This is the first LD in the next device search setup. We now know how TEST/CMD goes about the setup process, so let's let the program do the work now and just look at the results.

The best way to do this is with a breakpoint. Before setting a breakpoint, though, it helps to know where to set the breakpoint! So, type D <ENTER> to start disassembling to the screen from the current instruction.

You should see some instructions that are identical to the ones that we just executed, except that the LD's to E and D are different and the buffer addresses are different. At address 3033 there is another set of practically identical instructions. Since the instructions seem to be following a logical pattern (which, indeed, they are) press <BREAK> to return to the command mode and type:

```
B 1 3033
```

which will set breakpoint #1 at 3033H. The current instruction is still LD E,44 and the PC contains 301F. Now type:

```
G <ENTER>
```

Practically instantly you should have seen the computer print a message telling you where the video DCB is. That's because when you typed in the G command TASMON actually gave control to the TEST/CMD program briefly, allowing it to run at full speed. It then ran into the breakpoint at 3033, which is what the PC should contain now. The current instruction should be LD E,50.

Now we want to remove the breakpoint, since we won't be using it again. Type:

```
C 1
```

This will remove breakpoint #1. Remember that you can remove ALL breakpoints by typing:

```
C <ENTER>
```

Now we just want to finish executing the program, so type:

```
G <ENTER>
```

which is what you typed after you set the breakpoint. The program will take over again, but since there is no breakpoint for it to hit, it will continue running and return you to TRSDOS Ready. You will have to reload TASMON to continue.

SESSION 3 - DISASSEMBLE PROGRAM TO DISK

The "O" or OUTPUT command is used to accomplish this task.

The first step is to load "TEST/CMD" into memory by entering:

```
L <ENTER>
TEST/CMD
3000 30BB 3000
```

Next, enter the OUTPUT command as follows:

```
O 3000 30BB 3000
TEST/ASM
```

The disassembly will be written out to disk with the file name TEST/ASM starting at 3000 and ending at 30BB with a transfer address of 3000. Now exit TASMON by keying in:

```
E <ENTER>
```

The system will reenter DOS. Suppose you have an Editor/Assembler. Enter the E/A by typing its file name from DOS.

Next, load TEST/ASM with the "LD" command of Editor/Assembler (or similiar command if using a different E/A). As stated previously under the explanation of the OUPUT command, the first command to enter is a RENUMBER command (assuming that your E/A requires line numbers - ALE doesn't, ZEUS renumbers automatically, and others work still differently.) TASMON writes out the file with line numbers of 00000 so this command may be required. To do this enter:

`N 100,10` which renumbers the program in increments of 10 with a starting line number of 100, assuming that you are using Radio Shack's Editor/Assembler.

The source listing should be:

```
00100          ORG      3000H
00110          DI
00120          LD       C,1CH
00130          LD       A,02H
00140          RST      28H
00150          LD       C,1FH
00160          LD       A,02H
00170          RST      28H
00180          LD       E,4BH
00190          LD       D,49H
00200          LD       A,52H
00210          RST      28H
00220          EX       DE,HL
```

```
00230          LD      HL,Z3071H
00240          LD      A,63H
00250          RST     28H
00260          LD      HL,Z3054H
00270          CALL    Z304AH
00280          LD      E,44H
00290          LD      D,4FH
00300          LD      A,52H
00310          RST     28H
00320          EX      DE,HL
00330          LD      HL,Z3092H
00340          LD      A,63H
00350          RST     28H
00360          LD      HL,Z3078H
00370          CALL    Z304AH
00380          LD      E,50H
00390          LD      D,52H
00400          LD      A,52H
00410          RST     28H
00420          EX      DE,HL
00430          LD      HL,Z30B5H
00440          LD      A,63H
00450          RST     28H
00460          LD      HL,Z3099H
00470          CALL    Z304AH
00480          LD      A,16H
00490          RST     28H
00500 Z304AH   LD      A,(HL)
00510          INC     HL
00520          OR      A
00530          RET     Z
00540          LD      C,A
00550          LD      A,02H
00560          RST     28H
00570          JR      Z304AH
00580 Z3054H   LD      D,H
00590          LD      L,B
00600          LD      H,L
00610          JR      NZ,30C4H
00620          LD      H,L
00630          LD      A,C
00640          LD      H,D
00650          LD      L,A
00660          LD      H,C
00670          LD      (HL),D
00680          LD      H,H
00690          JR      NZ,Z30A6H
00700          LD      B,E
00710          LD      B,D
00720          JR      NZ,30D8H
00730          LD      H,L
00740          LD      (HL),E
00750          LD      L,C
00760          LD      H,H
```

```
00770          LD     H,L
00780          LD     (HL),E
00790          JR     NZ,30CFH
00800          LD     (HL),H
00810          LD     A,(7820H)
00820          LD     A,B
00830          LD     A,B
00840          LD     A,B
00850          LD     C,B
00860          DEC    C
00870          NOP
00880 Z3078H   LD     D,H
00890          LD     L,B
00900          LD     H,L
00910          JR     NZ,30F3H
00920          LD     L,C
00930          LD     H,H
00940          LD     H,L
00950          LD     L,A
00960          JR     NZ,30C7H
00970          LD     B,E
00980          LD     B,D
00990          JR     NZ,30F9H
01000          LD     H,L
01010          LD     (HL),E
01020          LD     L,C
01030          LD     H,H
01040          LD     H,L
01050          LD     (HL),E
01060          JR     NZ,30F0H
01070          LD     (HL),H
01080          LD     A,(7820H)
01090          LD     A,B
01100          LD     A,B
01110          LD     A,B
01120          LD     C,B
01130          DEC    C
01140          NOP
01150 Z3099H   LD     D,H
01160          LD     L,B
01170          LD     H,L
01180          JR     NZ,310EH
01190          LD     (HL),D
01200          LD     L,C
01210          LD     L,(HL)
01220          LD     (HL),H
01230          LD     H,L
01240          LD     (HL),D
01250          JR     NZ,30EAH
01260          LD     B,E
01270          LD     B,D
01280          JR     NZ,311CH
01290          LD     H,L
01300          LD     (HL),E
```

```
01310          LD       L,C
01320          LD       H,H
01330          LD       H,L
01340          LD       (HL),E
01350          JR       NZ,3113H
01360          LD       (HL),H
01370          LD       A,(7820H)
01380          LD       A,B
01390          LD       A,B
01400          LD       A,B
01410          LD       C,B
01420          DEC      C
01430          NOP
01440          END      3000H
```

Notice that the source code here is the same at the original source code of "TEST/CMD" except that the labels are different and the text message now appears as Z-80 instructions. Text messages are generally easy to convert from Z-80 instructions back to text. This is done by converting the instructions to numbers. Anyone who has hand assembled a program has done this. The only problem exists when spaces are present in the text. The code for a space is 20H, which also happens to be the Z-80 instruction for a "JR NZ,e". The problem does not exist in finding the space, but in finding the character after the space. The character after the space is the index of the relative jump minus two.

To determine the character after the space (or JR NZ) at line 00610 do the following:

Start counting instructions starting at the last known address. In this case the last know address is 3056 (or Z3056H - TASMON simply puts a "Z" in front of the address when making it a label). By doing this it is determined that the address of the JR NZ,30C4H instruction in line 00610 is 3057. We add one to the last known address because instructions such as "LD D,H" are only one byte long. however, if a "JR NZ,e" instruction is encountered, two must be added to the address since this instruction is two bytes long.

Now subtract 3057 from 30C4 or more generally, subtract the address of the jump instruction from the destination of the jump. The result of this subtraction in our case is 6DH.

Now subtract two more from this value. This subraction is necessary since the index of a relative jump is stored in memory as the index minus two. Subtracting two from 6DH gives 6BH, which is an ASCII "k". See line 570 of the first program listing in session 2.

The instructions such as "LD C,B" must be converted back to ASCII by refering to the Z-80 instruction tables in a book such as Radio Shack's TRS-80 ASSEMBLY LANGUAGE PROGRAMMING.

It is also important to note that some labels were not generated because the address that they reference contains the second or third byte of an instruction, in this case a LD A,(7820H) instruction. To resolve these references, put the label in front of the LD A,(7820H) and change the reference from "label" to "label+2". This will make the reference refer to the correct address, even though the label is off.

An easier way to fix messages is to view the program with an ASCII dump from TASMON and record the addresses of the text messages. If a printer is available, pressing the "*" key will dump the screen contents to the printer thus giving a hardcopy listing of the ASCII dump.

SESSION 4 - TRACE A BASIC PROGRAM

In this example TASMON must reside in memory from E000-FFFF. It must also be protected, so from TRSDOS type:

```
MEMORY (HIGH = X'DFFF') <ENTER>
```

Now load TASMON from disk. If TASMON is not already located at E000-FFFF, move it there with the X command as described earlier. Now set RESTARTS to "executed in full" mode by pressing the "J" key until you see the backslash:

```
J !
```

Next load BASIC by typing:

```
L <ENTER>
BASIC/CMD.BASIC
AAAA BBBB CCCC
```

AAAA, BBBB, and CCCC are the starting, ending, and execution addresses, respectively, of BASIC. Now set the PC to CCCC by typing:

```
R PC CCCC
```

and start the TRACE by typing:

```
T !
```

The initialization routine for BASIC is now being traced. The speed of initialization can be sped up by pressing the "7" key.

After a long initialization process, the READY message will appear. We are now tracing through BASIC. Enter the following program:

```
10 PRINT "START"
20 FOR I = 1 TO 5
30 PRINT I; I/2; I*2
40 NEXT I
50 PRINT "DONE"
60 END
```

Now type:

```
LIST
```

The BASIC program should list upon the screen. Notice that TASMON is continually redisplaying the registers. This short program may even be RUN from TASMON's TRACE mode.

If a BASIC error occurs, TASMON will be exited completely. To fix this condition a breakpoint must be set at 5920. Do this by entering:

```
B 1 5920
```

To exit BASIC and return to TASMON press the BREAK key. This must be done before any TASMON command may be entered.

The breakpoint at 5920 will occasionally cause TASMON to be reentered. To continue stepping BASIC simply restart tracing as follows:

```
T !
```

If a BASIC program being run is to be halted and control returned to the BASIC command mode, press the BREAK key and change the PC register to 5920 as follows:

```
R PC 5920
```

Then continue tracing.

Let's start with a fresh screen by pressing the CLEAR key.

Now start tracing BASIC if not already doing so.

List the program again by typing:

```
LIST
```

The program should list on the screen.

To RUN the program type:

```
RUN
```

The message "START" will be printed on the screen followed by five rows of three numbers and the "END" message.

APPENDIX A - DOS ERROR MESSAGES

Error number	Error description
00	No error
01	Parity error during header read
02	Seek error during read
03	Lost data during read
04	Parity error during read
05	Data record not found during read
06	Attempt to read system data record
07	Attempt to read system data record
08	Device not available
09	Parity error during header write
0A	Seek error during write
0B	Lost data during write
0C	Parity error during write
0D	Data record not found during write
0E	Write fault on disk drive
0F	Write protected diskette
10	Illegal logical file number (bad DCB)
11	Directory read error
12	Directory write error
13	Illegal file name (bad DCB)
14	GAT read error
15	GAT write error
16	HIT read error
17	HIT write error
18	File not in directory
19	File access denied
1A	Directory space full
1B	Disk space full
1C	EOF encountered
1D	NRF out of file range
1E	Full directory
1F	Program not found
20	Illegal drive number
21	No device space available
22	Load file format error
23	Memory fault
24	Attempt to load to ROM
25	Illegal access attempted
26	File has not been opened
27-3E	Not defined
3F	Unknown error code

APPENDIX B - TASMON COMMAND SUMMARY

This notation is used in the command summary

HH = 4 digit hex value
SS = 4 digit hex starting point
EE = 4 digit hex ending point
TT = 4 digit hex transfer point
n = Single digit from 1 to 9
h = 2 digit hex value

A SS	ASCII dump of memory starting at SS.
B n HH	Set breakpoint n at HH.
B <ENTER>	Display the breakpoints.
C n	Clear breakpoint n.
C <ENTER>	Clear all breakpoints.
D SS	Disassemble memory starting at SS.
E <ENTER>	Exit TASMON and return to DOS or BASIC
F SS h h h h	Find search key h h h h starting at SS.
G HH	Start execution at HH.
G <ENTER>	Start execution at user's PC.
H SS	Dump memory in hex starting at SS.
I	Single step - CALLs stepped through.
J (I or !)	Toggle RESTARTS between stepped through and execute in full.
K SS	Set user's screen buffer at SS and clear the screen buffer.
K <ENTER>	Display the user's screen for as long as the ENTER key is held down.
K Y	Turn the KEEP SCREEN command on.
K N	Turn the KEEP SCREEN command off.

L <offset>	Load in CMD disk file named "file" with an optional offset.
M H SS	Modify memory in hex mode starting at SS.
M A SS	Modify memory in ASCII mode starting at SS.
N n h	Set number of executions for breakpoint n to h.
N I	Initialize all execution numbers to 01.
N <ENTER>	Reset all execution numbers to their default values.
O SS EE TT	Output disassembled listing starting at SS, ending at EE with a transfer address of TT to disk with the file name "file".
P SS EE	Disassemble to the printer starting at SS and ending at EE.
R rp HH	Replace register pair "rp" with HH.
S H1 H2 +	Add H2 to H1.
S H1 H2 -	Subtract H2 from H1.
T I	Trace through a program with CALLs stepped through.
T !	Trace through a program with CALLs executed in full.
U	Go to user routine. Does nothing unless a routine is patched in.
V	View the disk file titled "file". Returns file starting, ending and transfer addresses without loading into memory.
W SS EE TT	Write a CMD disk file starting at SS, ending at EE with a transfer address of TT and file name of "file".
X SS EE TT	Relocate memory from SS to EE and place it in memory starting at TT.
Y SS EE TT	Block move from SS to EE and place in memory starting at TT.

Z SS EE h	Set memory from SS to EE equal to h.
RT ARROW	Skip current instruction in user's PC and point to next instruction.
LFT ARROW	Back up user's PC to the previous instruction.
*	Dump screen contents to the printer.
DN ARROW	1) Single step - CALLs executed in full 2) Display next line of a memory dump 3) Point to next byte when modifying memory
SHIFT-CLEAR KEY	Clear the screen and display the registers.
BREAK KEY	Return to command mode.

APPENDIX C - SAMPLE USER FUNCTIONS

This appendix will give an example of patching in a USER command (the "U" command). This routine will allow HARD COPY TRACING and DISPLAY THE LAST FIFTEEN EXECUTED INSTRUCTIONS.

HARD COPY TRACING is the same as normal tracing except the current PC address and Z-80 mnemonic are sent to the printer. If the printer is not on when "HARD COPY TRACE" is selected, nothing is printed and execution continues as if the TRACE command had been selected.

DISPLAY THE LAST FIFTEEN EXECUTED INSTRUCTIONS while tracing will display the user's PC and Z-80 mnemonic on TASMON's display lines.

This patched routine assumes that TASMON4 version 1.00 is being used. Also, TASMON should be located in memory starting at E000H. The following bytes are entered:

```
M H FD6F 6D
```

and enter the following bytes from there:

```
FD6F: 00 00 CD 32 E9 CD EC E0 FE 44 28 0B FE 48 28 46 FE 55
FD81: CA F7 E0 18 EE CD 32 E9 32 30 FE C3 43 E8 AF 32 2F FE
FD93: 32 30 FE C3 3E E0 3A 30 FE B7 28 22 CD 23 E9 2A 42 F8
FDA5: CD 37 F1 3E 20 CD 32 E9 21 38 00 06 11 E5 C5 06 01 3E
FDB7: 0F EF CD 94 F7 C1 E1 23 10 F1 3A E1 F8 C9 CD 32 E9 32
FDC9: 2F FE C3 43 E8 3A 2F FE B7 CA 03 E1 E5 D5 1E 50 16 52
FDDB: 3E 52 EF EB 0E 00 3E 05 EF D1 E1 C2 03 E1 2A 42 F8 7C
FDED: CD 15 FE 7D CD 15 FE 3E 20 CD A3 F7 CD 03 E1 06 14 21
FDFF: 38 00 E5 C5 06 01 3E 0F EF CD A3 F7 C1 E1 23 10 F1 3E
FE11: 0D C3 A3 F7 F5 CB 3F CB 3F CB 3F CB 3F CD 24 FE F1 E6
FE23  0F C6 30 FE 3A 38 02 C6 07 C3 A3 F7 00 00
```

MODIFY MEMORY in hex as follows:

```
M H E0E7   71 6C
E0E8 ' E0 FD
E0E9   C3 <BREAK>
```

```
M H E732   CE C9
E733 ' E1 FD
E734   D1 <BREAK>
```

```
M H E868 : 3A CD
E869   CC 99
E86A   F8 FD
E86B   B7 <BREAK>
```

NOTE: DO NOT hit the BREAK key to exit from this last memory modification until the correct values are in place. Failure to do this will probably cause a reset!

```
M H E101 4 3E 71
E102 ' E0 FD
E103 : 3A <BREAK>
```

To write the patched version of TASMON out under the file name "UPTASMON/CMD", enter the following command:

```
W E000 FFFF E000
UPTASMON/CMD <ENTER>
```

To execute the "HARD COPY TRACE" command, press the "U" key followed by the "H" key for HARD COPY TRACE. Next, enter the CALL stepping mode. This is an "I" for CALLs stepped through or a DOWN ARROW for CALLs executed in full.

TASMON will step through memory as it would with the TRACE command except the following type output is sent to the printer:

```
8000  LD      A,(37E8)
```

All TRACE command keys function with the "HARD COPY TRACE" patch.

The DISPLAY LAST FIFTEEN EXECUTED INSTRUCTIONS patch is executed by pressing the "U" key and the "D" key for DISPLAY LAST FIFTEEN EXECUTED INSTRUCTIONS. Next, enter the CALL stepping mode. This is an "I" for CALLs stepped through or a DOWN ARROW for CALLs executed in full.

After each instruction is executed, its address and Z-80 mnemonic are displayed on TASMON's command lines. Up to sixteen previously executed instructions will be displayed.

All TRACE command keys function with the "DISPLAY LAST FIFTEEN EXECUTED INSTRUCTIONS" routine.

This patch may be in a different location for either past or future versions of TASMON. Any future version may have these commands added to its repertoire.

If even more user routines are to be added, the address at FD7D-FD7E can be modified to the starting address of the new routine. To execute this routine press the "U" key to jump to this user patch and another "U" to jump to the new routine.

MODEL IV TASMON TECHNICAL NOTES

Model IV TASMON was written in such a way as to be as functionally compatible with the Model I/III versions as possible. There are a few quirks, though, that the Model IV user should be aware of.

First of all, tape support has been eliminated. The reason for this should be obvious.

Logical Systems, Inc., the authors of TRSDOS 6.x, obviously don't want people fooling around with the system, but rather than sacrifice the power that TASMON had in order to accede to LSI imperatives, we elected to ignore their warnings about tampering with the system at the hardware level and implement the same features that were in the Model I/III versions of TASMON.

Scrolling was a problem. TASMON scrolls the area directly underneath the register display while maintaining the display. The video control SVC's were simply not flexible enough to allow this to happen.

The "keep screen" feature was another tricky one. There ARE SVC's to move the screen to/from a 2K buffer, and I used those. The problem, then, was viewing the kept screen. Ideally, it had to be identical to the Model I/III technique in which the user simply typed "K" and held down the <ENTER> key for as long as they wanted to see the screen. In order to do this, though, I had to access the keyboard directly - another LSI no no. Again, rather than sacrifice the keep screen capability, I wrote the code to access the keyboard as necessary, i.e., directly, rather than through a SVC.

Since this code is very system dependent, and might have to change as TRSDOS 6.x changes (unlikely - it's not THAT system dependent!) we elected to put the code in an overlay.

Another reason for using the overlay is that the code there toggles the high RAM bank in and out, and if this code were in TASMON itself and TASMON were in the upper 32K of RAM, in the process of toggling high RAM back and forth, TASMON would toggle itself right out of existence! The solution to this problem was to make the code part of an overlay and to make it self-contained, i.e., no referrences to addresses within TASMON, and to have its own stack. If you examine the overlay code, you will see how this has been accomplished.

There is a drawback to using an overlay to do TASMON's scrolling. You cannot trace or single-step through any RST 28H that calls in a DOS overlay, since to do so would mean that the DOS would load its overlay; TASMON would load ITS overlay on the next call to the scrolling routine; the DOS, not being aware that its overlay was no longer in place, would try to execute TASMON's overlay; and massive amounts of chaos would exist. So, don't trace or single-step RST 28H unless you know for a fact that the RST will not load a DOS overlay.

Paul F. Snively

PATCHES TO CHANGE CERTAIN CHARACTERISTICS OF TASMON

These patches are optional and may be applied as desired to your working copy of TASMON. They may be applied in memory only, in which case they will remain in effect until the end of your sessi with TASMON, or you may use TASMON's W D command to write a copy of your patched TASMON disk.

PLEASE NOTE that these patches apply only to the following versions of TASMON, however information is given to allow you to find the proper location to apply the patch if you have another version.

MODEL I: VERSION 2.22 MODEL III: VERSION D7 MODEL 4: VERSION 1.11

PATCH # 1 - CHANGE FIRST CHARACTER OF LABELS DURING DISASSEMBLY.

Using TASMON's M A (Modify memory using ASCII) command, change the following byte from the letter "Z" (5AH) to the desired first character for labels:

MODEL I: 7486H MODEL III: 748AH MODEL 4: F436H

If you have a different version of TASMON, key in F nnnn 3E 5A (where nnnn is normally 6000H in the Models I & III, and E000H in the Model 4). The displayed address PLUS ONE is the byte to change.

PATCH # 2 - REMOVE 7-BYTE "HEADER" FROM DISASSEMBLIES TO DISK.

TASMON was written to output disassembled source code that could be loaded directly into Apparat's modified version of the Radio Shack Editor-Assembler program (as found on NEWDOS/80 master disks). Most Editor-Assembler programs are able to read this source code format, but some (notably Radio Shack's Series I) will not read these source code files because of the seven-byte "header" that is placed at the beginning of these files. To prevent TASMON from writing this "header", use TASMON's M H (Modify memory) command to change two bytes starting at the addresses shown below. The bytes should be changed from 3E D3 to 18 19 in the Models I & III, and to 18 0C in the Model 4. This inserts a JR instruction that bypasses the code that writes the header to disk. Change the two bytes starting at:

MODEL I: 7312H MODEL III: 731CH MODEL 4: F2EEH

If you have a different version of TASMON, key in F nnnn 3E D3 (where nnnn is normally 6000H in the Models I & III, and E000H in the Model 4) to find the starting address to change.

PATCH # 3 - CHANGE AMOUNT OF KEYBOARD DEBOUNCE (MODELS I & III ONLY).

This patch is especially useful when you are using a speed-up modification, or when running the Model III version of TASMON on a Model 4 using the 4 MHz clock speed. Using TASMON's M H (Modify memory) command, change the following byte from 0AH to the value that gives the desired amount of keybounce control. For example, to double the normal keybounce delay, change the byte to 14H:

MODEL I: 7A10H MODEL III: 7A53H

If you have a different version of Model I or III TASMON, key in F 6000 01 00 0A. The displayed address PLUS TWO is the byte to change.

Patches supplied by Jack Decker.