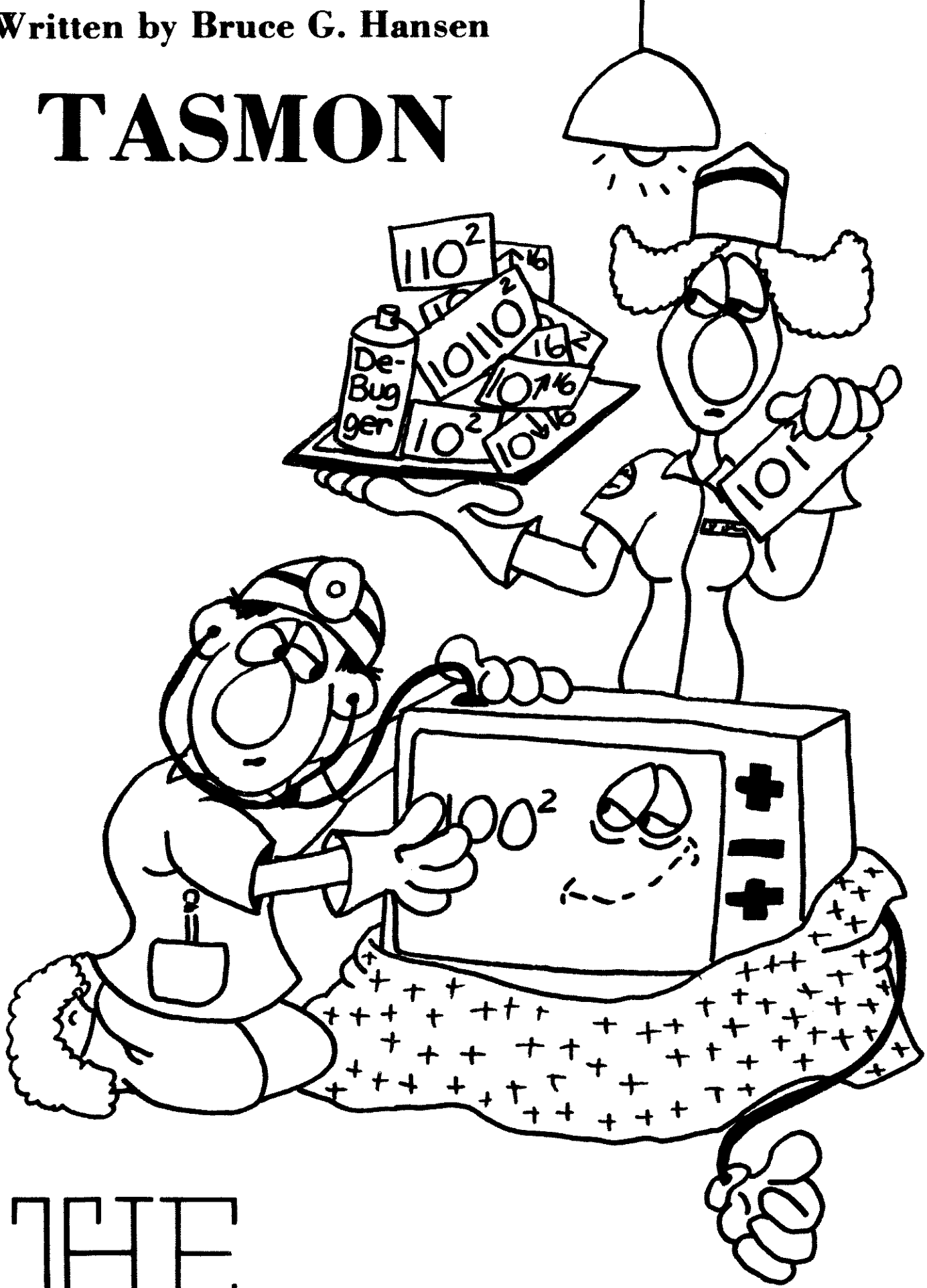
Written by Bruce G. Hansen

TASMON



THE
ALTERNATE
SOURCE

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 or tape as an Editor/Assembler source file with labels generated for pertinent addresses. SYSTEM tapes and machine language disk files can be read in and written out.

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 or tape loads in memory from 6000-7FFF with an entry point of 6000.

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 or tape). 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 its own 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. If no lower case mod is installed, all lower case characters are converted to upper case.

When TASMON is entered, the user's stack pointer is set at 41FE. 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  41E4    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 eight lines below the register display.

MEMORY USAGE

TASMON uses approximately 8K of memory. Some ROM routines are used to decrease program size. No RAM outside of TASMON is used with the exception of the symbol table when disassembling to disk or tape and the user's screen memory when using the KEEP SCREEN command.

LOADING AND REENTERING TASMON

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

TASMON

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

To load a tape version of TASMON type:

```
>SYSTEM
*? TASMON
*? /<ENTER>
```

The display will be the same as with the disk version.

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

1) Go to BASIC and type:

```
>SYSTEM
*? /start address
```

where "start address" is the decimal starting address of where TASMON was located in memory.

2) With a DOS system enter DEBUG and type:

G start address

where "start address" 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. Do not EXIT when disk drives are turning or the computer will lock up.

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 up in memory one byte. The down arrow will move down in memory one 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 15 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 15 lines (78H bytes with ASCII and Hex dumps, 15 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 DUMP

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

H 5200

will cause this type of display to appear on fifteen 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 DUMP

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

A F00C

will cause this type of display on fifteen 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 printed so 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 ROUTINE

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:

F 6000 FE 55 CA <ENTER> 6085

The FIND command just found the first occurence of the menu select routine for the "U" key. The 6000 address should be substituted with the starting address of TASMON (6000 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 or tape. 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 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 6 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    0060H
```

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    0060H
```

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 ROM call to 0060 is a stall 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 F000 F035 E000	Move memory from F000 through to F035 to E000

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 as a SYSTEM tape.

LOADING SYSTEM TAPES AND CMD DISK FILES

To LOAD a SYSTEM tape into memory press the "L" key, a "T" (for tape), and hit ENTER or a four digit offset value.

If the ENTER key is depressed, the module will load into memory normally. If an offset value was entered, this value is added to the load addresses of the tape and data is loaded at the new address. The reason for this offset value is that tapes loading at addresses 4000-51FF will destroy DOS. If DOS is not intact, TASMON can not write or read disk files. If the load is offset so it does not interfere with DOS, TASMON disk commands will function normally. For example:

L T <ENTER>	Load a SYSTEM tape
L T 4000	Load a SYSTEM tape and add 4000 to its load addresses.

The file name of the SYSTEM tape is displayed when loading.

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

```
L D <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 T <ENTER>
F000 F035 F010
```

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

If a SYSTEM tape is offset, use the SUBTRACT command to figure where it would normally load by subtracting the load offset from the starting, ending and transfer addresses. If the module would interfer with DOS, the user is left with two options:

1) Enter the block move program given below
2) Relocate the program

The block move program discussed here will create a module like those made with Apparat's LMOFFSET. This code will move the module to its correct starting address and start it running. To enter this appendage program MODIFY MEMORY in hex mode starting one byte after the ending address of the module just loaded.

The block move program appears as follows:

```
21 xx xx        LD   HL,starting address
11 yy yy        LD   DE,starting address - offset
01 zz zz        LD   BC,ending address - starting address +1
ED B0           LDIR
C3 tt tt        JP   transfer address - offset
```

An example of this procedure would be as follows:

```
L T 2000
6350 6BFA 6500
```

A SYSTEM tape was loaded with an offset of 2000. The starting address is 6350, the ending address is 6BFA and the transfer address is 6500. Subtracting the offset of 2000 from these values gives 4350, 4BFA and 4500, which would interfere with DOS (DOS's high memory address is 51FF). The following bytes would be entered as the appendage program starting at 6BFB, or one byte after the ending address of the example tape, with the MODIFY MEMORY command:

```
21 50 63 11 50 43 01 71 09 ED B0 C3 00 45
```

In Z-80 mnemonics, the program is:

```
LD      HL,6350
LD      DE,4350
LD      BC,0971
LDIR
JP      4500
```

Notice that "xx", "yy", "zz", and "tt" where substituted by the appropriate values. In this case 6350 for "xx", 4350 for "yy", 0971 for "zz" and 4500 for "tt." Also note that the addresses are entered in Z-80 format - LSB first, MSB second (i.e. 4350 is entered as 50 43).

The starting, ending and transfer addresses of the loaded SYSTEM tape and the appendage program are 6350, 6C08 and 6BFB respectively. The starting address is the same since no code was added to the beginning of the program. The new ending address is the original ending address plus the length of the appendage program (6BFA + 000E or 6C08). The transfer address is changed so it executes the appendage program instead of the loaded SYSTEM tape (the appendage program will jump to the loaded program after

the move is through). Since the block move appendage program starts at 6BFB, the transfer address is also 6BFB.

The second option of relocating the program may not always work for reasons discussed under RELOCATE.

The module loaded above will be used as an example of this procedure. The starting, ending and transfer addresses were 6350, 6BFA and 6500 respectively.

The first step is to block move the module from its offset location to its normal executing location. In this case the module is located from 6350 to 6BFA and should be at 4350 to 4BFA. These numbers are figured by subtracting the load offset from the offset starting, ending and transfer addresses. In this case the 4350 is derived by subtracting 2000 from 6350, etc. The "Y" block move is used move the program to its normal addresses as follows:

Y 6350 6BFA 4350	move memory from 6350 through 6BFA to memory starting at 4350

Now the program is at its normal execution location, 4350 through 4BFA. DOS has also been overwritten so all TASMON disk commands are now disabled.

Now the RELOCATE command can be used to move the program up to high memory and change all necessary instructions of the program so it will run at high memory. Suppose we wanted to move it so it would start at 7350. The "X" command would be used as follows:

X 4350 4BFA 7350	relocate memory from 4350 through 4BFA to memory starting at 7350

Now the program resides at 7350 through 7BFA. The entry address can be found by adding the relocate offset to the original entry address. In this case the relocate offset is 3000 (7350 - 4350) and the original entry address is 4500 which gives a new entry address of 7500.

To verify that the relocated version of the program functions correctly, type the following:

G 7500	Start execution at 7500 ("G" is discussed below)

If the relocated version of the program is to be written back out to disk, press the RESET button to reboot DOS and write the file out as described in the WRITE command instructions. The starting, ending and transfer addresses of the relocated version are 7450, 7BFA and 7500 respectively.

Most, but not all programs will function correctly when moved by this process. When a program is found not to work when relocated, use the block move technique described above.

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 or tape file, except the VIEW command does not load the file into memory.

To execute the VIEW command press "V" and a "T" for tape or "D" for disk. If tape was selected no other parameters are entered. If disk was selected a file name must be entered. For example:

```
V D
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.

WRITING OUT A SYSTEM TAPE OR CMD DISK FILE

To write a file out press "W" (for WRITE) followed by a "T" for tape or a "D" for disk. The starting, ending and entry addresses are entered next in that order. Lastly, the filename is entered, up to six characters for tape or a DOS filename for disk. 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 D 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"

DISASSEMBLED OUTPUT TO DISK OR TAPE

The OUTPUT command will disassemble to disk or tape as an Editor/Assembler source file. The code sent to disk or tape is also echoed on the screen. To execute this command press the "O" key (for OUTPUT), a "D" for disk or "T" for tape, 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 located at 4049-404A 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. 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 D F000 F035 F010	Output to disk starting at F000, ending at F035, and entry address of F010.
TEST/ASM	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, RAM or ROM. Breakpoints in ROM will not function when using the GO command (discussed below), but they do work when using the TRACE command. 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 18, 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
```

SET 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. NOTE: a CALL to ROM will not halt at a breakpoint within the CALL with this stepper.

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.

A unique feature of TASMON is that ROM instructions may be single stepped by either type of single stepper. It is recommended that the DOWN ARROW type (CALLs executed in full) be used since some ROM routines can take quite a while to execute.

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 COMMAND

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). Everytime 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.

TASMON allows tracing through ROM if desired (some of the routines take quite a while to finish) and breakpoints in ROM are honored. Some special conditions must be met for breakpoints in ROM to halt the program however.

If a CALL is used and a breakpoint is set somewhere within the ROM CALL, tracing with CALLs executed in full will not halt on the breakpoint. Tracing with CALLs stepped through will halt if the breakpoint is hit.

For breakpoints in ROM to function with the TRACE command, the address of the breakpoint must be single stepped. When stepping through a CALL, all instructions of the CALL are single stepped. When executing CALLs in full, the CALL is executed in its entirety in one step. Breakpoints can not actually be loaded into ROM, only RAM. Therefore, tracing with CALLs stepped through is required to honor breakpoints within CALLs to ROM.

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.

Since breakpoints can not really be set in ROM, GO will not halt execution in ROM.

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 40 to 63 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 1024 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 1024 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 F000	Set user's screen buffer at F000-F3FF 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 author for $15. The author's address is:

Bruce G. Hansen
220 Iris Street
Lansing, MI 48917
(517) 323-2260

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 ROM routines of the BASIC interpreter to perform these tasks.

NOTE: Disk BASIC seems to have problems when single stepping in the manner described below.

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

One way is to enter LEVEL II BASIC and set a MEMORY SIZE high enough to protect TASMON. Next, load TASMON via the SYSTEM command. Since TASMON takes about 8K of memory, the memory sizes should be:

16K machine = 24575
32K machine = 40959
48K machine = 57343

Disk users can load TASMON from DOS and enter LEVEL II BASIC by pressing the BREAK key and RESET button. Next, enter the MEMORY SIZE.

TASMON must be entered next. To do this type:

```
>SYSTEM
*? /start address
```

Where "start address" is the starting address of TASMON.

The next step is to set a breakpoint at 41B2. This the address of a CALL used by ROM to return to 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:

```
>SYSTEM
*? /start address
```

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

RESTARTS must be set to "step through" mode. Press the "J" key until this mode is enabled. The "step through" mode is on when an "I" 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 at 0060. For these reasons CALLs executed in full is recommended for stepping BASIC.

The re-entry address of BASIC is 1A19. Modify the PC register to this address before stepping BASIC as follow:

R PC 1A19

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

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

The breakpoint at 41B2 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 41B2 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 6 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 - Load TASMON, relocate it to high memory and write it back out to disk.

The distributed version of TASMON loads from 6000-7FFF with an entry point of 6000. By distributing the program in this form, only one version is needed for a 16K, 32K or 48K machine. However, the owner of a 48K machine will probably want TASMON to run at high memory or E000-FFFF. To do this enter the following commands:

From DOS enter TASMON by typing:

TASMON

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

Next use the RELOCATE command to move the program to memory starting at E000. The format is:

X 6000 7FFF E000 — which relocates memory from 6000-7FFF to memory starting at E000.

Now TASMON resides at 6000-7FFF and at E000-FFFF. To save the high memory version to disk use the WRITE command. The format is:

W D E000 FFFF E000
HTASMON/CMD

Which dumps memory from E000-FFFF with a transfer address of E000 to disk with the file name "HTASMON/CMD".

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

SESSION 2 - Load Small System Software's BARRICADE game program from tape, relocate it to high memory and write it back out to disk.

It is assumed in this example that TASMON resides in memory from 8000-9FFF.

To load BARRICADE enter the following command:

```
L T 2000              Load a SYSTEM tape with an offset of 2000
6350 6D6F 6350
```

The offset of 2000 is needed since BARRICADE loads over DOS. The starting, ending and transfer addresses are 6350, 6D6F and 6350 respectively.

The next step is to enter LEVEL II BASIC in a non-DOS environment. To do this hold down the BREAK key and hit the RESET button.

Now reenter TASMON by keying in the following:

```
>SYSTEM
*? /32768      Assuming TASMON starts at 8000H (32768
               decimal). This address must be the same as
               the starting address of TASMON.
```

Now block move the program back down to its normal execution location with the BLOCK MOVE "Y" command as follows:

```
Y 6350 6D6F 4350
```

The 4350 destination address was derived by subtraction the load offset (2000) from the starting address (6350).

BARRICADE now resides at its normal addresses. To relocate it to memory from 7350-7D6F enter the following:

```
X 4350 4D6F 7350
```

This will relocate memory from 4350-4D6F to memory starting at 7350. As a result, a relocated high memory version of BARRICADE is at memory from 7350-7D6F. The entry address of the relocated module is 7350. This is derived by adding the relocate offset of 3000 (7350-4350) to the normal entry address of 4350.

Now reenter DOS by hitting the RESET button again. To reenter TASMON type in:

```
TASMON
```

The final step is to write the relocated version of the program to disk. The WRITE command is used to accomplish this as follows:

W D 7350 7D6F 7350
BARRIC/CMD

Memory from 7350-7D6F was written to disk under the file name "BARRIC/CMD" with an entry address of 7350.

To run the relocated version of BARRICADE type the file name of the new module from DOS.

SESSION 3 - Load a machine language file from disk and execute it by single stepping, tracing and GOing.

The short program used in this example appears as follows:

```
00100        ORG     5F00H               ;START OF PROGRAM
00110        LD      HL,TEXT             ;START OF TEXT
00120 LOOP   LD      A,(HL)              ;GET A BYTE
00130        INC     HL                  ;POINT TO NEXT BYTE
00140        CP      0                   ;END OF MESSAGE?
00150        JR      Z,STOP              ;JUMP IF END
00160        CALL    033AH               ;WRITE BYTE
00170        JR      LOOP                ;GET ANOTHER BYTE
00180 STOP   JP      STOP                ;KEEP ON JUMPING
00190 TEXT   DEFM    'THIS IS A TEST'    ;TEXT MESSAGE
00200        DEFB    0DH                 ;CARRIAGE RETURN
00210        DEFB    00                  ;MESSAGE DELIMITER
00220        END     5F00H               ;ENTRY POINT IS 5F00
```

The purpose of this program is to write a message on the screen. In this case the message is "THIS IS A TEST". For this session TASMON is assumed to be in memory from 6000-7FFF 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 D <ENTER>
TEST/CMD
5F00 5F20 5F00
```

The file "TEST/CMD" loaded from 5F00-5F20 with an entry point of 5F00.

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 5F00. Use the REPLACE command to do this:

```
R PC 5F00
```

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

```
D 5F00
```

The first fifteen 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:

```
5F00 21115F     LD      HL,5F11         LD      HL,5F11
5F03 7E         LD      A,(HL)          IX  4C41  IY  094C
5F04 23         INC     HL              AF' 4B43  BC' 4353
5F05 FE00       CP      00              DE' AA52  HL' 0B0A
5F07 2805       JR      Z,5F0E          AF  00FF  BC  4C44
5F09 CD3A03     CALL    033A            DE  4C48  HL  A070
5F0C 18F5       JR      5F03            SP  41E4  PC  5F00
5F0E C30E5F     JP      5F0E            SZ1H1PNC  (HL)  4C
5F11 54         LD      D,H             -
5F12 48         LD      C,B
5F13 49         LD      C,C
5F14 53         LD      D,E
5F15 2049       JR      NZ,5F60
5F17 53         LD      D,E
5F18 2041       JR      NZ,5F5B
```

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 5F00 (which is a LD HL,5F11) hit the DOWN ARROW or "I" key. The HL register pair will now have 5F11 in it, PC equals 5F03 and the instruction at 5F03 (or PC) is LD A,(HL).

Single step this instruction. The A register will hold 54 or an ASCII "T", the first character of the message. PC will now be 5F04. The next instruction is "INC HL". Single step PC again. HL equals 5F12 or the address of the next character of the message.

PC now holds 5F05. The instruction there is a CP 00. This instruction checks for the end of the message which is a 00 byte. Single step this instruction. Notice that the Z-80 flags changed (the Z flag will not be set because 54 does not equal 00).

The next instruction is JR Z,5F0E which is where the program jumps if the message is through being printed.

The next instruction is CALL 033A. PC points to this instruction by holding 5F09. This is a ROM CALL to display the character in the A register on the screen. To single step this instruction hit the DOWN ARROW key. If the "I" key is depressed the CALL will be stepped through one instruction at a time. You may want to try this just to see how ROM writes the character on the screen. The routine does take some time to step through however.

After the byte is displayed on the screen the program jumps up to get another character from the message. The process repeats

until the entire message is displayed at which time the program merely jumps upon itself in an endless loop.

Single stepping through the program by hand can take some time, but it is necessary when a program may have bugs present. The TRACE command can be used to single step through a section of code at a higher rate of speed while still displaying the Z-80 registers.

Before running the TRACE command change the PC register back to the start of the program by entering:

R PC 5F00

Now start TRACE by pressing the "T" key. Next hit the DOWN ARROW key to select tracing with CALLs executed in full.

The program will single step at about one instruction per second. Press the "4" key. Notice that the program is executing a little faster. The speed control keys are 0-7 where 7 is the fastest rate.

Press the BREAK key after the entire message has been printed. The entire program was just traced through. But suppose we want to stop the program every time it checks for the end of the message at 5F05 (CP 00). To do this a BREAKPOINT can be set. A breakpoint is analogous to a BASIC STOP command.

Since we want to stop the program at 5F05, a breakpoint will be placed there. To set a breakpoint type:

B 1 5F05 which sets breakpoint 1 at 5F05.

Now set the PC register back to the start of the program:

R PC 5F00

Start TRACE again by entering:

T ↓

The program stopped when PC was 5F05 which was where our breakpoint was set. Up to nine breakpoints can be set in this manner.

Suppose we want to display the next five characters on the screen without having breakpoint 1 halt execution before each character is displayed.

To do this the number of executions of breakpoint 1 must be set to 5 as follows:

N 1 05 Set number of executions for breakpoint 1 to 05.

Now continue tracing by entering:

T ↵

The first five characters were printed on the screen and the program stopped at the breakpoint again. To clear that breakpoint type:

C 1 which clears breakpoint 1

If tracing is continued at this point execution will not halt until the BREAK key is depressed.

The CALL at 5F09 appears as follows in ROM:

```
033A    PUSH    DE
033B    CALL    0033
033E    PUSH    AF
033F    CALL    0348
0342    LD      (40A6),A
0345    POP     AF
0346    POP     DE
0347    RET
```

Suppose we want to observe the registers for some reason when PC is 033B. In order to observe the program at this point a breakpoint in ROM must be set. To do this enter:

B 1 033B which sets breakpoint 1 at 033B

The number of execution for breakpoint 1 was previously set to 05. To set this and every other execution number to 01 enter:

N I Initialize execution numbers to 01

Before tracing through the program the PC register must be set to the beginning of the program:

R PC 5F00

Breakpoints in ROM can only be "seen" by TASMON if the instruction where the breakpoint is set is single stepped. For example, if the following code was present in ROM:

```
1000    LD      A,5
1002    LD      B,6
1004    RET
```

And a breakpoint was set at 1002, the breakpoint would halt the program only if the instruction at 1002 was single stepped.

If a CALL 1000 instruction was executed the breakpoint would not halt the program if CALLs were executed in full (the instruction

at 1002 would not be single stepped), but it would halt execution if CALLs were stepped through (each instruction of the CALL is single stepped).

Therefore, for this breakpoint to work on TRACE we must specify CALLs stepped through as follows:

T I

The program will halt when PC equals 033B. Suppose we want to step through instructions residing at 033B-0341 with CALLs executed in full. To do this set another breakpoint at 0342, or one instruction after the CALL 0348:

B 2 0342 which sets breakpoint 2 at 0342

Now execute TRACE with CALLs executed in full:

T ↓

The program will halt at 0342 which was where our breakpoint was set. If tracing was continued with CALLs stepped through the breakpoint at 033B would halt the program. If CALLs are executed in full neither breakpoint will not halt the program for reasons discussed above.

The third way of executing the user's program is the GO command. This command will run the program at full speed. The only way to halt the program when using GO is to hit a breakpoint. If we want to run the entire program through at full speed a breakpoint should be set at 5F0E, or the ending instruction of the program. 5F0E contains a JP 5F0E which is just an endless loop where the program jumps after the message is through being displayed. To set the breakpoint enter:

B 3 5F0E which sets breakpoint 3 at 5F0E

To start the program using the GO command key in:

G 5F00

The message should be printed on the screen instantaneously and control should be returned to TASMON. If the breakpoint at 5F0E was not set, execution would not cease and the program would jump upon itself until the RESET button was pressed.

SESSION 4 - Write the TEST/CMD program out to disk as an Editor/Assembler source file.

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

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

```
L D <ENTER>
TEST/CMD
5F00 5F20 5F00
```

Next, enter the OUTPUT command as follows:

```
O D 5F00 5F20 5F00
TEST/ASM
```

The disassembly will be written out to disk with the file name TEST/ASM starting at 5F00 and ending at 5F20 with a transfer address of 5F00. Now exit TASMON by keying in:

```
E <ENTER>
```

The system will reboot DOS. Suppose you have Apparat's or MISOSYS's Editor/Assemblers. If you do not have either, I highly recommend the purchase of the MISOSYS version called DISKMOD (which requires the cassette E/A sold by Radio Shack). 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. TASMON writes out the file with line numbers of 00000 so this command is required. To do this enter:

```
N 100,10      which renumbers the program in increments of
              10 with a starting line number of 100.
```

The source listing should be:

```
00100           ORG     5F00H
00110           LD      HL,Z5F11H
00120 Z5F03H    LD      A,(HL)
00130           INC     HL
00140           CP      00H
00150           JR      Z,Z5F0EH
00160           CALL    033AH
00170           JR      Z5F03H
00180 Z5F0EH    JP      Z5F0EH
00190 Z5F11H    LD      D,H
00200           LD      C,B
00210           LD      C,C
00220           LD      D,E
00230           JR      NZ,5F60H
```

```
00240           LD      D,E
00250           JR      NZ,5F5BH
00260           JR      NZ,5F70H
00270           LD      B,L
00280           LD      D,E
00290           LD      D,H
00300           DEC     C
00310           NOP
00320           END     5F00H
```

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 00230 do the following:

Start counting instructions starting at the last known address. In this case the last know address is 5F11 (or Z5F11H - 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,5F60H instruction in line 00230 is 5F15. 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 5F15 from 5F60 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 4BH.

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 4BH gives 49H, which is an ASCII "I".

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.

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 5 - Load a CMD disk file into memory and write it out as a SYSTEM tape.®

The first step is to load the disk file into memory. It is good practice to VIEW the file first. In this example TASMON is assumed to reside in memory from E000-FFFF.

To VIEW the file enter:

```
V D
NOVA/CMD
5C00 7FE0 5F0B
```

The disk file "NOVA/CMD" was VIEWed and the starting, ending and transfer addresses were found to be 5C00, 7FE0 and 5F0B respectively. Since TASMON resides from E000-FFFF the module will not interfer with TASMON. However, if the module would interfer with TASMON, the RELOCATE command could be used to move TASMON to a location in memory where the module would not overlap.

The next step is to load the module into memory:

```
L D <ENTER>
NOVA/CMD
5C00 7FE0 5F0B
```

The last step is to write the SYSTEM tape out using the same starting, ending and transfer addresses of the disk file:

```
W T 5C00 7FE0 5F0B
NOVA
```

The SYSTEM tape was written out with the file name "NOVA".

SESSION 6 - Use the TRACE command to step through the start up procedure for ROM and execute a BASIC program.

In this example TASMON must reside in memory from 6000-7FFF and if an expansion interface is connected to the keyboard, the EI must be turned off. The reason for turning the EI off is that the ROM initialization routine checks if a disk system is present. In our example we do not wish this.

The first step is to set the PC register to 0000:

R PC 0000

Now set RESTARTS to stepped through mode by pressing the "J" key:

J I

Now start TRACE by typing:

T ↓

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

After a long initialization process, the MEMORY SIZE message will appear. Enter the following,

MEMORY SIZE ? 24575 <ENTER>

The memory size was set at 24575 to protect TASMON.

We are now tracing through LEVEL II 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 41B2. Do this by entering:

B 1 41B2

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 41B2 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 1A19 as follows:

R PC 1A19

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.

SESSION 7 - Relocate GSF, a utility program by RACET COMPUTES

GSF is a utility program with routines to scroll the screen in any direction, reverse graphics, draw graphics lines, read/write tape blocks at high speed and other commands. The most noteworthy of the commands is the multiple variable sort.

GSF is a fine program but has one fault, it resides at high memory and provides no way to move itself down in memory.

Other programs may also need to reside at high memory thus interfering with GSF. The solution is to relocate GSF to a new lower loading point in memory.

GSF uses a vectoring technique which fools TASMON's RELOCATE command. Jump addresses for each routine are stored in a table instead of actual Z-80 jump instructions. GSF stores this table in the following format:

1st byte	number of arguments for this routine
2nd and 3rd bytes	address of this routine

This jump table is stored at these locations for the three versions of GSF:

Memory version	start of table	end of table
16K	7F87	7FFE
32K	BF87	BFFE
48K	FF87	FFFE

GSF will be relocated normally but the jump table addresses must be changed by hand. The procedure goes as follows:

Suppose TASMON resides in memory from 8000-9FFF and the 48K version of GSF is stored on disk with the filename "GSF48/OBJ".

The first step is to load GSF into memory:

```
L D <ENTER>
GSF48/OBJ
F2D8 FFFF FE80
```

GSF48/OBJ loads in memory from F2D8 to FFFF. In our example we would like GSF to end at FED2 instead of FFFF. We must first find the program offset by entering:

```
S FFFF FED2 - 012D
```

The new starting and entry addresses are found by entering:

```
S F2D8 012D - F1AB      new starting address
S FE80 012D - FD53      new entry address
```

Now GSF can be relocated to its new position in memory starting at F1AB:

X F2D8 FFFF F1AB relocate memory from F2D8 to FFFF to memory starting at F1AB

The starting address of the jump table of the lower memory version of GSF must be found by subtracting the offset from the original table location:

S FF87 012D - FE5A

The original jump table must be copied to the new lower memory version of GSF. The jump table may be incorrectly interpreted when relocated since it is not normal Z-80 code. As a result it can not be assumed that information saved at the lower memory GSF is valid. Table data lies from FF87 to FFFE for the loaded version of GSF as indicated by the above table. This data is copied to the new lower version as follows:

Y FF87 FFFE FE5A

To view the jump table enter:

H FE5A

The jump table will be displayed in hex on the left side of the screen. The display should appear as follows:

```
FE5A 02 CC FA 02 97 FA 03 E9
FE62 FA 03 7C FB 02 C5 FB 00
FE6A D5 FB 00 EB FB 00 01 FC
FE72 00 1A FC 02 33 FC 03 46
FE7A FC 02 4A FD 03 A7 FD 03
FE82 17 FE 03 52 FE 01 62 FE
FE8A 01 7A FE 03 FE F4 04 0C
FE92 F5 02 16 F5 02 36 F5 03
FE9A 93 F5 04 99 F5 01 FC F2
FEA2 00 FF 00 FF 00 FF 00 FF
FEAA 00 FF 00 FF 00 FF 00 FF
FEB2 00 FF 00 FF 00 FF 00 FF
FEBA 00 FF 00 FF 00 FF 00 FF
FEC2 00 FF 00 FF 00 FF 00 FF
FECA 00 FF 00 FF 00 FF 00 FF
```

Hit the BREAK key to reenter command mode.

The jump table is a collection of three byte values as described above. The first set of three is at FE5A through FE5C. The bytes at these addresses are 02, CC and FA respectively. These bytes are interpreted as:

02 Number of arguments for this routine

FACC The address of this routine (CC FA is the Z-80 format for the address FACC)

The next step to relocating GSF is to change all of the jump addresses to their correct values. The correct value is found by subtracting the offset from the original value. The first one is done as follows:

S FACC 012D - F99F

Now the address FACC must be replaced by F99F. This is done by modifying memory at the jump table location of FACC at FE5B:

M H FE5B CC 9F
FE5C FA F9
FE5D 02 <BREAK>

All of the other routines are modified in a similiar manner. The table ends at FEA1 or where the 00 and FF values start appearing. After all of the entries have been modified the new version of GSF may be saved to disk. The starting, ending and entry addresses determined above were F1AB, FED2 and FD53 respectively:

W D F1AB FED2 FD53
NEWGSF/OBJ

The new version of GSF may be loaded from DOS by typing:

LOAD NEWGSF/OBJ <ENTER>

When BASIC is entered the memory size must be set to one less than the starting address of GSF (assuming GSF is the lowest high memory program). In our example this value is 61866 or F1AB minus one.

The DEFUSR statement used to enter GSF uses the entry address of the program. In our example this is FD53. The DEFUSR would be entered as follows:

DEFUSR= &HFD53

The new relocated version of GSF will function exactly as the normal version.

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 T <offset>	Load in a SYSTEM tape with optional offset.

L D <offset> "file"	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 T SS EE TT file	Output disassembled listing starting at SS, ending at EE with a transfer address of TT to tape with the file name "file".
O D SS EE TT file	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 T	View a SYSTEM tape. Returns starting, ending and transfer addresses without loading into memory.
V D file	View the disk file titled "file". Returns starting, ending and transfer addresses without loading into memory.
W T SS EE TT file	Write a SYSTEM tape starting at SS, ending at EE with a transfer address of TT and file name of "file".

W D SS EE TT file	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
CLEAR KEY	Clear the screen and display the registers.
BREAK KEY	Return to command mode.

APPENDIX C - MODEL I VERSION

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

This patched routine assumes that TASMON version 2.12 is being used. Also, TASMON should be located in memory starting at 6000H. The following bytes are entered:

M H 7F7C 00

and enter the following bytes from there:

```
7F7C: 00 00 CD 5C 69 CD E3 60 FE 44 28 0B FE 48 28 3E FE 55
7F8E: CA EE 60 18 EE CD 5C 69 32 21 80 C3 30 68 AF 32 21 80
7FA0: 32 20 80 C3 34 60 3A 21 80 B7 28 1A CD 0F 69 2A 3C 7A
7FB2: CD 68 71 3E 20 CD 5C 69 21 28 3C 06 11 7E 23 CD BA 79
7FC4: 10 F9 3A DB 7A C9 CD 5C 69 32 20 80 C3 30 68 3A 20 80
7FD6: B7 CA FA 60 3A E8 37 FE 40 D2 FA 60 2A 3C 7A 7C CD 06
7FE8: 80 7D CD 06 80 3E 20 CD E8 79 CD FA 60 06 14 21 28 3C
7FFA: 7E CD E8 79 23 10 F9 3E 0D C3 E8 79 F5 CB 3F CB 3F CB
800C: 3F CB 3F CD 15 80 F1 E6 0F C6 30 FE 3A 38 02 C6 07 C3
801E: E8 79 00 00
```

MODIFY MEMORY in hex as follows:

```
M H 60DE   EE 7E
60DF ' 60 7F
60E0   C3 <BREAK>

M H 6729   FA D3
672A ' 60 7F
672B   C3 <BREAK>

M H 6855 : 3A CD
6856   DB A6
6857   7A 7F
```

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 60F8 4 34 9C
60F9 ' 60 7F
60FA : 3A <BREAK>
```

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

```
W D 6000 8021 6000
UPTASMON/CMD <ENTER>
```

A patched tape version can be written by substituting a "T" for the "D" in the WRITE command shown above.

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 SEVEN EXECUTED INSTRUCTIONS patch is executed by pressing the "U" key and the "D" key for DISPLAY LAST SEVEN 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 eight previously executed instructions will be displayed.

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

If even more user routines are to be added, the address at 7F8F-7F90 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.

APPENDIX C - MODEL III VERSION

This appendix is simply a Model III version of the above appendix C for the Model I. The instructions for using this patch are not given here - refer to those given above.

To start inputting the patch the following bytes are entered:

```
M H 7FE8   00
```

and enter the following bytes from there:

```
7FE8: 00 00 CD 5B 69 CD E1 60 FE 44 28 0B FE 48 28 3F
7FF8: FE 55 CA EC 60 18 EE CD 5B 69 32 85 80 C3 2D 68
8008: AF 32 84 80 32 85 80 C3 33 60 3A 85 80 B7 28 1A
8018: CD 0E 69 2A 98 7A CD 6B 71 3E 20 CD 5B 69 21 28
8028: 3C 06 11 7E 23 CD 1C 7A 10 F9 3A 37 7B C9 CD 5B
8038: 69 32 84 80 C3 2D 68 3A 84 80 B7 CA F8 60 2A 98
8048: 7A 7C CD 6A 80 7D CD 6A 80 3E 20 CD 45 7A CD F8
8058: 60 06 14 21 28 3C 7E CD 45 7A 23 10 F9 3E 0D C3
8068: 45 7A F5 CB 3F CB 3F CB 3F CB 3F CD 79 80 F1 E6
8078: 0F C6 30 FE 3A 38 02 C6 07 C3 45 7A 00 00
```

MODIFY MEMORY in hex as follows:

```
M H 60DC    EC EA
60DD  ' 60 7F
60DE    C3 <BREAK>

M H 6726    F8 3F
6727  ' 60 80
6728    C3 <BREAK>

M H 6852  : 3A CD
6853  7 37 12
6854    7B 80
```

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 60F6 3 33 08
60F7  ' 60 80
60F8  : 3A <BREAK>
```

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

```
W D 6000 8085 6000
UPTASMON/CMD <ENTER>
```

If more user routines are to be added, the address at 7FFB-7FFC may be modified to the starting address of the new routine.

APPENDIX D

PATCH FOR RADIO SHACK SERIES I EDITOR/ASSEMBLER

Radio Shack recently started shipping a new version of their editor/assembler. This version is titled Series I. The tape version of this assembler uses the same tape format as their past assembler did. However, the disk version uses a different format for writing source files to disk. This change requires a patch in TASMON to allow the Output disassembly function to work properly.

For Model I users, change 723C to 18 and 723D to 19.

For Model III users, change 7242 to 18 and 7243 to 19.

This should take care of any problems using the Series I package.

APPENDIX E

TECHNICAL INFORMATION

This appendix will give some useful patches and memory locations in TASMON.

A problem exists when disassembling to tape with a non- expansion interface Model I computer. TASMON uses the value at 4049H for the top of memory. This is the correct value for an expansion interface equipped machine but not a 16K LEVEL II machine. The patch required for the 16K machine is as follows:

```
Starting address = 7218H
Change:  49 40   to
         B1 40
```

The following assembly language listing is for a patch to allow ASCII characters to be displayed while disassembling to the screen or printer. The listing is shown for the Model I with appropriate changes for the Model III. Note that it is assembled below 6000H. The reason for this is that if it were placed at the end TASMON, this patch would conflict with the Appendix C patch. This routine can be put anyplace in memory by changing the ORG statement as needed.

```
00100           ORG     5FD0H     ;5FD0H for MODEL III
00110           NOP
00120           NOP
00130 ASCDIS    PUSH    HL
00140           PUSH    BC
00150 WR10      CALL    SHEX1
00160           INC     HL
00170           DJNZ    WR10
00180           POP     BC
00190           POP     HL
00200           CALL    SPCD1
00210 WR20      LD      A,(HL)
00220           CP      20H
00230           JR      C,WR40
00240           CP      80H
00250           JR      NC,WR40
00260           CALL    BYTOUT
00270 WR30      INC     HL
00280           DJNZ    WR20
00290           JP      6B1AH     ;6B1DH for Model III
00300 WR40      CALL    SPCD1
00310           JR      WR30
00320 SHEX1     EQU     7066H     ;7069H for Model III
00330 SPCD1 ,   EQU     71B7H     ;71BAH for Model III
00340 BYTOUT    EQU     7006H     ;7009H for Model III
00350 ;
00360 ;
```

```
00370           ORG       6B14H      ;6B17H for Model III
00380           JP        ASCDIS
00390           END
```

There are two important notes which should have been made more clear in the manual.

The first one deals with disassembled output to disk/tape. TASMON does not leave very much free memory for a symbol table when it is located at high memory. If the memory being disassembled requires more than 50 bytes for its symbol table (25 symbols) and TASMON is at high memory, then the symbol table will start building down on top of TASMON! Care should be taken to avoid this problem.

The other note regards the memory dumps, in particular the disassembled dump. When the disassembled dump is activated, pressing the "-" key will cause the dump to go back 15 instructions. As far as we know, TASMON is the only monitor available with this feature.

The following is a list of various patch points. The function of each is also defined.

Disabling labels when disassembling: MODEL I - set memory address 72C6H to a C9H. To reenable, set 72C6H to an AFH. For the MODEL III, use memory address 72C7H.

Let the ASCII dump display graphics characters: MOD I - set memory address 6806H to the hex value of the highest displayed byte. For example, setting this value to C0H will allow graphics characters to be displayed. The value is originally set to 80H. For the MOD III use a memory address of 6803H.

Change the label character used when disassembling to disk/tape to something other than a "Z". TASMON writes out labels as a "Z" followed by the hex address of the instruction. This character can make it difficult when trying to find labels since the "Z" character is used in such instructions as "JR Z,$5" and "RET Z". A "Q" could be a better choice. For the MOD I, change the byte at address 73B0H to something other than a "Z". The address for the MOD III is 73B1H.

The list of routine here may be useful for users who wish to access routines already present in TASMON when writing USER routines. The addresses are shown in the format "MOD I/MOD III" where the first number is the MODEL I address and the second is the MODEL III address. It is assumed that these addresses will be CALLed unless otherwise noted (i.e. if the address given was 6038, do a "CALL 6038" to execute it).

60EE/60EC Jump back to TASMON and reset TASMON's stack. This routine is useful for returning to TASMON from USER routines. TASMON will scroll its display and ask for another command.

69AB/69AA Clear the video display.

6AA9/6AAC Disassemble instructions starting at the one pointed to by the HL register pair. The BC register pair holds the number of instructions to disassemble and the A register holds the display flag. A=1 means display the dump, A=0 means do not display the disassembly.

633F/633D Display the current user registers.

6543/6541 Input a four digit hex value into the HL register pair and echo inputted text to the screen.

60E3/60E1 Get a keyboard character. If the "*" key is hit, do a screen dump. If the BREAK key is pressed, jump back to TASMON. The character depressed is returned in the A register.

7168/716B Display the contents of the HL register pair in hex on the screen.

7A26/7A82 Start of user's register save area. The registers are stored in memory in the same order that they are displayed from TASMON.

7070/7073 Display the A register on the video display in hex.

62EF/62ED TASMON's reentry point from a breakpoint. A breakpoint is a CALL instruction. Calling this address will cause the current Z-80 registers to be saved as the user's registers. The PC register will be POPed off the stack.

78E8/7933 TASMON's keyboard driver. If you wish to install your own keyboard driver, put a "JP" to your driver at this address.

79BA/7A1C TASMON's video driver. If patching in your own routine, do not assume that the registers are saved before entering your routine.

79E8/7A45 TASMON's printer driver. If patching in your own routine, do not assume that the registers are saved before entering your routine.

TASMON

With TASMON, 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 or tape as an Editor/Assembler source file with labels generated for pertinent addresses. SYSTEM tapes and machine language disk files can be read in and written out.

The following is just a partial list of TASMON capabilities:

- Replace Registers
- Modify Memory
- Hex Memory Dump
- ASCII Memory Dump
- Disassembled Dump
- Disassemble to Printer
- Dump Screen to Printer
- Sum Hex Values
- Subtract Hex Values
- Find 1-4 Consecutive Bytes
- Skip Forward One Instruction
- Back Up One Instruction
- Clear Screen
- Relocate System Programs
- Load a System Tape
- Load a /CMD Disk File
- Write a System Tape
- Write a /CMD Disk File
- Disassemble to Disk
- Disassemble to Tape
- ROM or RAM Breakpoints (9)
- Set Breakpoints
- Display Breakpoints
- Clear Breakpoints
- Single Step with CALL in full
- Single Step through CALLs
- Trace at Eight Speeds
- GO Execute Program

TASMON is fully relocatable, file oriented, and excellently implemented. The users manual is complete with sample sessions and a command reference card. 32K is recommended. Model III version is available.