THE ZAPPLE™ MONITOR

USER'S MANUAL

The ZAPPLE Monitor

Version 1.1

December 30, 1976

THE ZAPPLE MONITOR

A. INTRODUCTION

This monitor system is a result of many years of work by many people. It was first used on an 8008 system (Intellec 8 by Intel), and then later modified for the 8080. Although the actual code has been greatly modified, the basic concept has been retained in this Z-80 implementation.

There are many approaches that can and have been used regarding a "SYSTEM MONITOR". The one that is used here is probably the most desirable for either the industrial or the hobbyist/experimenter environment. This monitor may be classified as a "DEBUG" monitor. That is, it contains all the needed tools to fully debug both hardware & software, as well as support the I/O used by the system or transient USER programs. It should be painfully obvious to anyone that has watched this field grow that each person has his own idea as to what he wants (or can afford) for his Input/Output devices. This makes exchanging software extremely difficult, and takes some of the fun out of it. TDL's ZAPPLE monitor may help solve this problem. All of our resident software contains NO I/O routines whatsoever! What we are suggesting is that since everyone's I/O is different, let's make the applications software "I/O INDEPENDENT", and then supply a universal monitor system that each person can CUSTOM-FIT to his own particular needs. As long as the I/O VECTORS are honored, then a BASIC or a FORTRAN or a TEXT EDITOR (etc.) program does not have to be concerned whether you have a parallel input keyboard & video display or a model 33.

The ZAPPLE system monitor program may be called a "CHARACTER ORIENTED" system. This means that there are no buffers needed to buffer keyboard input, or hold data waiting for output. It handles all I/O through vectors at the beginning of the program. It also contains features that have come about as the need for them was felt during it's use by the author. This monitor will come to be the most important piece of software in your system.

The Zapple Monitor occupies 2K of memory, is relocatable, expandable, and ROMable.

The expandability feature is of tremendous user importance as it allows the user to attach his own additional monitor routines at the end of the monitor. Such routines often include I/O drivers. Typical additions might be a VDM driver routine or cassette driver routine. Specific routines will be published in the TDL User's group newsletter from time to time.

The monitor also includes many useful subroutines that may be used by user written programs (see the Assembly listing).

B. LOADING PROCEDURE

All TDL software is relocatable. That is, it may be loaded and run from any address the user chooses, providing only that sufficient memory space above the starting address is provided. Loading of all this software is accomplished via the Monitor. The monitor however requires a bitswitched loader for its loading.

If you now have the 1-K ZAP monitor in running in your system, it is a simple matter to load the 2-K ZAPPLE. Set the tape up on the reader AFTER the binary boot-strap, and type: R,F000(cr), and start the reader. If ZAP is now located at 0F000H, you may load a temporary copy at some other location, and then load ZAPPLE up at 0F000H. Remember, this is the location that future software expects to find the monitor. (although it may be located elsewhere, and the I/O vectors modified in any future software accordingly.)

ZAPPLE will initially be set up for the old MITS standard, I.E.:

DEVICE	STATUS	DATA	TEST BIT
MAIN CONSOLE	0	1	0=RDA, TBE=7
CRT	4	5	SAME
CASSETTE	6	7	SAME

The test bits are active low. I.E.

Receiver Data Available (RDA)= 0 (there is data)
Transmitter Buffer Empty (TBE)= 0 (You may load the buffer.)

Once the program has been loaded, the user is free to modify any of the drivers, using the software listings included with the Monitor documentation. This will be adequate as all current and future programs reference the monitor system for its I/O handling. This establishes a large degree of hardware independence for the applications software.

To initially load the monitor, a small "bootstrap" program is required. This program may use any I/O port, with any "Data Available" test bit, and may have any polarity. Once loaded, if standards other than the previously mentioned ones were used, bit-switching will be needed to at least bring up the main console device. Once this is done, the monitor itself may be used to modify the other I/O drivers.

Since the Zapple Monitor is relocatable, it is necessary to tell the loader where it is to be loaded. This is done by setting the SENSE switches to the starting PAGE desired. For example, to load the monitor to run at 0F000H, set the sense switches to 11110000. It is recommended that the monitor be loaded in the highest available location in memory, so that both the monitor and its stack area be "out

of the way" of your other software. Additionally, it is wise to leave some blank memory above the monitor so that your own user routines may be added on. For our in-house systems we typically load the monitor in one 4K board, addressed from F000 to FFFF. Thus, the monitor will be addressed from F000 to F7FF, leaving F800 to FFFF available for additional I/O drivers.

It should be noted that the monitor does NOT require contiguous memory below it. It will function equally well with itself located at F000 and the remainder of your RAM from 0 to 1FFF for example.

TDL software is sent out assuming the monitor resides at F000 (hex) or 360:000 (crazy octal) or 170000 (octal) or 61470 (decimal). If this address is either not possible or inconvenient, you may subtract the size of the monitor (800 hex) from the top address available. For example, if you have a 12K system, and you wish to put the monitor in at 10K, then you would subtract 800 hex from 3000 hex. This means setting the sense switches at 28 Hex.

The LOADER for ZAPPLE has been modified from the one used with the ZAP monitor. It is now easier to set-up for almost any type of device that may be used to read paper tape. The following is the minimum procedure required:

1. Load the following bootstrap program in at address 0.
(This boot program would be used with the old MITS I/O standards. Examples of this and other possible boot loaders are contained in appendix A.)

```
addr   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

0000  C3 1A 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010  00 00 00 00 00 00 00 00 00 00 31 00 02 21 F3 01
0020  CD 2B 00 BD 28 FA 2D 77 20 F6 E9 DB 00 E6 01 20
0030  FA DB 01 C9
```

2. Set the sense switches to the page address where you wish the Monitor to reside.

3. Hit RESET on the processor.

4. Place the tape in the reader device.

5. Start the reader, and WHILE the initial pattern of "1110011" is being read THEN:

6. Hit RUN on the processor.

After the program loads, at the end of the tape it will "sign on" to the console device. If you are using non-standard I/O, (that is, not as per the standard already

stated) it will be sitting in a loop, attempting to sign on. At this time, stop the processor, and modify the routines to handle your own I/O set-up.

This short program actually loads a larger, more sophisticated loader which actually performs the relocating. It is also a checksumed loader. If while loading a checksum error is detected, the Programmed Output lights on an IMSAI will all flash at a rate of about 1 HZ. The machine may be stopped and the tape backed up to an area before the error was detected, (2-3 feet) and the machine reset and started from zero. If you do this, do not change the sense switches.

The relocating checksumed loader listing is given later in this manual. It is similar to the "R" command in the monitor, the primary difference being that the loader at the beginning of the tape gets its address reference from the sense switches. That of the Monitor itself gets the data from the operator console.

COMMANDS

The following is a list of commands for the Zapple Monitor. Precise definitions and usage notes are covered in the next section.

A - ASSIGN reader, punch, console or list device options from the console.
B - BYE (system shut down).
C - COMPARE the contents of memory with the reader input and display any differences.
D - DISPLAY the contents of any defined memory area in Hex.
E - END OF FILE statement generator.
F - FILL any defined area of memory with a constant.
G - GOTO an address and execute. With breakpointing.
H - HEX MATH. Gives the sum and difference of two Hex numbers.
I * USER DEFINED.
J - JUSTIFY MEMORY - a non-destructive test for hard memory failures.
K * USER DEFINED.
L - LOAD a binary file.
M - MOVE a defined memory area to another starting address.
N - NULLS to the punch device.
O * USER DEFINED.
P - PUT ASCII characters into memory from the keyboard.
Q - QUERY I/O ports - may output or input any value to or from any I/O port.
R - READ a Hex file. Performs checksum, relocating, offsetting, etc.
S - SUBSTITUTE and/or examine any value at any address (in hex).
T - TYPEs the contents of a defined memory block in their ASCII equivalent.
U - UNLOAD a binary tape to the punch device.
V - VERIFY the contents of a defined memory block against that of another block and display the differences.
W - WRITE a checksummed hex file to the punch device.
X - eXAMINE and/or modify any or all registers including the special Z-80 registers.
Y - "Yis there". Search memory for defined byte strings and display all the addresses where they are found.
Z - "Z end". Locate and display the highest address in memory.

D. COMMAND SET USAGE

The following section lists the commands, and describes their format and their use. It should be noted that the Zapple Monitor recognizes both upper and lower case letters for its commands, and that in general, a command which is printing can be stopped with a CONTROL C, which is checked during a carriage return - line feed sequence. The following EXAMPLES show a comma [,] as a delimiter between parameters, however a space may also be used. If an error is made while inputting a command from the keyboard, it may be terminated by a rubout and the command re-typed. An asterisk is displayed indicating an ABORT of some kind.

COMMAND — DESCRIPTION

A — ASSIGNMENT OF I/O DEVICES: The monitor system is capable of supporting up to 4 logical devices, these being: The CONSOLE, The READER, the PUNCH, and the LIST DEVICE. To these may be connected 4 different actual I/O devices, for a total of 16 direct combinations of I/O device and function. The specific permutations are:

LOGICAL DEVICE	ASSIGNED DEVICES
CONSOLE	TTY CRT BATCH USER (user defined)
READER	TTY CASSETTE PAPER (HIGH SPEED READER user written) USER (user defined)
PUNCH	TTY CASSETTE PAPER (HIGH SPEED PUNCH user written) USER (user defined)
LIST DEVICE	TTY CRT LINE PRINTER (user written) USER (user defined)

The default mode for each logical device is always the teleprinter.

Assignments are made using the following format:

EXAMPLE: AC=C(cr)

assigns the console equal to the Crt (video terminal) device. similarly:

EXAMPLE: AR=T(cr)

assigns the reader device to be the teleprinter.

While performing a command which requires a reader input (C,L,R), if the assigned reader is the Teleprinter, the software will look for a character from the TTY input. If a character is not recieved within a few seconds, it will ABORT, printing an asterisk [*], and return to the command mode. Similarly, if the assigned reader is the Cassette device, and you WISH to abort for some reason, changing the position of any of the SENSE switches will force an ABORT. On the external reader routines, returning with the carry set indicates an abort (or OUT OF DATA) condition.

When assigning a device, only the first letter initial of its name is required.

The Monitor itself is set-up to support the TTY, CRT and Cassette routines. The other assignments require the addition of user's routines. These are addressed via the commands, which vector to starting addresses.

EXAMPLE: AL=L(cr)

assigns the list device to be the line printer. It vectors to (start address) +812H, or 12H above the end of the monitor. That would be the address for the line printer routine. For details of these arrangements, see the Source Documentation.

Within the above, the assign console equals batch "AC=B(cr)" deserves further mention. In BATCH mode, the READER is made the Keyboard input, and the LIST DEVICE is made the console output. This allows the running of a job directly from the reader input, with the result being output to the list device.

A typical use of this assignment would be the reconstruction of a lengthly text editing job where the text and your editing commands have all been saved on paper tape. With the BATCH MODE, you may assign the reader equals the TTY, the List device equals the TTY, and Console equals BATCH. Running the tape through the reader is the same as you redoing the entire text editing by hand, and the output will go to the TTY and be printed. On a very lengthly job, you could even start the process, and go away until it's done. Its usefullness is limited only by your imagination.

B BYE. This command completly shuts down the system. It is useful where children might have access to the system, where a telephone communications link is established under remote control, or anytime when the operator wishes to make the system inaccessible to unauthorized use.

EXAMPLE: B

completly kills the keyboard. Recovery from the shut-down is accomplished simply by inputting a CONTROL-SHIFT N from the keyboard. (ASCII equivalent is a Record Separator - "RS"; HEX character is a 1EH.) The monitor will sign on and print a greater-than sign (>), however the register storage area will not be cleared.

C COMPARE the reader input with memory. This command is useful for verifying correct loads, verifying that a dumped tape matches with its source etc.

EXAMPLE: C1000,2000(cr,start reader)

compares the memory block 1000H to 2000H with the input from the reader device.

For those with automatic readers, the operation is very simple. Assign the Reader equal to the device you wish to enter the data against, type C(starting address),(ending address)(cr), and the reader will start. The first character read by the reader will be the one matched with the starting address. If any discrepencies are encountered, the reader will stop, and the address (in hex) of the error will be printed on the display. The reader will restart, and continue in this fashion until the entire tape is compared.

If your reader cannot operate automatically, start the reader manually. If an error is encountered, however, while the incorrect address is being printed, the reader will continue, and get "out of sync" with the compare action. Therefore, it is necessary to manually stop the reader if an error is encountered, and manually reposition the tape to the byte following the error. (An excellent article on how to convert ASR33 type readers to automatic operation was recently presented in INTERFACE magazine.)

D DISPLAY memory contents. This command displays the contents of memory in Hex. Memory is displayed

16 bytes per line, with the starting address of the line given as the first piece of data on the line.

EXAMPLE: D100,1FF(cr)

will display in hex the values contained in the memory block 100H to 1FFH.

E END OF FILE. This command generates the end of file pattern for the checksum loader. It is used after punching a block of memory to the punch device using the "W" command. An address parameter for the end of file may be given if so desired.

EXAMPLE: E(cr)

will generate an "end of file marker".

EXAMPLE: E100(cr)

generates the EOF marker with the address parameter "100H". When loading such a file, upon completion, the address contained in the End of File will be placed in the "P" register. Execution of the program may then be initiated by typing "G(cr)".

F FILL command. This command fills a block of memory with a specific value. It is quite handy for initializing a block to a specific value (such as for tests, zeroing memory when starting up, etc.) *NOTE: Avoid doing this over the monitor's stack area. This area may be determined as being between the value you get when typing the Z command, and the value in the S register upon sign-on. It is approximately 60H bytes below the "Top of memory" (Z).

The format for the command is:

EXAMPLE: F100,1FF,FF

fills memory block 100H to 1FFH with the value FFH.

G GOTO command. This command allows the user to cause the processor to GOTO an address and execute the program from that address. In the actual performing of the G command, a program, which has been placed in the stack area during the sign-on of the monitor, is executed. This program will first take all of the values in the register storage area (displayed with the X command), and stuff them in their correct registers in the CPU, and finally JMP to the program address being requested by the

operator. If this short program up in the stack has been destroyed (as a result of a "blow-up", or the F or M commands, etc.) the monitor will not be able to GO anywhere, and a manual restart of the monitor will be required. Whenever the monitor is restarted at the initialization point (first address I.E. 0F000H), the contents of the registers are set to ZERO with the exception of the S (stack), which contains a valid stack address. This actual value depends on the amount of memory in the system, etc. In its simplest form, the letter "G" accompanied by a parameter causes the processor to go to that address and start execution.

EXAMPLE: G1000

would cause the processor to goto address 1000(H) and execute from that address.

Additionally, one or two breakpoints may be set.

EXAMPLE: G1000,1005,1010

would cause the program to start execution at address 1000H, and IN THE EVENT that the program gets to address 1005, OR 1010, the program will stop execution, and return to the monitor, printing an "at" sign, and the address of the breakpoint that was executed. (I.E. @1010) It then prints the ">" prompt, awaiting further instructions. This action also cancels any breakpoints previously set.

Breakpoints must be set at locations containing an instruction byte. This is a SOFTWARE breakpoint system, and requires either RAM at RST 7 (restart 7, addr. 0038H), or if using ROM, a permanent JMP to the monitor TRAP address (0F01EH) at 0038H. Remember, this is a SOFTWARE breakpoint system, and the program being debugged must be in non-protected Read/Write memory.

```
EXAMPLE:    *C2   JNZ   1234H
             34
             12
            *3E   MVI   A,CR
             0D
            *21   LXI   H,1000H
             00
             10
            *77   MOV   M,A
            *23   INX   H
            *CD   CALL  5678H
             78
             56
```

The asterisks (*) mark the bytes that may be used as breakpoints.

H HEX MATH. This command allows the execution of hexidecimal arithmetic directly from the console. it will give the sum and difference of any two hex numbers entered.

```
EXAMPLE:    H1000,1010(cr)
            2010 FFF0
            >
```

2010H being the sum, and FFF0 being the difference of the two hex values.

J The J command is a non-destructive memory test. The command reads any given byte, complements it, writes into the location the complement, compares the complement with the accumulator, and rewrites the original byte into the location. The command is used with two parameters, delineating the block of memory to be checked.

```
EXAMPLE:    J1000,1FFF
```

would perform the above test on the block 1000H to 1FFFH.

If errors are detected, the address at which the error is found and the error are displayed on the console before the test is continued.

```
EXAMPLE:    J1000,1FFF(cr)
            1F00  00001000
            >
```

would indicate that the 4th bit (D3) at location 1F00H did not correctly complement itself.

This test is useful for the discovery of hard memory failures, and also serves as a quick check for accidentally protected memory. A fully protected memory block would print out as entirely "1s". (11111111)

L LOAD BINARY FILE. This command loads a binary file from either a cassette or paper tape.

```
EXAMPLE:    L1000(cr)
```

would load the tape at address 1000H. This would require that the program be an absolute program, designed for address 1000H. The start-of-file mark (automatically generated by the "U" command) is a series of 8 0FFH's (rubouts). When this is detected at the start of file, the bell will ring on the TTY to indicate the start of the load process. When the end-of-file is detected (again, a series of 8 rubouts) the load is terminated, and the address of

the NEXT location that would have been loaded is printed on the console. There are two constraints on this type of file system. The middle of the program cannot contain more than 6 0FF's (11111111) in a row (an unusual occurence), and if 0FFH is the LAST data byte in the file, it will be ignored. This too is unusual, and only a minor inconvenience.
Binary programs loaded at other than their design address will not run. The "L" command does not perform checksum functions, and cannot handle relocatable files. This is a pure and simple byte-for-byte binary loader (see "U" command).

M MOVE COMMAND. This command is used to move a block of memory from one location to another. The original block is NOT affected by the move, remaining intact so long as the block moved into does not overlap with the block currently occupied. This command, like the "F" command should be used with some caution as moving a block into an area occupied by the stack, or the program or the monitor will cause unpredictable results.

EXAMPLE: M1000,1FFF,2000(cr)

moves the contents of memory contained in the block 1000H to 1FFFH to a starting address of 2000H. The new block has the limits 2000H to 2FFFH.

This command is very useful for working on programs without destroying the original, verifying blocks of memory loaded with existing memory, etc.

N NULL. This command punches nulls to the punch device. 72 nulls are punched whenever the command is used. It may be used repetitively for any desired leader length.

EXAMPLE: (N)
*Note: The "N" or "n" will NOT echo, so as to not spoil the paper tape.

It will punch 72 nulls to the punch device.

P PUT ASCII characters into memory. This command allows ASCII characters to be written directly into memory. It is useful for placing labels in files etc.

EXAMPLE: P1000(cr)

activates the command, and any further inputs via the keyboard would be placed into memory in their ASCII equivalent. The command is terminated by a CONTROL D character, with the address of the

location following the last entry printed on the console (the Control-D is NOT stored). Recovery of the input data is affected by use of the "T" or "U" command.

Q QUERY INPUT/OUTPUT PORTS. This command allows any value to be output to any I/O port, and allows the value in binary on any I/O port to be read on the console.

EXAMPLE: QO1,7(cr)

would output an ASCII "7" to I/O PORT 1. (ASCII seven is a "bell" so on a TTY, the bell would ring.)

EXAMPLE: QI1(cr) 00001101

inputs the value at port 1, in the illustration above, we see that bits 0,2 and 3 are high, the others low. This is useful for observing the condition of status bits and other diagnostic activities.

R READ A CHECKSUMMED HEX FILE. This command reads checksummed hex files in the INTEL format, as well as being capable of loading the relocatable TDL files at any selected address and bias offset. When reading an ABSOLUTE file (INTEL format), there may be only a BIAS added. These files cannot be relocated. The format is: R[bias],[relocation](cr).

If a checksum error or a failure to write the data to memory occurs, the loading process is stopped, an asterisk is printed (indicating some error condition), and the address that was attempting to be written will be displayed on the console device. This is to assist in determining the failure.

EXAMPLE: R(cr, start reader)

will load a hex file at its absolute address.

EXAMPLE: R,1000(cr,start reader)

will load a TDL relocatable hex file at address 1000H and modify the program to run at address 1000H.

EXAMPLE: R1000,100(cr,start reader)

loads the file set up to run at 100H, but with a positive BIAS of 1000H added to it. Thus, the file, set up to run at 100H will be loaded at 1100H.

EXAMPLE: R1000(cr)

will load the file, set up to run at address 0000H, at address 1000. In other words, using the TDL relocating format, you may load any program, to execute anywhere in memory, anywhere in memory. (Think about it.....)

S SUBSTITUTE and examine. This command allows any address in memory to be examined directly, and allows substituion of one value for another at that address if desired.

EXAMPLE: SF810(sp)00-(sp)1A-(sp)C3-(sp)(cr)
>

In this case the "S" command examines address F810H. The hitting of the space bar (sp) displays the value at that address. (assuming value 00H at that address.) Hitting the space bar again displays the NEXT location in memory (F811H), and so forth. Simply typing S(sp) starts display from address 0000H. By repetitive typing of (sp), all of memory could be displayed one address at a time.

EXAMPLE: SF810(sp)00-(kb)FF(cr)

This command examines address F810H, showing the value 00H at that address. Immediately typing in FFH from the keyboard SUBSTITUTES FFH for 00H at that address. Repeating the example above would show:

EXAMPLE: SF810(sp)FF-

When an address is being examined, the address being examined may be moved BACKWARD by entering a backarrow (ba) or SHIFT-O, or underline, depending on the terminal used.

EXAMPLE: SF810(sp)00-(ba)AA-

shows that at address F80FH, the value AA exists. Typing a space bar will examine F810H again.

T TYPE ASCII characters from memory. This command allows the contents of memory to be displayed in their ASCII equivalents. All non-printing characters will be displayed as periods [.]. It is may used to display the results of the "P" command which allows keyboard entry of ASCII characters directly into memory. Also useful for finding text strings and messages in software. The initial address is first displayed, then the first 64 characters, the next address, etc. until the upper limit has been reached.

EXAMPLE: T1000,2000(cr)

displays the ASCII equivalents of memory locations 1000H to 2000H. If the "P" command had been used to place a "message" into memory somewhere in that memory block, it would soon be apparent on the console display.

U UNLOAD BINARY. This command simply dumps core to the punch device. It may be used with a cassette system as well, with no start-up problems. It does not generate a checksum. The format which is generated will be a leader, eight 0FFHs, binary data, eight 0FFHs, and a trailer. The 0FFHs are "rubouts" and are called file cues. These are detected and counted to determine the start and the end of files.

EXAMPLE: U00,FF(cr,start reader)

will generate a binary tape, formated as described above, of the values contained in memory locations 00H to FFH.

V VERIFY. This command allows the user to verify the contents of one memory block against the contents of another memory block. This is very useful for functions such as verifying that a file generated from a program is a duplicate of the actual program, etc.

EXAMPLE: V1000,2000,3000

will compare the contents of the memory block 1000H to 2000H against the contents of the memory block commencing at 3000H and extending to 4000H. Any differences will be displayed.

```
EXAMPLE:    V1000,2000,3000
            100F  00 FF
```

indicates that the contents of address 100FH is a 00 while that at 300FH is an FF.

W WRITE Hex file. This command dumps memory to the punch device in the standard "Intel-style" hex file format. Both start and end of file parameters are required. The proper "end of file" (EOF) is generated by the "E" command.

```
EXAMPLE:    W00,FF(cr,start punch)
            (after punching)
            E(cr)
```

will generate a checksummed hex file of the values in the memory block 00H to FFH. If the assigned punch and console are the same, the program will pause and wait for the operator to turn on the punch (ASR33, etc.). Use of the "N" command at either the beginning and/or end of the file is optional, but recommended.

X eXAMINE REGISTERS. The "X" command allows the user to examine and/or modify all of the Z80 registers.

```
A=Accumulator
B,C,D,E,H,L=CPU REGISTERS
M=Memory (pointed to by H&L)
P=Program Counter (PC)
S=Stack Pointer (SP)
I=Interrupt Register
X=Index (IX)
Y=Index (IY)
R=Refresh Register
```

EXAMPLE: X(cr)

displays the contents of MAIN registers A,B,C,D,E,F,H,L,M,P,S and I, in hex.

EXAMPLE: X'(cr)

displays the contents of PRIME registers A,B,C,D,E,F,H,L,M,X,Y and R.

Typing the letter "X" (or X'), followed by a specific register letter will display the contents of that register. Entering a new value via the keyboard (kb) will substitute the new value in the specific register. Hitting the space bar will display the next register in which you may then perform substitutions, etc. In the unique case of the "M" register, you may modify the 16 bit pointer (H&L) to that memory location.

```
EXAMPLE:    XA 00-(kb)FF(cr)
            XA FF-(sp)00-(kb)FF(cr)
            XA FF-(sp)FF-(cr)
            >
```

first examines the contents of register "A" (00H), then substitutes an FF. In the next line, the FF is displayed, a space character displays the next register (again a 00H), and substitutes an FF for this value. The last line displays both registers as containing FFHs.

Y SEARCH. This command allows unique byte strings, from one up to 255 bytes to be searched for in

memory, and the addresses where they are found to be displayed. It is advisable to search for unique patterns rather than single bytes. The search operation may be stopped with a control-C.

```
EXAMPLE:    YC3,21,F3,01(cr)
            0081
            00B2
            0F08
            >
```

indicates that the byte string (in hex) C3, 21, F3, 01, is found in memory at locations 0081H, 00B2H and 0F08H. This routine will search all 65-K of memory for a unique sequence of bytes in less than one second.

Z — Z TOP OF MEMORY. This command locates and gives the highest address of available memory in your system.

```
EXAMPLE:    Z
            7FFF
            >
```

indicates that the highest available memory is at address 7FFFH. Note that NO carriage return is required. Also, If only one 1K board were in the system, and it was adressed to have its top byte at address 7FFFH, the Z command would so indicate regardless of the absence of lower memory.

ZAPPLE SOURCE DOCUMENTATION

ZAPPLE was assembled using TDL's Relocating Macro Assembler. In the event that you are not familiar with it's format, here is a brief description.

If you are familiar with the 8080 INTEL mnemonics, you have a head start. We at TDL have tried to make the cross-over from the 8080 to the Z-80 as painless as possible, and have used all of the previous OP-CODE mnemonics which were compatable between the 8080 & Z-80. In addition, any obvious extensions were used to simplify learning of the new Z-80 op-codes. For example, just as in the 8080 you have a "LHLD" for "Load H&L Direct", in the Z-80 there is also "LBCD" for "Load B&C Direct", and "LDED" for "Load D&E Direct", etc.

The ZAPPLE is assembled in a RELOCATING format. Therefore, the assembly listing starts effectively at address ZERO. In order to calculate the correct addresses, the value used while loading ZAPPLE (F000H in most cases) must be ADDED to the addresses. This would normally affect the high byte only, assuming that it was loaded on a page border. In the listing, any value that is of a "relocatable" nature is tagged with an apostrophe. Absolute values do not have an apostrophe following them. The assembler also lists the 16 bit values in their TRUE representation, rather than switching the high & low bytes. The actual object tape generated contains the bytes reversed as the 8080 & Z-80 require.

For further information on the format used in this listing, please refer to the "TDL RELOCATING MACRO ASSEMBLER MANUAL", available from TDL.

EXPERIMENTING WITH ZAPPLE

One thing that is rather nice about playing with computer programs is that you can experiment, manipulate, discect, make mistakes, 'blow them up', etc., and when the patient dies (or is "POKED TO DEATH"), he can be bought back to life by simply re-loading the program!

Please feel free to examine and modify this monitor to suit your tastes and needs. The most important thing to avoid changing however is the monitor VECTORS, and the RULES regarding them. They are:

1. Any I/O operation (CI, RI, CO, PO, etc.) should modify only the "A" register. When outputting, the character is passed in "C", and should be in "A" upon returning. When inputting, the character is returned in "A" register. *NOTE: On the "RI" Vector, the carry is normally cleared unless there is no more data to be obtained from the reader device, at which time the carry is SET to indicate an OUT OF DATA condition.

2. CSTS. This routine modifies only the contents of "A" register. It will make "A" equal to ZERO if there are no characters waiting at the assigned console input, and 0FFH if there ARE characters waiting. We are talking about the CONTENTS of "A", not the flags. The calling program would then test the contents of "A" with perhaps an "ORA A" instruction, for example, and if the result was non-zero, it would indicate a CHARACTER WAITING condition at the console keyboard.

3. IOCHK/IOSET. Allows applications software to dynamically change the I/O configuration. Any new configuration is passed in "C" reg. when IOSET is called, and the current configuration is returned in "A" reg. when IOCHK is called. *NOTE: The program in the monitor that allows modifying and assigning various I/O devices uses a R/W I/O port (one I/O port with the input tied to the output). However, the program may be modified to use a specific RAM location to store the 8-bit value. The later involves changing the IOSET/IOCHK routines accordingly. For example: "CMA, OUT 2" becomes "STA 0F8FFH", and "IN 2, CMA" becomes "LDA 0F8FFH". The use of the R/W I/O port is preffered, as it is much less sensitive to being accidentally altered during a de-bugging session, or if the program goes nuts, etc. Also, the port just above the R/W one is used (hardwired) to indicate the I/O configuration desired upon monitor initialization (may be changed to a "MVI A,XX", where XX is the desired asssignment pattern.)

This whole scheme is easily accomplished using a "3P+S" board or equivalent. (see listing for any software

details).

4. MEMCK. This routine modifies only the "A" & "B" registers. It is used to allow an applications program to find out how much memory it may use. It will load the A & B registers with the highest value of CONTINOUS memory (starting from zero) MINUS the area needed for the monitor to function properly. (A=low byte, B=high byte). This value is also placed in the STACK register when the monitor is initialized. This is then used as an initial stack value (when a "GO" command is first issued), in case the programmer has forgotten to initialize the stack. (also see "X" command).

USER WRITTEN COMMAND ROUTINES.

There are 3 command letters left open for your use. They are "I","K", & "O". Both "I" & "O" are naturals for implementing custom I/O routines. (That's what this monitor is all about.) "K" is left for your own imagination. The locations in the command table NOW contain the vector for the ERROR routine. However, in the listing, vectors to the 0F800H block are given, and should be patched to those vectors as the commands are implemented. Then, JMPs to the ACTUAL routines should be placed in the 0F800H portion. At the conclusion of the CUSTOM COMMAND, a RET instruction will return to the normal monitor command loop, printing the ">" prompt. The ideal situation, once you have settled on your own customizing of the monitor, is for the monitor to be in ROM from 0F000H to 0F7FFH (2-K ROM BOARD), and then RAM from 0F800H on upward to a maximum of 0FFFFH. (This sounds like a good use for those old 1-K static memory cards!)

USER WRITTEN I/O ROUTINES.

There are occasions when some device needs a specialized piece of software in order to make it work. Line printers, parallel keyboards, punches, optical readers, etc. These will have to be handled on an individual basis. The general idea is to NOT MODIFY any registers other than those mentioned above, and to NOT upset the stack pointer. Things may be pushed during the routine in order to avoid modifying the other registers, as long as the POP's match the PUSH's. All routines that are vectored out of the monitor should end with a RET instruction. Remember to clear the carry before returning from a USER defined "RI" routine, unless you are intending to indicate an OUT-OF-DATA condition. In that case, you SHOULD set the carry flag before returning (STC).

Using MEMORY as a Reader/Punch device can also be very useful. Here is an example of how this might be accomplished:

```
MEMRD:  PUSH   H       ;FIRST SAVE H&L
        LHLD   01EH    ;PICK UP A POINTER
        MOV    A,M     ;GET MEMORY BYTE
        INX    H
        SHLD   01EH    ;REPLACE POINTER
        POP    H       ;RESTORE H&L
        ORA    A       ;INSURE CARRY CLEAR
        RET            ;ALL DONE
```

```
MEMWR:      PUSH    H       ;SAVE H&L
            LHLD    01CH    ;OUTPUT POINTER
            MOV     M,C     ;STORE OUTPUT BYTE
            INX     H
            SHLD    01CH    ;REPLACE POINTER
            POP     H       ;RESTORE H&L
            MOV     A,C     ;FOLLOW THE RULES
            RET             ;ALL DONE
```

There are many variations of the above, and will depend on the configuration of your system, etc.

Any reasonable SPECIFIC questions regarding interfacing other devices, software, etc., which are sent to TDL, IN WRITING, will be looked at and answered within a reasonable period of time, either by return mail, or in the USER'S GROUP newsletter.

It is an almost impossible task to fully cover all of the intricate details involved in the operation of ZAPPLE. The best thing you can do now is re-read this entire manual, and then start experimenting on your own. You will have to use some common-sense if a particular subject has not been fully explained. As any lackings in this manual become evident, they WILL be covered in the NEWSLETTERS to follow. We also appreciate your feedback, and feel free to write and complain (or praise!) us about this manual or any other TDL product. YOU help US, and we'll help YOU. But most of all.....

HAVE FUN!

Roger Amidon,
TECHNICAL DESIGN LABS, INC.
RESEARCH PARK Bldg. H
PRINCETON, NEW JERSEY
08540

APPENDIX A. SUPPORT PROGRAMS FOR RELOCATING BOOT LOADER, V3.2
GENERAL DESCRIPTION

```
.LIST
.REMARK /
   THIS VERSION OF THE TDL BOOT LOADER AND
TDL RELOCATING LOADER SHOULD MAKE IT EASIER
FOR PEOPLE WITH WIDELY DIVERGENT HARDWARE
TO LOAD THE MONITOR.

THE GENERAL MEMORY MAP LOOKS LIKE THIS:
0000 - 00FF  BOOT LOADER
0100 - 01FF  RELOCATING LOADER
0200 - FFFF  WHERE MONITOR MAY BE PLACED

THE BOOT LOADER MEMORY MAP:
0000 - 0019  HARDWARE INITIALIZATION ROUTINE
001A - 001C  LXI SP,200H
001D - 001F  LXI H,01F3H (CHANGED BY UPPER LOADER)
0020 - 0022  CALL READER (CALL CHANGED TO JMP)
0023 - 00FF  BOOT LOADER AND READER ROUTINES

THE THREE INSTRUCTIONS SHOWN IN THE BOOT LOADER
MEMORY MAP ARE FIXED AND MUST BE AS SHOWN,
BECAUSE THE RELOCATING LOADER USES OR MODIFIES
THEM.

THE READER ROUTINE IS EXPECTED TO RETURN AN
8 BIT CHARACTER FROM THE TAPE EACH TIME IT
IS CALLED.

THE BOOT LOADER ROUTINE LOADS THE RELOCATING
LOADER INTO MEMORY STARTING AT 01F3H AND
DOWNWARD TO 0100H.
/
.PAGE
```

UART STYLE BOOT LOADER ROUTINES

```
                        .LIST
                        ;
                        ;
0000  C31A00            ..INIT: JMP     ..LOAD  ;NO INITIALIZATION NEEDED
                        ;
001A                    .LOC 1AH
                        ;
001A  310002            ..LOAD: LXI     SP,200H ;SET STACK
001D  21F301                    LXI     H,01F3H ;LOAD LOADER
0020  CD2B00            ..RDR:  CALL    ..READ  ;GET A CHARACTER
0023  BD                        CMP     L       ;TEST LEADER
0024  28FA                      JRZ     ..RDR   ;WALK OVER LEADER
0026  2D                        DCR     L       ;MOVE POINTER
0027  77                        MOV     M,A     ;SAVE DATA
0028  20F6                      JRNZ    ..RDR   ;GET MORE DATA OR
002A  E9                        PCHL            ; GO TO LOADER
                        ;
                        ;  ALTAIR SIOA REV 1.0 READER ROUTINE
                        ;
002B  DB00              ..READ: IN      0       ;STATUS PORT
002D  E601                      ANI     1       ;DATA AVAILABLE BIT
002F  20FA                      JRNZ    ..READ  ;0=DATA AVAILABLE
0031  DB01                      IN      1       ;DATA PORT
0033  C9                        RET             ;DONE

                        ;
                        .LIST
                        ;
                        ;  PTCO 3P+S READER ROUTINE
                        ;
002B  DB00              ..READ: IN      0       ;STATUS PORT
002D  E640                      ANI     040H    ;DATA AVAILABLE BIT
002F  28FA                      JRZ     ..READ  ;1=DATA AVAILABLE
0031  DB01                      IN      1       ;DATA PORT
0033  C9                        RET             ;DONE
                        ;
                        ;
                        .PAGE
```

```
                        .LIST
                        ;
                        ;
                        ; THIS ROUTINE WOULD BE USED FOR AN I/O BOARD
                        ; THAT USES A MOTOROLA ACIA.
                        ; SUCH AS AN ALTAIR 2SIO.
                        ;
0000  3E03      ..INIT: MVI     A,003H  ;RESET
0002  D320              OUT     20H
0004  3E11              MVI     A,011H  ;CLOCK/16, 8 DATA BITS
0006  D320              OUT     20H     ;NO PARITY
0008  C31A00            JMP     ..LOAD
                        ;
001A                    .LOC 1AH
                        ;
001A  310002    ..LOAD: LXI     SP,200H ;SET STACK
001D  21F301            LXI     H,01F3H ;LOAD LOADER
0020  CD2B00    ..RDR:  CALL    ..READ  ;GET A CHARACTER
0023  BD                CMP     L       ;TEST LEADER
0024  28FA              JRZ     ..RDR   ;WALK OVER LEADER
0026  2D                DCR     L       ;MOVE POINTER
0027  77                MOV     M,A     ;SAVE DATA
0028  20F6              JRNZ    ..RDR   ;GET MORE DATA OR
002A  E9                PCHL            ; GO TO LOADER
                        ;
                        ; READER ROUTINE
                        ;
002B  DB20      ..READ: IN      20H     ;STATUS PORT
002D  E601              ANI     1       ;DATA AVAILABLE BIT
002F  28FA              JRZ     ..READ  ;1=DATA AVAILABLE
0031  DB21              IN      21H     ;DATA PORT
0033  C9                RET             ;DONE
                        ;
                        ;
                        .PAGE
```

INTEL USART BOOT LOADER ROUTINE

```
                        .LIST
                        ;
                        ;
                        ; THIS ROUTINE WOULD BE USED FOR AN I/O BOARD
                        ; THAT USES AN INTEL USART.
                        ; SUCH AS AN IMSAI 2SIO.
                        ;
0000  3ECE      ..INIT: MVI     A,0CEH  ;CLOCK/16, 8 DATA BITS
0002  D303              OUT     3       ;NO PARITY, 2 STOP BITS
0004  3E17              MVI     A,017H  ;ENABLE XMIT & REC
0006  D303              OUT     3       ;RESET ERROR FLAGS
0008  C31A00            JMP     ..LOAD
                        ;
001A                    .LOC 1AH
                        ;
001A  310002    ..LOAD: LXI     SP,200H ;SET STACK
001D  21F301            LXI     H,01F3H ;LOAD LOADER
0020  CD2B00    ..RDR:  CALL    ..READ  ;GET A CHARACTER
0023  BD                CMP     L       ;TEST LEADER
0024  28FA              JRZ     ..RDR   ;WALK OVER LEADER
0026  2D                DCR     L       ;MOVE POINTER
0027  77                MOV     M,A     ;SAVE DATA
0028  20F6              JRNZ    ..RDR   ;GET MORE DATA OR
002A  E9                PCHL            ; GO TO LOADER
                        ;
                        ; READER ROUTINE
                        ;
002B  DB03      ..READ: IN      3       ;STATUS PORT
002D  E602              ANI     2       ;DATA AVAILABLE BIT
002F  28FA              JRZ     ..READ  ;1=DATA AVAILABLE
0031  DB02              IN      2       ;DATA PORT
0033  C9                RET             ;DONE
                        ;
                        ;
                        .PAGE
```

```
                        .LIST
                        ;
                        ; THIS IS AN EXAMPLE OF A ROUTINE THAT
                        ; "MIGHT" BE USED TO CONTROL A PARALLEL
                        ; READER.
                        ;
0000  3E20      ..INIT: MVI     A,20H   ;INITIALIZE THE HARDWARE
0002  D31B              OUT     01BH
0004  3E30              MVI     A,30H
0006  D31B              OUT     01BH
0008  3E28              MVI     A,28H
000A  D31B              OUT     01BH
000C  3E20              MVI     A,20H
000E  D31B              OUT     01BH
0010  C31A00            JMP     ..LOAD
                        ;
001A                    .LOC 1AH
                        ;
001A  310002    ..LOAD: LXI     SP,200H ;SET STACK
001D  21FE01            LXI     H,01FEH ;LOAD LOADER
0020  CD2B00    ..RDR:  CALL    ..READ  ;GET A CHARACTER
0023  BD                CMP     L       ;TEST LEADER
0024  28FA              JRZ     ..RDR   ;WALK OVER LEADER
0026  2D                DCR     L       ;MOVE POINTER
0027  77                MOV     M,A     ;SAVE DATA
0028  20F6              JRNZ    ..RDR   ;GET MORE DATA OR
002A  E9                PCHL            ; GO TO LOADER
                        ;
                        ; READER ROUTINE
                        ;
002B  3E20      ..READ: MVI     A,20H
002D  D31B              OUT     1BH
002F  3E30              MVI     A,30H
0031  D31B              OUT     1BH
0033  DB1B      ..LOOP: IN      1BH     ;STATUS
0035  E601              ANI     1
0037  28FA              JRZ     ..LOOP
0039  DB1A              IN      1AH     ;DATA
003B  2F                CMA             ;UPSIDE DOWN
003C  F5                PUSH    PSW
003D  3E28              MVI     A,28H
003F  D301              OUT     1B
0041  3E20              MVI     A,20H
0043  D31B              OUT     1BH
0045  F1                POP     PSW
0046  C9                RET
                        ;
                        ;
                        .END
```

```
                        ;
                        ;
                        .TITLE   / APPENDIX B.     <*TDL RELOCATING LOADER, VERSION
                         3.2 - DEC. 28, 1976*>/
                        ;
                        ;       STAND-ALONE VERSION, TO BE USED
                        ;       AS A BINARY BOOT-STRAP LOADER.
                        ;
                        .PABS           ;ABSOLUTE ASSEMBLY
                        ;
00FF                    SENSE   = 0FFH  ;ALTAIR/IMSAI/TDL/ETC SENSE SWITCHES
001E                    HLMOD   = 01EH  ;ADDRESS MODIFIED TO A JMP
0020                    USER    = 0020H ;USER WRITTEN I/O ROUTINE
0200                    TOP     = 0200H ;STACK AREA
                        ;
0100                    .LOC    100H    ;LOADER ON PAGE ONE
                        ;
                        ;       SET-UP
                        ;
0100  3EC3              BEGIN:  MVI     A,JMP   ;IN CASE OF TROUBLE
0102  32 001D                   STA     HLMOD-1 ; STORE A JMP TO HERE
0105  21 0100                   LXI     H,BEGIN ; AT BOTTOM
0108  22 001E                   SHLD    HLMOD   ;
                                                ;
010B  32 0020                   STA     USER    ;MODIFY READER CALL
                                                ; TO A JMP
010E  31 0200                   LXI     SP,TOP  ;INSURE A STACK
0111  DBFF                      IN      SENSE   ;SEE WHERE TO LOAD
0113  FE02                      CPI     2       ;CAN'T BE LESS THAN PAGE 2
0115  DA 0159                  JC      ERROR   ;ABORT IF SO
0118  47                      MOV     B,A     ;SAVE RELOCATION
0119  0E00                    MVI     C,0     ;FORCE PAGE BORDER
011B  D9                      EXX             ;SAVE IT IN BC'
                        ;
                        ;       ACTUAL LOADER CODE
                        ;
011C  CD 01BE           LOD0:   CALL    RDR     ;GET A CHARACTER
011F  D63A                      SUI     ':'     ;ABSOLUTE FILE?
0121  47                      MOV     B,A     ;SAVE INFO
0122  E6FE                    ANI     0FEH    ;KILL BIT ZERO
0124  20F6                    JRNZ    LOD0    ;FILE NOT STARTED YET
0126  57                      MOV     D,A     ;ZERO CHECKSUM
0127  CD 01A0                 CALL    SBYTE   ;GET FILE LENGTH
012A  5F                      MOV     E,A     ;SAVE IN E
012B  CD 01A0                 CALL    SBYTE   ;LOAD MSB
012E  F5                      PUSH    PSW     ;SAVE IT
012F  CD 01A0                 CALL    SBYTE   ;LOAD LSB
0132  E1                      POP     H       ;H=MSB
0133  6F                      MOV     L,A     ;L=LSB
0134  E5                      PUSH    H
0135  DDE1                    POP     X       ;INDEX X=LOAD ADDR
0137  D9                      EXX             ;ALTERNATE REG.'S
0138  C5                      PUSH    B       ;BC'=RELOCATION
0139  D9                      EXX
013A  CD 01A0                 CALL    SBYTE   ;GET FILE TYPE
```

```
013D  3D              DCR     A       ;1=REL. 0=ABS.
013E  78              MOV     A,B     ;GET OLD INFO
013F  C1              POP     B       ;RELOCATION FACTOR
0140  2003            JRNZ    ..A     ;MUST BE ABSOLUTE LOAD
0142  DD09            DADX    B       ;ELSE RELOCATE
0144  09              DAD     B       ; BOTH HL & X
0145  1C      ..A:    INR     E       ;TEST LENGTH
0146  1D              DCR     E       ;0=DONE
0147  2822            JRZ     DONE
0149  3D              DCR     A       ;TEST OLD INFO
014A  2824            JRZ     LODR    ;RELATIVE FILE
014C  CD 01A0   ..L1:   CALL    SBYTE   ;NEXT...
014F  CD 01C4           CALL    STORE   ;STORE IT
0152  20F8              JRNZ    ..L1    ;MORE COMING
0154  CD 01A0   LOD4:   CALL    SBYTE   ;GET CHECKSUM
0157  28C3              JRZ     LOD0    ;ALL O.K.
                ;
0159  AF        ERROR:  XRA     A       ;FLASH ADDRESS & SENSE LINES
015A  2F                CMA
015B  D3FF              OUT     SENSE
015D  1B        ..SIT1: DCX     D
015E  7A                MOV     A,D
015F  B3                ORA     E
0160  20FB              JRNZ    ..SIT1
0162  D3FF              OUT     SENSE
0164  1B        ..SIT2: DCX     D
0165  7A                MOV     A,D
0166  B3                ORA     E
0167  20FB              JRNZ    ..SIT2
0169  18EE              JMPR    ERROR
                ;
                ;
016B  7C        DONE:   MOV     A,H
016C  B5                ORA     L       ;CAN'T GO TO ZERO
016D  28FE              JRZ     .       ;TIGHT LOOP HERE
016F  E9                PCHL            ;ELSE SIGN ON PROGRAM
                ;
0170  2E01      LODR:   MVI     L,1
0172  CD 0190   ..L1:   CALL    LODCB   ;GET CONTROL BYTE
0175  3807              JRC     ..L3    ;DOUBLE BIT
0177  CD 01C4   ..L5:   CALL    STORE   ;WRITE IT
017A  20F6              JRNZ    ..L1    ;MORE TO GO
017C  18D6              JMPR    LOD4    ;TEST CHECKSUM
                ;
017E  4F        ..L3:   MOV     C,A     ;LOW BYTE
017F  CD 0190           CALL    LODCB   ;NEXT
0182  47                MOV     B,A     ;HIGH BYTE
0183  D9                EXX
0184  C5                PUSH    B       ;GET RELOCATION
0185  D9                EXX
0186  E3                XTHL
0187  09                DAD     B
0188  7D                MOV     A,L     ;RELOCATE LOW BYTE
0189  CD 01C4           CALL    STORE   ;SAVE IT
018C  7C                MOV     A,H     ;RELOCATED HIGH BYTE
```

```
018D  E1              POP     H       ;RESTORE HL
018E  18E7            JMPR    ..L5    ;SAVE HIGH, REPEAT
                ;
0190  2D        LODCB:DCR     L       ;COUNT BITS
0191  2007           JRNZ    ..LC1   ;MORE LEFT
0193  CD 01A0         CALL    SBYTE   ;GET NEXT
0196  1D              DCR     E       ;COUNT BYTES
0197  67              MOV     H,A     ;SAVE THE BITS
0198  2E08            MVI     L,8     ;8 BITS/BYTE
019A  CD 01A0   ..LC1:CALL    SBYTE   ;GET A DATA BYTE
019D  CB24            SLAR    H       ;TEST NEXT BIT
019F  C9              RET
01A0  C5        SBYTE:PUSH    B       ;PRESERVE BC
01A1  CD 01B3         CALL    RIBBLE  ;GET 1/2 BYTE
01A4  07              RLC
01A5  07              RLC
01A6  07              RLC
01A7  07              RLC
01A8  4F              MOV     C,A     ;SAVE LEFT HALF
01A9  CD 01B3         CALL    RIBBLE  ;GET OTHER HALF
01AC  B1              ORA     C       ;MAKE WHOLE
01AD  4F              MOV     C,A     ;IN C
01AE  82              ADD     D       ;UPDATE CHECKSUM
01AF  57              MOV     D,A     ;NEW VALUE
01B0  79              MOV     A,C     ;CONVERTED BYTE
01B1  C1              POP     B
01B2  C9              RET
                ;
01B3  CD 01BE   RIBBLE:CALL   RDR
01B6  D630            SUI     '0'
01B8  FE0A            CPI     10
01BA  D8              RC
01BB  D607            SUI   'A'-'9'-1 ;ADJUST
01BD  C9              RET
                ;
01BE  CD 0020   RDR:  CALL    USER    ;USER WRITTEN ROUTINE AT 10H
01C1  E67F            ANI     7FH
01C3  C9              RET
                ;
01C4  DD7700    STORE:MOV     0(X),A  ;WRITE TO MEMORY
01C7  DDBE00          CMP     0(X)    ;VALID WRITE?
01CA  208D            JRNZ    ERROR   ; NO.
01CC  DD23            INX     X       ;ADVANCE POINTER
01CE  1D              DCR     E       ;DECREMENT COUNT
01CF  C9              RET
                ;
                .END
```

APPENDIX B. <*TDL RELOCATING LOADER, VERSION 3.2 - DEC. 28, 1976*>
+++++ SYMBOL TABLE +++++

BEGIN	0100	DONE	016B	ERROR	0159	HLMOD	001E
LOD0	011C	LOD4	0154	LODCB	0190	LODR	0170
RDR	01BE	RIBBLE	01B3	SBYTE	01A0	SENSE	00FF
STORE	01C4	TOP	0200	USER	0020		

ADDENDUM:

Here is a DUMP of the LOADER, Version 3.2. It may be used to insure proper loading after the boot part of the tape has been read. This should not be required unless you are having trouble loading the monitor.

Remember: The new format requires the monitor be loaded at 0200H minimum. We strongly urge that you load at 0F000H. If you still wish to locate the monitor between 0 and 0200H, first load a temporary copy up higher, and then use THAT one to load it elsewhere. This monitor runs ANYWHERE when loaded by a copy of itself, but when using an initial boot strap, it is forced to a page boundry. Running the monitor on other than a page border sounds a little pointless in any case.

addr	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0100	3E	C3	32	1D	00	21	00	01	22	1E	00	32	20	00	31	00
0110	02	DB	FF	FE	02	DA	59	01	47	0E	00	D9	CD	BE	01	D6
0120	3A	47	E6	FE	20	F6	57	CD	A0	01	5F	CD	A0	01	F5	CD
0130	A0	01	E1	6F	E5	DD	E1	D9	C5	D9	CD	A0	01	3D	78	C1
0140	20	03	DD	09	09	1C	1D	28	22	3D	28	24	CD	A0	01	CD
0150	C4	01	20	F8	CD	A0	01	28	C3	AF	2F	D3	FF	1B	7A	B3
0160	20	FB	D3	FF	1B	7A	B3	20	FB	18	EE	7C	B5	28	FE	E9
0170	2E	01	CD	90	01	38	07	CD	C4	01	20	F6	18	D6	4F	CD
0180	90	01	47	D9	C5	D9	E3	09	7D	CD	C4	01	7C	E1	18	E7
0190	2D	20	07	CD	A0	01	1D	67	2E	08	CD	A0	01	CB	24	C9
01A0	C5	CD	B3	01	07	07	07	07	4F	CD	B3	01	B1	4F	82	57
01B0	79	C1	C9	CD	BE	01	D6	30	FE	0A	D8	D6	07	C9	CD	20
01C0	00	E6	7F	C9	DD	77	00	DD	BE	00	20	8D	DD	23	1D	C9

```
                        ;       << ZAPPLE 2-K MONITOR SYSTEM >>
                        ;               by
                        ;
                        ;       TECHNICAL DESIGN LABS, INC.
                        ;       RESEARCH PARK
                        ;       PRINCETON, NEW JERSEY 08540
                        ;
                        ;       COPYRIGHT JULY 1976 TDL INC.
                        ;
                        ;       ASSEMBLED by Roger Amidon
                        ;
                        .PREL   ;THIS MONITOR SUPPLIED IN RELOCATING FORMAT
                        ;
0000'                   BASE    = .
0800'                   USER    = BASE+800H
                        ;
                        .TITLE " <Zapple Monitor, Version 1.11, Dec. 18 1976>"
                        .SBTTL  / Copyright 1976 by TECHNICAL DESIGN LABS, INC./
                        ;
0038                    RST7    = 38H   ;RST 7 (LOCATION FOR TRAP)
0002                    IOBYT   = 2     ;R/W PORT FOR TEMP. STORAGE
0003                    SENSE   = 3     ;PORT FOR INITIAL I/O CONFIGURATION (IN)
00FF                    SWITCH  = 0FFH  ;FRONT PANEL SENSE SWITCHES
0003                    RCP     = SENSE ;READER CONTROL PORT (OUT)
0000                    NN      = 0     ;"I" REGISTER INITIAL VALUE
                        ;
                        ;       <I/O DEVICES>
                        ;
                        ;-TELEPRINTER
                        ;
0001                    TTI     = 1     ;DATA IN PORT
0001                    TTO     = 1     ;DATA OUT PORT
0000                    TTS     = 0     ;STATUS PORT (IN)
0001                    TTYDA   = 1     ;DATA AVAILABLE MASK BIT
0080                    TTYBE   = 80H   ;XMTR BUFFER EMPTY MASK
                        ;
                        ;-C.R.T. SYSTEM
                        ;
0005                    CRTI    = 5     ;DATA PORT (IN)
0004                    CRTS    = 4     ;STATUS PORT (IN)
0005                    CRTO    = 5     ;DATA PORT (OUT)
0001                    CRTDA   = 1     ;DATA AVAILABLE MASK
0080                    CRTBE   = 80H   ;XMTR BUFFER EMPTY MASK
                        ;
                        ;-CASSETTE SYSTEM
                        ;
0007                    RCSD    = 7     ;DATA IN PORT
0006                    RCSS    = 6     ;STATUS PORT (IN)
0001                    RCSDA   = 1     ;DATA AVAILABLE MASK
0007                    PCASO   = 7     ;DATA PORT (OUT)
0006                    PCASS   = 6     ;CONTROL PORT (OUT)
0080                    PCSBE   = 80H   ;XMTR BUFFER EMPTY MASK
                        ;
                        ;       <CONSTANTS>
                        ;
```

```
0000            FALSE   = 0               ;ISN'T SO
FFFF            TRUE    = # FALSE       ;IT IS SO
000D            CR      = 0DH           ;ASCII CARRIAGE RETURN
000A            LF      = 0AH           ;ASCII LINE FEED
0007            BELL    = 7             ;DING
00FF            RUB     = 0FFH          ;RUB OUT
0000            FIL     = 00            ;FILL CHARACTERS AFTER CRLF
0007            MAX     = 7             ;NUMBER OF QUES IN EOF
                ;
                ;       <I/O CONFIGURATION MASKS>
                ;
00FC            CMSK    = 11111100B     ;CONSOLE DEVICE
00F3            RMSK    = 11110011B     ;STORAGE DEVICE (IN)
00CF            PMSK    = 11001111B     ;STORAGE DEVICE (OUT)
003F            LMSK    = 00111111B     ;LIST DEVICE
                ;
                ;
                ;-CONSOLE CONFIGURATION
0000            CTTY    = 0     ;TELEPRINTER
0001            CCRT    = 1     ;C.R.T.
0002            BATCH   = 2     ;READER FOR INPUT, LIST FOR OUTPUT
0003            CUSE    = 3     ;USER DEFINED
                ;
                ;-STORAGE INPUT CONFIGURATION
0000            RTTY    = 0     ;TELEPRINTER READER
0004            RPTR    = 4     ;HIGH-SPEED RDR (EXTERNAL ROUTINE)
0008            RCAS    = 8     ;CASSETTE
000C            RUSER   = 0CH   ;USER DEFINED
                ;
                ;-STORAGE OUTPUT CONFIGURATION
0000            PTTY    = 0     ;TELEPRINTER PUNCH
0010            PPTP    = 10H   ;HIGH-SPEED PUNCH (EXTERNAL ROUTINE)
0020            PCAS    = 20H   ;CASSETTE
0030            PUSER   = 30H   ;USER DEFINED
                ;
                ;-LIST DEVICE CONFIGURATION
0000            LTTY    = 0     ;TELEPRINTER PRINTER
0040            LCRT    = 40H   ;C.R.T. SCREEN
0080            LINE    = 80H   ;LINE PRINTER (EXTERNAL ROUTINE)
00C0            LUSER   = 0C0H  ;USER DEFINED
                ;
                ;
                ;       VECTORS FOR USER DEFINED ROUTINES
                ;
0800'           .LOC    USER
0800'           CILOC:  .BLKB 3 ;CONSOLE INPUT
0803'           COLOC:  .BLKB 3 ;CONSOLE OUTPUT
0806'           RPTPL:  .BLKB 3 ;HIGH-SPEED READER
0809'           RULOC:  .BLKB 3 ;USER DEFINED STORAGE (INPUT)
080C'           PTPL:   .BLKB 3 ;HIGH-SPEED PUNCH
080F'           PULOC:  .BLKB 3 ;USER DEFINED STORAGE (OUTPUT)
0812'           LNLOC:  .BLKB 3 ;LINE PRINTER
0815'           LULOC:  .BLKB 3 ;USER DEFINED PRINTER
0818'           CSLOC:  .BLKB 3 ;CONSOLE INPUT STATUS ROUTINE
081B'           J =.
```

```
                        ;
                        ;       PROGRAM CODE BEGINS HERE
                        ;
0000'                   .LOC    BASE
0000' C3 0031'                  JMP     BEGIN   ;GO AROUND VECTORS
                        ;
                        ;       <VECTORS FOR CALLING PROGRAMS>
                        ;
                        ; THESE VECTORS MAY BE USED BY USER WRITTEN
                        ; PROGRAMS TO SIMPLIFY THE HANDLING OF I/O
                        ; FROM SYSTEM TO SYSTEM.  WHATEVER THE CURRENT
                        ; ASSIGNED DEVICE, THESE VECTORS WILL PERFORM
                        ; THE REQUIRED I/O OPERATION, AND RETURN TO
                        ; THE CALLING PROGRAM. (RET)
                        ;
                        ; THE REGISTER CONVENTION USED.FOLLOWS-
                        ;
                        ; ANY INPUT OR OUTPUT DEVICE-
                        ;       CHARACTER TO BE OUTPUT IN 'C' REGISTER.
                        ;       CHARACTER WILL BE IN 'A' REGISTER UPON
                        ;       RETURNING FROM AN INPUT OR OUTPUT.
                        ; 'CSTS'-
                        ;       RETURNS TRUE (0FFH IN 'A' REG.) IF THERE IS
                        ;       SOMETHING WAITING, AND ZERO (00) IF NOT.
                        ; 'IOCHK'-
                        ;       RETURNS WITH THE CURRENT I/O CONFIGURATION
                        ;       BYTE IN 'A' REGISTER.
                        ; 'IOSET'-
                        ;       ALLOWS A PROGRAM TO DYNAMICALLY ALTER THE
                        ;       CURRENT I/O CONFIGURATION, AND REQUIRES
                        ;       THE NEW BYTE IN 'C' REGISTER.
                        ; 'MEMCK'-
                        ;       RETURNS WITH THE HIGHEST ALLOWED USER
                        ;       MEMORY LOCATION. 'B'=HIGH BYTE, 'A'=LOW.
                        ; 'TRAP'-
                        ;       THIS IS THE 'BREAKPOINT' ENTRY POINT,
                        ;       BUT MAY BE 'CALLED'. IT WILL SAVE
                        ;       THE MACHINE STATE. RETURN CAN BE MADE WITH
                        ;       A SIMPLE 'G[CR]' ON THE CONSOLE.
                        ;
0003' C3 0613'                  JMP     CI      ;CONSOLE INPUT
0006' C3 0633'                  JMP     RI      ;READER INPUT
0009' C3 0478'                  JMP     CO      ;CONSOLE OUTPUT
000C' C3 04B4'                  JMP     PO      ;PUNCH OUTPUT
000F' C3 049A'                  JMP     LO      ;LIST OUTPUT
0012' C3 050C'                  JMP     CSTS    ;CONSOLE STATUS
0015' C3 010B'                  JMP     IOCHK   ;I/O CHECK
0018' C3 0106'                  JMP     IOSET   ;I/O SET
001B' C3 059F'                  JMP     MEMCK   ;MEMORY LIMIT CHECK
001E' C3 06BE'          TRAP:   JMP     RESTART ;BREAKPOINT
                        ;
                        ;       ANNOUNCEMENT OF MONITOR NAME & VERSION
                        ;
0021' 0D0A000000        MSG:    .BYTE   CR,LF,FIL,FIL,FIL
0026' 5A6170706C                .ASCII  'Zapple V'
```

```
002E' 312E31              .ASCII  '1.1'
0010                MSGL  = .-MSG
                    ;
                    ;     LET US BEGIN
                    ;
                    ; *NOTE- THE CODE UP TO THE 'IN SENSE' MAY
                    ; BE REPLACED BY ENOUGH CODE TO INITIALIZE
                    ; AN ACIA OR SIO DEVICE. ADDITIONAL DEVICES
                    ; MAY BE INITIALIZED USING THE 'Q' COMMAND.
                    ; (OR STANDARD ROUTINES FOR INITIALIZATION
                    ; MAY BE LOADED & EXECUTED IN THE USER AREA).
                    ;
0031' 3E00          BEGIN:  MVI     A,NN    ;FOR 'I' REG. IF NEEDED.
0033' ED47                  STAI
0035' 00                    NOP             ;SPARE BYTE
0036' AF                    XRA     A       ;CLEAR READER CONTROL PORT
0037' D303                  OUT     RCP
                    ;
0039' DB03                  IN      SENSE   ;INITIALIZE I/O CONFIGURATION
003B' D302                  OUT     IOBYT
003D' 31 0041'              LXI     SP,AHEAD-4      ;SET UP A FAKE STACK
0040' C3 05AD'              JMP     MEMSIZ+1        ;GET MEMORY SIZE
0043' 0045'                 .WORD   AHEAD
0045' F9            AHEAD:  SPHL            ;SET TRUE STACK
0046' EB                    XCHG
0047' 01 0023               LXI     B,ENDX-EXIT
004A' 21 07A8'              LXI     H,EXIT
004D' EDB0                  LDIR            ;MOVE TO RAM
004F' EB                    XCHG
0050' 01 FFA1               LXI     B,-5FH  ;SET UP A USER'S STACK VALUE
0053' 09                    DAD     B
0054' E5                    PUSH    H       ;PRE-LOAD STACK VALUE
0055' 21 0000               LXI     H,0     ;INITIALIZE OTHER REGISTERS
0058' 060A                  MVI     B,10    ; (16 OF THEM)
005A' E5            STKIT:  PUSH    H       ; TO ZERO
005B' 10FD                  DJNZ    STKIT
005D' 0610          HELLO:  MVI     B,MSGL  ;SAY HELLO TO THE FOLKS
005F' CD 043D'              CALL    TOM     ;OUTPUT SIGN-ON MSG
0062' 11 0062'      START:  LXI     D,START ;MAIN 'WORK' LOOP
0065' D5                    PUSH    D       ;SET UP A RETURN TO HERE
0066' CD 0504'              CALL    CRLF
0069' 0E3E                  MVI     C,'>'
006B' CD 0478'              CALL    CO
006E' CD 0736'      STAR0:  CALL    TI      ;GET A CONSOLE CHARACTER
0071' E67F                  ANI     7FH     ;IGNORE NULLS
0073' 28F9                  JRZ     STAR0   ;GET ANOTHER
0075' D641                  SUI     'A'     ;QUALIFY THE CHARACTER
0077' F8                    RM              ;<A
0078' FE1A                  CPI     'Z'-'A'+1
007A' D0                    RNC             ;>Z
007B' 87                    ADD     A       ;A*2
007C' 0600                  MVI     B,0
007E' 4F                    MOV     C,A     ;POINT TO PLACE ON TABLE
007F' 21 008A'              LXI     H,TBL   ;POINT TO COMMAND TABLE
0082' 09                    DAD     B       ;ADD IN DISPLACEMENT
```

```
0083' 5E                MOV     E,M
0084' 23                INX     H
0085' 56                MOV     D,M
0086' EB                XCHG            ;D&E=ROUTINE ADDRESS
0087' 0E02              MVI     C,2     ;SET C UP
0089' E9                PCHL            ;GO EXECUTE COMMAND
                ;
                ;               <COMMAND BRANCH TABLE>
                ;
008A'           TBL:
008A' 00BE'     .WORD   ASSIGN  ;A - ASSIGN I/O
008C' 010F'     .WORD   BYE     ;B - SYSTEM SHUT-DOWN
008E' 013C'     .WORD   COMP    ;C - COMPARE MEMORY VS. READER INPUT
0090' 015D'     .WORD   DISP    ;D - DISPLAY MEMORY ON CONS. IN HEX
0092' 0174'     .WORD   EOF     ;E - END OF FILE TAG FOR HEX DUMPS
0094' 0190'     .WORD   FILL    ;F - FILL MEMORY WITH A CONSTANT
0096' 019D'     .WORD   GOTO    ;G - GOTO [ADDR]<,>BREAKPOINTS (2)
0098' 0571'     .WORD   HEXN    ;H - HEX MATH. <SUM>,<DIFFERENCE>
009A' 0452'     .WORD   ERROR   ;I * USER DEFINED, INSERT VECTOR
081B'                   J=J     ;VECTOR ADDR
009C' 01EB'     .WORD   TEST    ;J - NON-DESTRUCTIVE MEMORY TEST
009E' 0452'     .WORD   ERROR   ;K * USER DEFINED, INSERT VECTOR
081E'                   J=J+3   ;VECTOR ADDR
00A0' 0681'     .WORD   LOAD    ;L - LOAD A BINARY FORMAT FILE
00A2' 0209'     .WORD   MOVE    ;M - MOVE BLOCKS OF MEMORY
00A4' 04E9'     .WORD   NULL    ;N - PUNCH NULLS ON PUNCH DEVICE
00A6' 0452'     .WORD   ERROR   ;O * USER DEFINED, INSERT VECTOR
0821'                   J=J+3   ;VECTOR ADDR
00A8' 011D'     .WORD   PUTA    ;P - 'PUT' ASCII INTO MEMORY.
00AA' 0757'     .WORD   QUERY   ;Q - QI(N)=DISP. N; QO(N,V)=OUT N,V
00AC' 0214'     .WORD   READ    ;R - READ A HEX FILE (W/CHECKSUMS)
00AE' 02CD'     .WORD   SUBS    ;S - SUBSTITUTE &/OR EXAMINE MEMORY
00B0' 02F6'     .WORD   TYPE    ;T - TYPE MEMORY IN ASCII
00B2' 04D1'     .WORD   UNLD    ;U - MEMORY TO PUNCH (BINARY FORMAT)
00B4' 0782'     .WORD   VERIFY  ;V - COMPARE MEMORY AGAINST MEMORY
00B6' 035E'     .WORD   WRITE   ;W - MEMORY TO PUNCH (HEX FORMAT)
00B8' 039E'     .WORD   XAM     ;X - EXAMINE & MODIFY CPU REGISTERS
00BA' 0316'     .WORD   WHERE   ;Y - FIND SEQUENCE OF BYTES IN MEM.
00BC' 0469'     .WORD   SIZE    ;Z - ADDRESS OF LAST R/W LOCATION
                ;
                ;
                ;
                ;       THIS ROUTINE CONTROLS THE CONFIGURATION
                ; OF THE VARIOUS I/O DRIVERS & DEVICES. THIS IS
                ; ACCOMPLISHED VIA A HARDWARE READ/WRITE PORT.
                ;       THIS PORT IS INITIALIZED UPON SIGN-ON
                ; BY THE VALUE READ ON PORT 'SENSE'.  IT MAY BE
                ; DYNAMICALLY MODIFIED THROUGH CONSOLE COMMANDS.
                ;
                ; THE VALUE ON THE 'IOBYT' PORT REPRESENTS THE
                ; CURRENT CONFIGURATION.  IT IS STRUCTURED THUSLY:
                ;
                ; 000000XX - WHERE XX REPRESENTS THE CURRENT CONSOLE.
                ; 0000XX00 - WHERE XX REPRESENTS THE CURRENT READER.
                ; 00XX0000 - WHERE XX REPRESENTS THE CURRENT PUNCH.
```

```
                        ; XX000000 - WHERE XX REPRESENTS THE CURRENT LISTER.
                        ;
                        ; WHEN USING A MEMORY LOCATION FOR IOBYT, THE
                        ; POLARITY IS REVERSED.  FOR AN I/O PORT,
                        ; WHEN XX = 11, THE DEVICE IS ALWAYS THE
                        ; TELEPRINTER.  WHEN XX = 00, THE DEVICE IS ALWAYS
                        ; USER DEFINED.  SEE OPERATORS MANUAL FOR FURTHER
                        ; DETAILS.
                        ;
00BE' CD 0736'    ASSIGN: CALL    TI        ;GET DEVICE NAME
00C1' 21 0794'            LXI     H,LTBL    ;POINT TO DEVICE TABLE
00C4' 01 0400             LXI     B,400H    ;4 DEVICES TO LOOK FOR
00C7' 11 0005             LXI     D,5       ;4 DEV. + IDENT.
00CA' BE          ..A0:   CMP     M         ;LOOK FOR MATCH
00CB' 2806                JRZ     ..A1
00CD' 19                  DAD     D         ;GO THRU TABLE
00CE' 0C                  INR     C         ;KEEP TRACK OF DEVICE
00CF' 10F9                DJNZ    ..A0
00D1' 1815                JMPR    ..ERR     ;WRONG IDENTIFIER
00D3' 59          ..A1:   MOV     E,C       ;SAVE DEVICE NUMBER
00D4' CD 0736'    ..A2:   CALL    TI        ;SCAN PAST '='
00D7' FE3D                CPI     '='
00D9' 20F9                JRNZ    ..A2
00DB' CD 0736'            CALL    TI        ;GET NEW ASSIGNMENT
00DE' 01 0400             LXI     B,400H    ;4 POSSIBLE ASSIGNMENTS
00E1' 23          ..A3:   INX     H         ;POINT TO ASSIGNMENT NAME
00E2' BE                  CMP     M         ;LOOK FOR PROPER MATCH
00E3' 2806                JRZ     ..A4      ;MATCH FOUND
00E5' 0C                  INR     C         ;KEEP TRACK OF ASSIGNMENT NMBR
00E6' 10F9                DJNZ    ..A3
00E8' C3 0452'    ..ERR:  JMP     ERROR     ;NO MATCH, ERROR
00EB' 3E03        ..A4:   MVI     A,3       ;SET UP A MASK
00ED' 1C                  INR     E
00EE' 1D          ..A5:   DCR     E         ;DEVICE IN E
00EF' 2808                JRZ     ..A6      ;GOT IT
00F1' CB21                SLAR    C         ;ELSE MOVE MASKS
00F3' CB21                SLAR    C
00F5' 17                  RAL
00F6' 17                  RAL               ;A=DEVICE MASK
00F7' 18F5                JMPR    ..A5
00F9' 2F          ..A6:   CMA               ;INVERT FOR AND'ING
00FA' 57                  MOV     D,A       ;SAVE IN D
00FB' CD 0604'    ..A7:   CALL    PCHK      ;WAIT FOR [CR]
00FE' 30FB                JRNC    ..A7
0100' CD 010B'            CALL    IOCHK     ;GET PRESENT CONFIGURATION
0103' A2                  ANA     D         ;MODIFY ONLY SELECTED DEVICE
0104' B1                  ORA     C         ;'OR' IN NEW BIT PATTERN
0105' 4F                  MOV     C,A       ;NEW CONFIGURATION
                        ;
                        ; THIS ALLOWS USER PROGRAMS TO MODIFY
                        ; THE I/O CONFIGURATION DYNAMICALLY
                        ; DURING EXECUTION.
                        ;
0106' 79          IOSET:  MOV     A,C       ;NEW I/O BYTE PASSED IN C REG
0107' 2F                  CMA               ;WE SAVE THE INVERTED BYTE
```

```
0108' D302              OUT     IOBYT   ;IN AN I/O PORT LATCH
010A' C9                RET
                ;
                ; THIS RETURNS THE CURRENT I/O
                ; CONFIGURATION IN THE  A REG.
                ;
010B' DB02      IOCHK:  IN      IOBYT   ;GET SAVED VALUE
010D' 2F                CMA             ;AND INVERT IT AGAIN
010E' C9                RET
                ;
                ;
                ; THIS ROUTINE IS USED AS A SIMPLE MEANS TO PREVENT
                ; UNAUTHORIZED SYSTEM OPERATION. THE SYSTEM LOCKS UP,
                ; MONITORING FOR A 'CONT.-SHIFT-N', AT WHICH TIME IT
                ; WILL SIGN-ON AGAIN. NO REGISTER ASSIGNMENTS OR I/O
                ; CONFIGURATIONS WILL BE ALTERED.
                ;
010F' CD 0504'  BYE:    CALL    CRLF
0112' CD 0730'  ..BY:   CALL    KI
0115' FE1E              CPI     1EH     ;CONTROL-SHIFT-N
0117' 20F9              JRNZ    ..BY
0119' D1                POP     D       ;REMOVE THE RETURN
011A' C3 005D'          JMP     HELLO   ;AND SIGN-ON AGAIN
                ;
                ; THIS ALLOWS ENTERING OF ASCII TEXT INTO MEMORY
                ; FROM THE CONSOLE DEVICE. THE PARITY BIT IS CLEARED,
                ; AND ALL WILL BE STORED EXCEPT THE BACK-ARROW [_]
                ; WHICH DELETES THE PREVIOUS CHARACTER, AND
                ; CONTROL-D, WHICH RETURNS CONTROL TO THE MONITOR.
                ; THIS COMMAND, COMBINED WITH THE 'Y' COMMAND,
                ; PROVIDES A RUDIMENTARY TEXT PROCESSING ABILITY.
                ;
011D' CD 0533'  PUTA:   CALL    EXPR1   ;GET THE STARTING ADDR.
0120' CD 0504'          CALL    CRLF
0123' E1                POP     H
0124' CD 0730'  ..A1:   CALL    KI      ;GET A CHARACTER
0127' FE04              CPI     4       ;CONTROL-D? (EOT)
0129' CA 0470'          JZ      LFADR   ;YES, STOP & PRINT ADDR.
012C' FE5F              CPI     '_'     ;ERASE MISTAKE?
012E' 2808              JRZ     ..A3    ; YES.
0130' 77                MOV     M,A     ;ELSE STORE IT IN MEMORY
0131' 4F                MOV     C,A
0132' 23                INX     H
0133' CD 0478'  ..A2:   CALL    CO      ;ECHO ON CONSOLE
0136' 18EC              JMPR    ..A1
0138' 2B        ..A3:   DCX     H       ;BACK UP POINTER
0139' 4E                MOV     C,M
013A' 18F7              JMPR    ..A2    ;ECHO & CONTINUE
                ;
                ; THIS ROUTINE COMPARES THE READER INPUT
                ; DEVICE WITH THE MEMORY BLOCK SPECIFIED.
                ; IT TESTS ALL EIGHT BITS, AND ANY DISCREPENCIES
                ; WILL BE OUTPUT TO THE CONSOLE. THIS IS USEFUL
                ; WHEN USED WITH THE BINARY DUMP FORMAT TO BOTH
                ; VERIFY PROPER READING & STORAGE, OR TO DETECT
```

```
                        ; PROGRAM CHANGES SINCE IT WAS LAST LOADED.
                        ;
013C' CD 04FF'  COMP:   CALL    EXLF    ;GET START ' STOP ADDR.
013F' CD 0462'  ..C:    CALL    RIFF    ;GET A FULL READER BYTE
0142' BE                CMP     M       ;8 BIT COMAPARE
0143' C4 014B'          CNZ     CERR    ;CALL IF INVALID COMPARE
0146' CD 0561'          CALL    HILOX   ;SEE IF RANGE SATISFIED
0149' 18F4              JMPR    ..C
                        ;
                        ; THIS SUBOUTINE IS USED TO DISPLAY THE
                        ; CURRENT LOCATION OF THE 'M' REGISTER POINTERS (HL),
                        ; AND THE VALUE AT THE LOCATION, AND THE CONTENTS
                        ; OF THE ACCUMULATOR. USED BY TWO ROUTINES.
                        ;
014B' 47        CERR:   MOV     B,A     ;SAVE ACC.
014C' CD 0473'          CALL    HLSP    ;DISPLAY H&L
014F' 7E                MOV     A,M
0150' CD 0582'          CALL    LBYTE   ;PRINT 'M'
0153' CD 0476'          CALL    BLK     ;SPACE OVER
0156' 78                MOV     A,B
0157' CD 0582'          CALL    LBYTE   ;PRINT ACC.
015A' C3 0504'          JMP     CRLF    ;CRLF & RETURN
                        ;
                        ; THIS DISPLAYS THE CONTENTS OF MEMORY IN BASE HEX
                        ; WITH THE STARTING LOCATION ON EACH LINE.(BETWEEN
                        ; THE TWO PARAMETERS GIVEN). 16 BYTES PER LINE MAX.
                        ;
015D' CD 04FF'  DISP:   CALL    EXLF    ;GET DISPLAY RANGE
0160' CD 0470'  ..D0:   CALL    LFADR   ;CRLF & PRINT ADDR.
0163' CD 0476'  ..D1:   CALL    BLK     ;SPACE OVER
0166' 7E                MOV     A,M
0167' CD 0582'          CALL    LBYTE
016A' CD 0561'          CALL    HILOX   ;RANGE CHECK
016D' 7D                MOV     A,L
016E' E60F              ANI     0FH     ;SEE IF TIME TO CRLF
0170' 20F1              JRNZ    ..D1
0172' 18EC              JMPR    ..D0
                        ;
                        ; THIS OUTPUTS THE END OF FILE (EOF) PATTERN
                        ; FOR THE CHECKSUM LOADER. IT IS USED AFTER
                        ; PUNCHING A BLOCK OF MEMORY WITH THE 'W'
                        ; COMMAND.  AN ADDRESS PARAMETER MAY BE GIVEN,
                        ; AND UPON READING, THIS ADDRESS WILL BE
                        ; AUTOMATICALLY PLACED IN THE 'P' COUNTER. THE
                        ; PROGRAM CAN THEN BE RUN WITH A SIMPLE 'G[CR]'
                        ; COMMAND.
                        ;
0174' CD 0533'  EOF:    CALL    EXPR1   ;GET OPTIONAL ADDR.
0177' CD 04AD'          CALL    PEOL    ;CRLF TO PUNCH
017A' 0E3A              MVI     C,':'   ;FILE MARKER CUE
017C' CD 04B4'          CALL    PO
017F' AF                XRA     A       ;ZERO LENGTH
0180' CD 05E8'          CALL    PBYTE
0183' E1                POP     H
0184' CD 05E3'          CALL    PADR    ;PUNCH OPTIONAL ADDR.
```

```
0187' 21 0000            LXI     H,0     ;FILE TYPE=0
018A' CD 05E3'           CALL    PADR    ;PUNCH IT
018D' C3 04E9'           JMP     NULL    ;TRAILER & RETURN
                  ;
                  ; THIS COMMAND WILL FILL A BLOCK OF MEMORY
                  ; WITH A VALUE. IE; F0,1FFF,0  FILLS FROM
                  ; <1> TO <2> WITH THE BYTE <3>. HANDY FOR
                  ; INITIALIZING A BLOCK TO A SPECIFIC VALUE, OR
                  ; MEMORY TO A CONSTANT VALUE BEFORE LOADING
                  ; A PROGRAM. (ZERO IS ESPECIALLY USEFUL.)
                  ;
0190' CD 0528'    FILL:   CALL    EXPR3   ;GET 3 PARAMETERS
0193' 71          ..F:    MOV     M,C     ;STORE THE BYTE
0194' CD 0567'            CALL    HILO
0197' 30FA                JRNC    ..F
0199' D1                  POP     D       ;RESTORE STACK
019A' C3 0062'            JMP     START   ; IN CASE OF ACCIDENTS
                  ;
                  ; THIS COMMAND ALLOWS EXECUTION OF ANOTHER
                  ; PROGRAM WHILE RETAINING SOME MONITOR
                  ; CONTROL BY SETTING BREAKPOINTS.
                  ;
                  ; TO SIMPLY EXECUTE, TYPE 'G<ADDR>[CR]'. TO SET
                  ; A BREAKPOINT TRAP, ADD THE ADDRESS(ES) TO THE
                  ; COMMAND. IE: G<ADDR>,<BKPT>[CR]. TWO BREAKPOINTS
                  ; ARE ALLOWED, ENOUGH TO SATISFY MOST REQUIREMENTS.
                  ; ONCE A BREAKPOINT HAS BEEN REACHED, THE
                  ; REGISTERS MAY BE EXAMINED OR MODIFIED. THE
                  ; PROGRAM CAN THEN BE CONTINUED BY TYPING ONLY
                  ; A 'G[CR]'. OR ANOTHER BREAKPOINT COULD BE
                  ; IMPLEMENTED AT THAT TIME BY TYPING 'G,<BKPT>[CR]'.
                  ;
                  ; *NOTE: THIS IS SOFTWARE CONTROLLED, AND THE
                  ; BREAKPOINT MUST OCCUR ON AN INTRUCTION
                  ; BYTE.
                  ;
019D' CD 0604'    GOTO:   CALL    PCHK    ;GET A POSSIBLE ADDRESS
01A0' 3840                JRC     ..G3    ;CR ENTERED
01A2' 2810                JRZ     ..G0    ;DELIMETER ENTERED
01A4' CD 055A'            CALL    EXF     ;GET ONE EXPRESSION
01A7' D1                  POP     D
01A8' 21 0034             LXI H,  PLOC    ;PLACE ADDRESS IN 'P' LOCATION
01AB' 39                  DAD     SP
01AC' 72                  MOV     M,D     ;HIGH BYTE
01AD' 2B                  DCX     H
01AE' 73                  MOV     M,E     ;LOW BYTE
01AF' 78                  MOV     A,B
01B0' FE0D                CPI     CR      ;SEE IF LAST CHARACTER WAS CR
01B2' 282E                JRZ     ..G3    ;YES, LEAVE
01B4' 1602        ..G0:   MVI     D,2     ;TWO BREAKPOINTS MAX
01B6' 21 0035             LXI H,  TLOC    ;POINT TO TRAP STORAGE
01B9' 39                  DAD     SP
01BA' E5          ..G1:   PUSH    H       ;SAVE STORAGE POINTER
01BB' CD 0533'            CALL    EXPR1   ;GET A TRAP ADDRESS
01BE' 58                  MOV     E,B     ;SAVE DELIMETER
```

```
01BF' C1                POP     B       ;TRAP ADDR.
01C0' E1                POP     H       ;STORAGE
01C1' 78                MOV     A,B     ;LOOK AT TRAP ADDR
01C2' B1                ORA     C
01C3' 280A              JRZ     ..G2    ;DON'T SET A TRAP AT 0
01C5' 71                MOV     M,C     ;SAVE BKPT ADDR
01C6' 23                INX     H
01C7' 70                MOV     M,B
01C8' 23                INX     H
01C9' 0A                LDAX    B       ;PICK UP INST. BYTE
01CA' 77                MOV     M,A     ;SAVE THAT TOO
01CB' 23                INX     H
01CC' 3EFF              MVI     A,0FFH  ;RST 7
01CE' 02                STAX    B       ;SOFTWARE INTERUPT
01CF' 7B        ..G2:   MOV     A,E     ;LOOK AT DELIMITER
01D0' FE0D              CPI     CR
01D2' 2803              JRZ     ..G2A
01D4' 15                DCR     D       ;COUNT BKPTS
01D5' 20E3              JRNZ    ..G1    ;GET ONE MORE
01D7' 3EC3      ..G2A:  MVI     A,JMP   ;SET UP JMP INSTRUCTION
01D9' 32 0038           STA     RST7    ; AT RESTART TRAP LOC.
01DC' 21 001E'          LXI     H,TRAP  ; TO MONITOR VECTOR
01DF' 22 0039           SHLD    RST7+1
01E2' CD 0504'  ..G3:   CALL    CRLF
01E5' D1                POP     D       ;CLEAR SYSTEM RETURN
01E6' 21 0016           LXI H,  22      ;FIND 'EXIT' ROUTINE
01E9' 39                DAD     SP      ;UP IN STACK
01EA' E9                PCHL            ;GO SOMEPLACE
                ;
                ; THIS IS A 'QUICKIE' MEMORY TEST TO SPOT
                ; HARD MEMORY FAILURES, OR ACCIDENTLY
                ; PROTECTED MEMORY LOCATIONS. IT IS NOT
                ; MEANT TO BE THE DEFINITIVE MEMORY DIAGNOSTIC.
                ; IT IS, HOWEVER, NON-DESTRUCTIVE. ERRORS ARE
                ; PRINTED ON THE CONSOLE AS FOLLOWS-
                ; <ADDR> 00000100 WHERE <1> IS THE BAD BIT.
                ; BIT LOCATION OF THE FAILURE IS EASILY
                ; DETERMINED. NON-R/W MEMORY WILL RETURN
                ; WITH-  11111111
                ;
01EB' CD 04FF'  TEST:   CALL    EXLF    ;GET TWO PARAMS
01EE' 7E        ..T1:   MOV     A,M     ;READ A BYTE
01EF' 47                MOV     B,A     ;SAVE IN B REG.
01F0' 2F                CMA
01F1' 77                MOV     M,A     ;READ/COMPLIMENT/WRITE
01F2' AE                XRA     M       ; & COMPARE
01F3' 280E              JRZ     ..T2    ;SKIP IF ZERO (OK)
01F5' D5                PUSH    D       ;SAVE END POINTER
01F6' 50                MOV     D,B     ;SAVE BYTE
01F7' 5F                MOV     E,A     ;SET-UP TO DISPLAY
01F8' CD 0473'          CALL    HLSP    ;PRINT BAD ADDR
01FB' CD 0769'          CALL    BITS    ;PRINT BAD BIT LOC.
01FE' CD 0504'          CALL    CRLF
0201' 42                MOV     B,D     ;RESTORE BYTE
0202' D1                POP     D       ;RESTORE DE
```

```
0203' 70          ..T2:   MOV     M,B     ;REPLACE BYTE
0204' CD 0561'            CALL    HILOX   ;RANGE TEST
0207' 18E5                JMPR    ..T1
                  ;
                  ; THIS COMMAND MOVES MASS AMOUNTS OF MEMORY
                  ; FROM <1> THRU <2> TO THE ADDRESS STARTING
                  ; AT <3>.  THIS ROUTINE SHOULD BE USED WITH
                  ; SOME CAUTION, AS IT COULD SMASH MEMORY IF
                  ; CARELESSLY IMPLEMENTED.
                  ;
                  ;       M<1>,<2>,<3>
                  ;
0209' CD 0528'    MOVE:   CALL    EXPR3   ;GET 3 PARAMETERS
020C' 7E          ..M:    MOV     A,M     ;PICK UP
020D' 02                  STAX    B       ;PUT DOWN
020E' 03                  INX     B       ;MOVE UP
020F' CD 0561'            CALL    HILOX   ;CHECK IF DONE
0212' 18F8                JMPR    ..M
                  ;
                  ; THIS COMMAND READS THE CHECK-SUMMED HEX FILES
                  ; FOR BOTH THE NORMAL INTEL FORMAT AND THE TDL
                  ; RELOCATING FORMAT. ON BOTH FILES, A 'BIAS' MAY
                  ; BE ADDED, WHICH WILL CAUSE THE OBJECT CODE TO
                  ; BE PLACED IN A LOCATION OTHER THAN ITS
                  ; INTENDED EXECUTION LOCATION. THE BIAS IS ADDED TO
                  ; WHAT WOULD HAVE BEEN THE NORMAL LOADING
                  ; LOCATION, AND WILL WRAP AROUND TO ENABLE
                  ; LOADING ANY PROGRAM ANYWHERE IN MEMORY.
                  ;
                  ; WHEN LOADING A RELOCATABLE FILE, AN ADDITIONAL
                  ; PARAMETER MAY BE ADDED, WHICH REPRESENTS THE
                  ; ACTUAL EXECUTION ADDRESS DESIRED. THIS ALSO MAY
                  ; BE ANY LOCATION IN MEMORY.
                  ;
                  ; EXAMPLES:
                  ;
                  ; R[CR] =0 BIAS, 0 EXECUTION ADDR.
                  ; R<ADDR1>[CR] =<1>BIAS, 0 EXECUTION ADDR.
                  ; R,<ADDR1>[CR] =0 BIAS, <1> EXECUTION ADDR.
                  ; R<ADDR1>,<ADDR2>[CR] =<1>BIAS, <2> EXECUTION ADDR.
                  ;
0214' CD 0533'    READ:   CALL    EXPR1   ;GET BIAS, IF ANY
0217' 78                  MOV     A,B     ;LOOK AT DELIMITER
0218' D60D                SUI     CR      ;ALL DONE?
021A' 47                  MOV     B,A     ;SET UP RELOCATION OF 0
021B' 4F                  MOV     C,A     ; IF CR ENTERED
021C' D1                  POP     D       ;BIAS AMOUNT
021D' 2804                JRZ     ..R0    ;CR ENTERED
021F' CD 0533'            CALL    EXPR1   ;GET RELOCATION
0222' C1                  POP     B       ;ACTUAL RELOCATION VALUE
0223' EB          ..R0:   XCHG
0224' D9                  EXX             ;HL'=BIAS, BC'=RELOCATION
0225' CD 0504'            CALL    CRLF
0228' CD 067B'    LOD0:   CALL    RIX     ;GET A CHARACTER
022B' D63A                SUI     ':'     ;ABSOLUTE FILE CUE?
```

```
022D' 47                MOV     B,A     ;SAVE CUE CLUE
022E' E6FE              ANI     0FEH    ;KILL BIT 0
0230' 20F6              JRNZ    LOD0    ; NO, KEEP LOOKING
0232' 57                MOV     D,A     ;ZERO CHECKSUM
0233' CD 02AE'          CALL    SBYTE   ;GET FILE LENGTH
0236' 5F                MOV     E,A     ;SAVE IN E REG.
0237' CD 02AE'          CALL    SBYTE   ;GET LOAD MSB
023A' F5                PUSH    PSW     ;SAVE IT
023B' CD 02AE'          CALL    SBYTE   ;GET LOAD LSB
023E' D9                EXX             ;CHANGE GEARS
023F' D1                POP     D       ;RECOVER MSB
0240' 5F                MOV     E,A     ;FULL LOAD ADDR
0241' C5                PUSH    B       ;BC'=RELOCATION
0242' D5                PUSH    D       ;DE'=LOAD ADDR
0243' E5                PUSH    H       ; HL'=BIAS
0244' 19                DAD     D       ; BIAS+LOAD
0245' E3                XTHL            ;RESTORE HL'
0246' DDE1              POP     X       ; X=BIAS+LOAD
0248' D9                EXX             ;DOWNSHIFT
0249' E1                POP     H       ;HL=LOAD ADDR
024A' CD 02AE'          CALL    SBYTE   ;GET FILE TYPE
024D' 3D                DCR     A       ;1=REL. FILE, 0=ABS.
024E' 78                MOV     A,B     ;SAVE CUE BIT
024F' C1                POP     B       ;BC=RELOCATION
0250' 2003              JRNZ    ..A     ;ABSOLUTE FILE
0252' 09                DAD     B       ;ELSE RELOCATE
0253' DD09              DADX    B       ;BOTH X & HL
0255' 1C        ..A:    INR     E       ;TEST LENGTH
0256' 1D                DCR     E       ;0=DONE
0257' 2819              JRZ     DONE
0259' 3D                DCR     A       ;TEST CUE
025A' 2822              JRZ     LODR    ;RELATIVE
025C' CD 02AE'  ..L1:   CALL    SBYTE   ;NEXT
025F' CD 02C1'          CALL    STORE   ;STORE IT
0262' 20F8              JRNZ    ..L1    ;MORE COMING
0264' CD 02AE'  LOD4:   CALL    SBYTE   ;GET CHECKSUM
0267' 28BF              JRZ     LOD0    ;GOOD CHECKSUM
0269' DDE5      ERR3:   PUSH    X
026B' E1                POP     H       ;TRANSFER
026C' CD 057D'          CALL    LADR    ;PRINT CURRENT LOAD ADDR
026F' C3 0452'  ERR2:   JMP     ERROR   ;ABORT
0272' 7C        DONE:   MOV     A,H     ;DON'T MODIFY IF ZERO
0273' B5                ORA     L
0274' C8                RZ
0275' EB                XCHG            ;ELSE STORE PC
0276' 21 0034           LXI H,  PLOC
0279' 39                DAD     SP
027A' 72                MOV     M,D     ;IN STACK AREA
027B' 2B                DCX     H
027C' 73                MOV     M,E
027D' C9                RET
027E' 2E01      LODR:   MVI     L,1     ;SET-UP BIT COUNTER
0280' CD 029E'  ..L1:   CALL    LODCB   ;GET THE BIT
0283' 3807              JRC     ..L3    ;DOUBLE BIT
0285' CD 02C1'  ..L5:   CALL    STORE   ;WRITE IT
```

```
0288' 20F6              JRNZ    ..L1
028A' 18D8              JMPR    LOD4    ;TEST CHECKSUM
028C' 4F        ..L3:   MOV     C,A     ;SAVE LOW BYTE
028D' CD 029E'          CALL    LODCB   ;NEXT CONTROL BIT
0290' 47                MOV     B,A     ;SAVE HIGH BYTE
0291' D9                EXX
0292' C5                PUSH    B       ;GET RELOCATION
0293' D9                EXX
0294' E3                XTHL            ;INTO HL
0295' 09                DAD     B       ;RELOCATE
0296' 7D                MOV     A,L     ;LOW BYTE
0297' CD 02C1'          CALL    STORE   ;STORE IT
029A' 7C                MOV     A,H     ;HIGH BYTE
029B' E1                POP     H       ;RESTORE HL
029C' 18E7              JMPR    ..L5    ;DO THIS AGAIN
029E' 2D        LODCB:  DCR     L       ;COUNT BITS
029F' 2007              JRNZ    ..LC1   ;MORE LEFT
02A1' CD 02AE'          CALL    SBYTE   ;GET NEXT
02A4' 1D                DCR     E       ;COUNT BYTES
02A5' 67                MOV     H,A     ;SAVE THE BITS
02A6' 2E08              MVI     L,8     ;8 BITS/BYTE
02A8' CD 02AE'  ..LC1:  CALL    SBYTE   ;GET A DATA BYTE
02AB' CB24              SLAR    H       ;TEST NEXT BIT
02AD' C9                RET
02AE' C5        SBYTE:  PUSH    B       ;PRESERVE BC
02AF' CD 05D0'          CALL    RIBBLE  ;GET A CONVERTED ASCII CHAR.
02B2' 07                RLC
02B3' 07                RLC
02B4' 07                RLC
02B5' 07                RLC             ;MOVE IT TO HIGH NIBBLE
02B6' 4F                MOV     C,A     ;SAVE IT
02B7' CD 05D0'          CALL    RIBBLE  ;GET OTHER HALF
02BA' B1                ORA     C       ;MAKE WHOLE
02BB' 4F                MOV     C,A     ;SAVE AGAIN IN C
02BC' 82                ADD     D       ;UPDATE CHECKSUM
02BD' 57                MOV     D,A     ;NEW CHECKSUM
02BE' 79                MOV     A,C     ;CONVERTED BYTE
02BF' C1                POP     B
02C0' C9                RET
02C1' DD7700    STORE:  MOV     0(X),A  ;WRITE TO MEMORY
02C4' DDBE00            CMP     0(X)    ;VALID WRITE?
02C7' 20A0              JRNZ    ERR3    ; NO.
02C9' DD23              INX     X       ;ADVANCE POINTER
02CB' 1D                DCR     E       ;COUNT DOWN
02CC' C9                RET
                ;
                ; THIS ROUTINE ALLOWS BOTH INSPECTION OF &
                ; MODIFICATION OF MEMORY ON A BYTE BY BYTE
                ; BASIS. IT TAKES ONE ADDRESS PARAMETER,
                ; FOLLOWED BY A SPACE.  THE DATA AT THAT
                ; LOCATION WILL BE DISPLAYED. IF IT IS
                ; DESIRED TO CHANGE IT, THE VALUE IS THEN
                ; ENTERED.  A FOLLOWING SPACE WILL DISPLAY
                ; THE NEXT BYTE.  A CARRIAGE RETURN [CR]
                ; WILL TERMINATE THE COMMAND.  THE SYSTEM
```

```
                        ; ADDS A CRLF AT LOCATIONS ENDING WITH EITHER
                        ; XXX0 OR XXX8. TO AID IN DETERMINING THE
                        ; PRESENT ADDRESS, IT IS PRINTED AFTER
                        ; EACH CRLF.  A BACKARROW [_] WILL BACK
                        ; UP THE POINTER AND DISPLAY THE
                        ; PREVIOUS LOCATION.
                        ;
02CD' CD 0533'  SUBS:   CALL    EXPR1   ;GET STARTING ADDR.
02D0' E1                POP     H
02D1' 7E        ..S0:   MOV     A,M
02D2' CD 0582'          CALL    LBYTE   ;DISPLAY THE BYTE
02D5' CD 05FF'          CALL    COPCK   ;MODIFY?
02D8' D8                RC              ; NO, ALL DONE
02D9' 280F              JRZ     ..S1    ;DON'T MODIFY
02DB' FE5F              CPI     '_'     ;BACKUP?
02DD' 2814              JRZ     ..S2
02DF' E5                PUSH    H       ;SAVE POINTER
02E0' CD 055A'          CALL    EXF     ;GET NEW VALUE
02E3' D1                POP     D       ;VALUE IN E
02E4' E1                POP     H
02E5' 73                MOV     M,E     ;MODIFY
02E6' 78                MOV     A,B     ;TEST DELIMITER
02E7' FE0D              CPI     CR
02E9' C8                RZ              ;DONE
02EA' 23        ..S1:   INX     H
02EB' 7D        ..S3:   MOV     A,L     ;SEE IF TIME TO CRLF
02EC' E607              ANI     7
02EE' CC 0470'          CZ      LFADR   ;TIME TO CRLF
02F1' 18DE              JMPR    ..S0
02F3' 2B        ..S2:   DCX     H       ;DECREMENT POINTER
02F4' 18F5              JMPR    ..S3    ;AND PRINT DATA THERE.
                        ;
                        ; THIS ROUTINE TRANSLATES THE DATA IN
                        ; MEMORY TO AN ASCII FORMAT.  ALL NON-
                        ; PRINTING CHARACTERS ARE CONVERTED TO
                        ; PERIODS. [.]
                        ; THERE ARE 64 CHARACTERS PER LINE.
                        ;
02F6' CD 04FF'  TYPE:   CALL    EXLF    ;DISPLAY RANGE
02F9' CD 0470'  ..T0:   CALL    LFADR   ;DISPLAY ADDRESS
02FC' 0640              MVI     B,64    ;CHARACTERS PER LINE
02FE' 7E        ..T1:   MOV     A,M
02FF' E67F              ANI     7FH     ;KILL PARITY BIT
0301' FE20              CPI     ' '     ;RANGE TEST
0303' 3002              JRNC    ..T3    ;=>SPACE
0305' 3E2E      ..T2:   MVI     A,'.'   ;REPLACE NON-PRINTING
0307' FE7C      ..T3:   CPI     07CH    ;ABOVE LOWER CASE z
0309' 30FA              JRNC    ..T2
030B' 4F                MOV     C,A     ;SEND IT
030C' CD 0478'          CALL    CO
030F' CD 0561'          CALL    HILOX   ;MORE TO GO?
0312' 10EA              DJNZ    ..T1    ;SEE IF TIME TO CRLF
0314' 18E3              JMPR    ..T0    ; YES.
                        ;
                        ; THIS IS A HEXADECIMAL SEARCH ROUTINE. IT
```

```
                        ; TAKES NO ADDRESS PARAMETERS. AS MANY
                        ; BYTES MAY BE ENTERED, SEPARATED BY A COMMA,
                        ; AS DESIRED.  THE MAXIMUM IS 255, BUT 3-4 IS
                        ; TYPICAL, AND MORE THAN 12 WOULD BE UNUSUAL.
                        ; THE ENTIRE MEMORY IS SEARCHED, STARTING
                        ; FROM ZERO, AND ALL STARTING ADDRESSES OF EACH
                        ; OCCURENCE OF THE SEARCH STRING ARE PRINTED
                        ; ON THE CONSOLE DEVICE.
                        ;
0316' 1600      WHERE:  MVI     D,0     ;COUNT SEARCH BYTES
0318' CD 0533'  ..W0:   CALL    EXPR1   ;GET ONE BYTE
031B' E1                POP     H       ;PICK IT UP
031C' 65                MOV     H,L     ;STICK IN HIGH BYTE
031D' E5                PUSH    H       ;PUT IT IN STACK
031E' 33                INX     SP      ;ADJUST STACK
031F' 14                INR     D       ;COUNT UP
0320' 78                MOV     A,B     ;TEST DELIMITER
0321' D60D              SUI     CR
0323' 20F3              JRNZ    ..W0    ;MORE TO GO
0325' 47                MOV     B,A     ;CHEAP ZEROES
0326' 4F                MOV     C,A
0327' 67                MOV     H,A
0328' 6A                MOV     L,D     ;GET BYTE COUNT IN L
0329' 2D                DCR     L       ;-1
032A' 39                DAD     SP      ;BYTES STORED IN STACK
032B' E5                PUSH    H
032C' C5                PUSH    B
032D' C5        FINDC:  PUSH    B       ;SAVE THAT POINTER
032E' CD 0504'          CALL    CRLF
0331' C1                POP     B       ;RESTORE
0332' E1        FIND:   POP     H       ;HL=SEARCH ADDR
0333' DDE1              POP     X       ;X=SEARCH BYTE POINTER
0335' 5A                MOV     E,D     ;RESET COUNT
0336' DD7E00            MOV     A,0(X)  ;GET THE FIRST SEARCH BYTE
0339' EDB1              CCIR            ;COMPARE, INCR., & REPEAT
033B' E2 0359'          JPO     DONE2   ;ODD PARITY=DONE
033E' DDE5              PUSH    X       ;SAVE POINTERS
0340' E5                PUSH    H
0341' 1D        FOUND:  DCR     E
0342' 280B              JRZ     TELL    ;FOUND ALL
0344' DD7EFF            MOV     A,-1(X) ;LOOK AT NEXT MATCH
0347' BE                CMP     M       ;TEST NEXT
0348' 20E8              JRNZ    FIND    ;NO MATCH
034A' 23                INX     H       ;BUMP POINTERS
034B' DD2B              DCX     X
034D' 18F2              JMPR    FOUND   ;TEST NEXT MATCH
034F' E1        TELL:   POP     H
0350' E5                PUSH    H
0351' 2B                DCX     H
0352' C5                PUSH    B       ;SAVE SEARCH COUNT LIMIT
0353' CD 057D'          CALL    LADR    ;TELL CONSOLE
0356' C1                POP     B       ;RESTORE
0357' 18D4              JMPR    FINDC
0359' 33        DONE2:  INX     SP
035A' 1D                DCR     E       ;RESET STACK
```

```
035B' 20FC              JRNZ    DONE2
035D' C9                RET
                ;
                ; THIS ROUTINE DUMPS MEMORY IN THE STANDARD
                ; INTEL HEX-FILE FORMAT.  A START & END
                ; PARAMETER IS REQUIRED. AT THE CONCLUSION
                ; OF THE DUMP, AN "END OF FILE" SHOULD BE
                ; GENERATED WITH THE "E" COMMAND.
                ;
035E' CD 04FF'  WRITE:  CALL    EXLF    ;GET TWO PARAMETERS
0361' CD 04EC'          CALL    WAIT    ;PAUSE IF TTY CONFIGURATION
0364' CD 04AD'  ..W0:   CALL    PEOL    ;CRLF TO PUNCH
0367' 01 003A           LXI     B,':'   ;START-OF-FILE CUE
036A' CD 04B4'          CALL    PO      ;PUNCH IT
036D' D5                PUSH    D       ;SAVE
036E' E5                PUSH    H       ; POINTERS
036F' 04        ..W1:   INR     B       ;CALCULATE FILE LENGTH
0370' CD 0567'          CALL    HILO
0373' 3824              JRC     ..W4    ;SHORT FILE
0375' 3E18              MVI     A,24    ;24 BYTES PER FILE
0377' 90                SUB     B       ;ENOUGH YET?
0378' 20F5              JRNZ    ..W1    ; NO.
037A' E1                POP     H       ;GET START ADDR BACK.
037B' CD 0381'          CALL    ..W2    ;SEND THE BLOCK
037E' D1                POP     D       ;RESTORE END OF FILE POINTER
037F' 18E3              JMPR    ..W0    ;KEEP GOING
0381' 57        ..W2:   MOV     D,A     ;INITIALIZE CHECKSUM
0382' 78                MOV     A,B     ;FILE LENGTH
0383' CD 05E8'          CALL    PBYTE   ;PUNCH IT
0386' CD 05E3'          CALL    PADR    ;PUNCH ADDRESS
0389' AF                XRA     A       ;FILE TYPE=0
038A' CD 05E8'          CALL    PBYTE   ;PUNCH IT
038D' 7E        ..W3:   MOV     A,M     ;GET A DATA BYTE
038E' CD 05E8'          CALL    PBYTE   ;PUNCH IT
0391' 23                INX     H       ;POINT TO NEXT BYTE
0392' 10F9              DJNZ    ..W3    ;DECREMENT FILE COUNT
0394' AF                XRA     A
0395' 92                SUB     D       ;CALCULATE CHECKSUM
0396' C3 05E8'          JMP     PBYTE   ;PUNCH IT, RETURN
0399' E1        ..W4:   POP     H       ;CLEAR STACK
039A' D1                POP     D       ; OF POINTERS
039B' AF                XRA     A       ;SET-UP A
039C' 18E3              JMPR    ..W2    ;FINISH UP & RETURN
                ;
                ;
                ;       THIS ROUTINE ALLOWS DISPLAYING THE
                ; USER'S CPU REGISTERS. THEY ALSO MAY BE
                ; USING THE REGISTER NAME AFTER TYPING THE "X".
                ; I.E.  XA 00-
                ; THE REGISTER MAY BE SKIPPED OVER, OR MODIFIED,
                ; SIMILARLY TO THE "S" COMMAND.
                ;
                ; TO DISPLAY THE "NORMAL" SYSTEM STATUS,
                ; SIMPLY TYPE "X[CR]".  TO DISPLAY THE
                ; ADDITIONAL Z-80 REGISTERS, FOLLOW
```

```
                        ; THE "X" WITH AN APOSTROPHE.  I.E. "X'[CR]",
                        ; OR TO EXAMINE A SINGLE "PRIME" REGISTER,
                        ; TYPE THE REGISTER IDENTIFIER AFTER THE
                        ; APOSTROPHE.  I.E.  X'X 0000-
                        ;
                        ; THESE REGISTER VALUES ARE PLACED INTO THE CPU
                        ; UPON EXECUTING ANY "GO" COMMAND. [G]
                        ;
039E' CD 0736'    XAM:  CALL    TI
03A1' 21 07CB'          LXI     H,ACTBL
03A4' FE0D              CPI     CR        ;FULL REG. DISPLAY
03A6' 285A              JRZ     ..X6
03A8' FE27              CPI     "'"       ;SEE IF PRIMES WANTED
03AA' 200A              JRNZ    ..X0
03AC' 21 07E7'          LXI     H,PRMTB
03AF' CD 0736'          CALL    TI
03B2' FE0D              CPI     CR        ;FULL REG. DISPLAY
03B4' 284C              JRZ     ..X6
03B6' BE          ..X0: CMP     M         ;TEST FOR REGISTER NAME
03B7' 2809              JRZ     ..X1
03B9' CB7E              BIT     7,M       ;SEE IF END OF TABLE
03BB' C2 0452'          JNZ     ERROR
03BE' 23                INX     H
03BF' 23                INX     H
03C0' 18F4              JMPR    ..X0
03C2' CD 0476'    ..X1: CALL    BLK
03C5' 23          ..X2: INX     H
03C6' 7E                MOV     A,M
03C7' 47                MOV     B,A       ;SAVE FOR FLAGS
03C8' E63F              ANI     3FH       ;CLEAR FLAGS FOR BIAS
03CA' EB                XCHG
03CB' 6F                MOV     L,A       ;DISPLACEMENT FROM STACK
03CC' 2600              MVI     H,0
03CE' 39                DAD     SP
03CF' EB                XCHG
03D0' 23                INX     H
03D1' 1A                LDAX    D         ;PICK UP REG. VALUE
03D2' CD 0582'          CALL    LBYTE     ;PRINT IT
03D5' CB78              BIT     7,B
03D7' 2805              JRZ     ..X3
03D9' 1B                DCX     D
03DA' 1A                LDAX    D
03DB' CD 0582'          CALL    LBYTE
03DE' CD 05FF'    ..X3: CALL    COPCK     ;MODIFY?
03E1' D8                RC                ;CR ENTERED, ALL DONE
03E2' 2819              JRZ     ..X5      ;SKIP TO NEXT REG.
03E4' E5                PUSH    H
03E5' C5                PUSH    B
03E6' CD 055A'          CALL    EXF       ;GET NEW VALUE
03E9' E1                POP     H
03EA' F1                POP     PSW
03EB' C5                PUSH    B
03EC' F5                PUSH    PSW
03ED' 7D                MOV     A,L
03EE' 12                STAX    D
```

```
03EF' C1                POP     B
03F0' CB78              BIT     7,B     ;SEE IF 8 BIT OR 16 BIT REG.
03F2' 2803              JRZ     ..X4    ;8 BIT
03F4' 13                INX     D
03F5' 7C                MOV     A,H     ;HIGH BYTE OF 16 BIT REG.
03F6' 12                STAX    D
03F7' C1        ..X4:   POP     B
03F8' E1                POP     H
03F9' 78                MOV     A,B     ;TEST DELIMITER
03FA' FE0D              CPI     CR
03FC' C8                RZ              ;CR ENTERED, ALL DONE
03FD' CB7E      ..X5:   BIT     7,M     ;SEE IF END OF TABLE
03FF' C0                RNZ             ;RETURN IF SO
0400' 18C3              JMPR    ..X2
0402' CD 0504'  ..X6:   CALL    CRLF
0405' CD 0476'  ..X7:   CALL    BLK
0408' 7E                MOV     A,M
0409' 23                INX     H
040A' B7                ORA     A
040B' F8                RM
040C' 4F                MOV     C,A
040D' CD 0478'          CALL    CO
0410' 0E3D              MVI     C,'='
0412' CD 0478'          CALL    CO
0415' 7E                MOV     A,M
0416' 47                MOV     B,A     ;SAVE FLAGS
0417' E63F              ANI     3FH     ;CLEAN UP FOR OFFSET
0419' 23                INX     H
041A' EB                XCHG
041B' 6F                MOV     L,A
041C' 2600              MVI     H,0
041E' 39                DAD     SP
041F' EB                XCHG
0420' CB70              BIT     6,B     ;TEST FOR SPECIAL "M"
0422' 200F              JRNZ    ..X9    ;PRINT OUT ACTUAL "M"
0424' 1A                LDAX    D
0425' CD 0582'          CALL    LBYTE   ;PRINT REG. VALUE
0428' CB78              BIT     7,B     ;SINGLE OR DOUBLE?
042A' 28D9              JRZ     ..X7    ;SINGLE.
042C' 1B                DCX     D
042D' 1A                LDAX    D
042E' CD 0582'  ..X8:   CALL    LBYTE
0431' 18D2              JMPR    ..X7
0433' E5        ..X9:   PUSH    H       ;SAVE HL
0434' 1A                LDAX    D       ;GET REG. POINTER
0435' 67                MOV     H,A     ;HIGH BYTE
0436' 1B                DCX     D
0437' 1A                LDAX    D
0438' 6F                MOV     L,A     ;LOW BYTE
0439' 7E                MOV     A,M     ;GET VALUE
043A' E1                POP     H       ;RESTORE HL
043B' 18F1              JMPR    ..X8    ;PRINT VALUE & CONTINUE
                ;
                ; THIS IS A MESSAGE OUTPUT ROUTINE.
                ; IT IS USED BY THE SIGN-ON AND CRLF.
```

```
                    ; POINTER IS IN HL (WHEN ENTERED AT
                    ; TOM1) AND LENGTH IN B REG.
                    ;
043D' 21 0021'      TOM:    LXI     H,MSG
0440' 4E            TOM1:   MOV     C,M     ;GET A CHARACTER
0441' 23                    INX     H       ;MOVE POINTER
0442' CD 0478'              CALL    CO      ;OUTPUT IT
0445' 10F9                  DJNZ    TOM1    ;KEEP GOING TILL B=0
0447' CD 050C'              CALL    CSTS    ;SEE IF AN ABORT REQUEST
044A' B7                    ORA     A       ; WAITING.
044B' C8                    RZ              ;NO.
                    ;
                    ; SEE IF CONTROL-C IS WAITING
                    ; ABORT IF SO.
                    ;
044C' CD 0730'      CCHK:   CALL    KI
044F' FE03                  CPI     3       ;CONTROL-C?
0451' C0                    RNZ
                    ;
                    ; SYSTEM ERROR ROUTINE. THIS
                    ; WILL RESTORE THE SYSTEM AFTER
                    ; A SYSTEM ERROR HAS BEEN TAKEN.
                    ; THE I/O CONFIGURATION IS NOT
                    ; AFFECTED.
                    ;
0452' CD 05AC'      ERROR:  CALL    MEMSIZ
0455' 11 FFEA               LXI     D,-22   ;STACK POINTER OFFSET
0458' 19                    DAD     D
0459' F9                    SPHL            ;RESET STACK
045A' 0E2A                  MVI     C,'*'   ;ANNOUNCE ERROR
045C' CD 0478'              CALL    CO
045F' C3 0062'              JMP     START   ;BACK TO WORK
                    ;
                    ; THIS GETS A READER CHARACTER,
                    ; AND COMAPRES IT WITH 'D' REG.
                    ; IT ABORTS ON AN 'OUT-OF-DATA'
                    ; CONDITION.
                    ;
0462' CD 0633'      RIFF:   CALL    RI      ;GET READER CHARACTER
0465' 38EB                  JRC     ERROR   ;ABORT ON CARRY
0467' BA                    CMP     D       ;TEST D
0468' C9                    RET
                    ;
                    ; THIS ROUTINE WILL RETURN THE
                    ; CURRENT VALUE OF THE HIGHEST
                    ; READ/WRITE MEMORY LOCATION THAT
                    ; IS AVAILABLE ON THE SYSTEM.
                    ; IT WILL "SEARCH" FOR MEMORY
                    ; STARTING AT THE BOTTOM OF MEMORY
                    ; AND GO UPWARDS UNTIL NON-R/W MEMORY
                    ; IS FOUND.
                    ;
0469' CD 05AC'      SIZE:   CALL    MEMSIZE ;GET THE VALUE
046C' 01 0023               LXI     B,(ENDX-EXIT)
046F' 09                    DAD     B       ;ADJUST IT
```

```
                        ;
                        ;
                        ; CRLF BEFORE HLSP ROUTINE
                        ;
0470' CD 0504'          LFADR:  CALL    CRLF
                        ;
                        ; PRINT THE CURRENT VALUE OF H&L,
                        ; AND A SPACE.
                        ;
0473' CD 057D'          HLSP:   CALL    LADR
                        ;
                        ; PRINT A SPACE ON THE CONSOLE
                        ;
0476' 0E20              BLK:    MVI     C,' '
                        ;
                        ; THIS IS THE MAIN CONSOLE
                        ; OUTPUT ROUTINE.
                        ;
0478' CD 010B'          CO:     CALL    IOCHK
047B' E603                      ANI     # CMSK
047D' 200A                      JRNZ    CO0
                        ;
                        ; TELEPRINTER CONFIGURATION
                        ; I/O DRIVER.
                        ;
047F' DB00              TTYOUT: IN      TTS
0481' E680                      ANI     TTYBE
0483' 20FA                      JRNZ    TTYOUT
0485' 79                        MOV     A,C
0486' D301                      OUT     TTO
0488' C9                        RET
0489' 3D                CO0:    DCR     A       ;CCRT?
048A' 200A                      JRNZ    CO1     ; NO.
                        ;
                        ; C.R.T. CONFIGURATION DRIVER.
                        ;
048C' DB04              CRTOUT: IN      CRTS
048E' E680                      ANI     CRTBE
0490' 20FA                      JRNZ    CRTOUT
0492' 79                        MOV     A,C
0493' D305                      OUT     CRTO
0495' C9                        RET
                        ;
0496' 3D                CO1:    DCR     A       ;BATCH?
0497' C2 0803'                  JNZ     COLOC   ; NO, MUST BE USER
                        ;
                        ; LIST OUTPUT DRIVER ROUTINE
                        ; -A USER VECTORED ROUTINE, USED
                        ; BY THE ASSEMBLER, ETC. ALSO,
                        ; WHEN THE ASSIGNED MODE IS "BATCH",
                        ; THIS IS THE ROUTINE USED FOR THE
                        ; MONITOR OUTPUT THAT WOULD NORMALLY
                        ; GO TO THE "CONSOLE".
                        ;
049A' CD 010B'          LO:     CALL    IOCHK
```

```
049D' E6C0              ANI     # LMSK
049F' 28DE              JRZ     TTYOUT
04A1' FE40              CPI     LCRT
04A3' 28E7              JRZ     CRTOUT
04A5' FE80              CPI     LINE
04A7' CA 0812'          JZ      LNLOC   ;EXTERNAL VECTOR
04AA' C3 0815'          JMP     LULOC   ;USER DEFINED VECTOR
                ;
                ; SEND CRLF TO PUNCH DEVICE
                ;
04AD' 0E0D      PEOL:   MVI     C,CR
04AF' CD 04B4'          CALL    PO
04B2' 0E0A              MVI     C,LF
                ;
                ; PUNCH OUTPUT DRIVER ROUTINE
                ;
04B4' CD 010B'  PO:     CALL    IOCHK
04B7' E630              ANI     # PMSK
04B9' 28C4              JRZ     TTYOUT  ;PUNCH=TELEPRINTER
04BB' FE20              CPI     PCAS    ;CASSETTE?
04BD' 200A              JRNZ    PO1     ; NO.
                ;
04BF' DB06      PO0:    IN      PCASS
04C1' E680              ANI     PCSBE
04C3' 20FA              JRNZ    PO0
04C5' 79                MOV     A,C
04C6' D307              OUT     PCASO
04C8' C9                RET
                ;
04C9' FE10      PO1:    CPI     PPTP
04CB' CA 080C'          JZ      PTPL    ;EXTERNAL VECTOR
04CE' C3 080F'          JMP     PULOC   ;USER VECTOR
                ;
                ;
                ; THIS IS A BINARY DUMP ROUTINE THAT MAY BE
                ; USED WITH BOTH PAPER-TAPE AND/OR CASSETTE
                ; SYSTEMS.  IT PUNCHES A START-OF-FILE MARK
                ; AND THEN PUNCHES IN FULL 8-BITS DIRECTLY
                ; FROM MEMORY.  IT IS FOLLOWED BY AN END-OF-
                ; FILE MARKER. THESE DUMPS MAY BE LOADED
                ; USING THE "L" COMMAND. THEY ARE USEFUL
                ; FOR FAST LOADING, AND MAY BE VERIFIED
                ; USING THE "C" (COMPARE) COMMAND.
                ;
                ;   U<A1>,<A2>[CR]
                ; PUNCHES FROM <A1> THRU <A2>
                ;
04D1' CD 04FF'  UNLD:   CALL    EXLF    ;GET TWO PARAMETERS
04D4' CD 04EC'          CALL    WAIT    ;PAUSE FOR PUNCH-ON (TTY)
04D7' CD 0596'          CALL    LEAD    ;PUNCH LEADER
04DA' CD 0591'          CALL    MARK    ;PUNCH FILE MARKER
04DD' 4E        ..U:    MOV     C,M     ;GET MEMORY BYTE
04DE' CD 04B4'          CALL    PO      ;PUNCH IT
04E1' CD 0567'          CALL    HILO    ;SEE IF DONE
04E4' 30F7              JRNC    ..U
```

```
04E6' CD 0591'          CALL    MARK    ;PUNCH END FILE MARKER
                ;
                ; THIS PUNCHES NULLS (LEADER/TRAILER).
                ; IT RETURNS "QUIET" IN CASE THE PUNCH
                ; AND CONSOLE ARE THE SAME.
                ;
04E9' CD 0596'  NULL:   CALL    LEAD    ;PUNCH NULLS
                ;
                ; THIS ROUTINE WILL PAUSE FOR
                ; A KEYBOARD CHARACTER. IT IS
                ; USED AS A DELAY TO GIVE THE
                ; OPERATOR TIME TO TURN ON THE
                ; TELEPRINTER PUNCH BEFORE SENDING
                ; A HEX FILE OR BINARY FILE TO
                ; THE PUNCH.  IT WILL SIMPLY
                ; RETURN IF THE PUNCH & CONSOLE
                ; ARE NOT BOTH ASSIGNED TO THE
                ; DEFAULT. (TELEPRINTER).
                ;
04EC' CD 010B'  WAIT:   CALL    IOCHK
04EF' E633              ANI     # CMSK ! # PMSK
04F1' C0                RNZ
04F2' C3 006E'          JMP     STAR0   ;RETURN "QUIET"
                ;
                ; CONVERT HEX TO ASCII
                ;
04F5' E60F      CONV:   ANI     0FH     ;LOW NIBBLE ONLY
04F7' C690              ADI     90H
04F9' 27                DAA
04FA' CE40              ACI     40H
04FC' 27                DAA
04FD' 4F                MOV     C,A
04FE' C9                RET
                ;
                ; GET TWO PARAMETERS, PLACE
                ; THEM IN DE & HL, AND THEN
                ; CRLF.
                ;
04FF' CD 0535'  EXLF:   CALL    EXPR
0502' D1                POP     D
0503' E1                POP     H
                ;
                ; CONSOLE CARRIAGE RETURN &
                ; LINE FEED ROUTINE.
                ;
                ; THE NUMBER OF FILL CHARACTERS
                ; MAY BE ADJUSTED TO 0-3 BY THE
                ; VALUE PLACED IN THE B REG. MINIMUM
                ; VALUE FOR "B" IS TWO (2). MAXIMUM
                ; IS FIVE (5).
                ;
0504' E5        CRLF:   PUSH    H       ;SAVE HL
0505' 0604              MVI     B,4     ;CRLF LENGTH (SET FOR 2 FILLS)
0507' CD 043D'          CALL    TOM     ;SEND CRLF
050A' E1                POP     H
```

```
050B' C9                RET
                ;
                ; TEST THE CURRENT CONSOLES
                ; KEYBOARD FOR A KEY-PRESS.
                ; RETURN TRUE (0FFH IN A REG)
                ; IF THERE IS A CHARACTER
                ; WAITING IN THE UART.
                ;
050C' CD 010B'  CSTS:   CALL    IOCHK
050F' E603              ANI     # CMSK
0511' 2004              JRNZ    CS0
0513' DB00              IN      TTS
0515' 1805              JMPR    CS1
0517' 3D        CS0:    DCR     A       ;CCRT
0518' 2009              JRNZ    CS3
051A' DB04              IN      CRTS
051C' E601      CS1:    ANI     TTYDA
051E' 3E00              MVI     A,FALSE
0520' C0        CS2:    RNZ
0521' 2F                CMA
0522' C9                RET
0523' 3D        CS3:    DCR     A       ;BATCH
0524' C8                RZ
0525' C3 0818'          JMP     CSLOC   ;USED DEFINED VECTOR
                ;
                ; GET THREE PARAMETERS AND
                ; CRLF.
                ;
0528' 0C        EXPR3:  INR     C
0529' CD 0535'          CALL    EXPR
052C' CD 0504'          CALL    CRLF
052F' C1                POP     B
0530' D1                POP     D
0531' E1                POP     H
0532' C9                RET
                ;
                ; GET ONE PARAMETER.
                ; NO CRLF.
                ;
0533' 0E01      EXPR1:  MVI     C,1
                ;
                ; THIS IS THE MAIN "PARAMETER-GETTING" ROUTINE.
                ; THIS ROUTINE WILL ABORT ON A NON-HEX CHARACTER.
                ; IT TAKES THE MOST RECENTLY TYPED FOUR VALID
                ; HEX CHARACTERS, AND PLACES THEM UP ON THE STACK.
                ; (AS GNE 16 BIT VALUE, CONTAINED IN TWO
                ; 8-BIT BYTES.)  IF A CARRIAGE RETURN IS ENTERED,
                ; IT WILL PLACE THE VALUE OF "0000" IN THE STACK.
                ;
0535' 21 0000   EXPR:   LXI     H,0     ;INITIALIZE HL TO ZERO
0538' CD 0736'  EX0:    CALL    TI      ;GET SOMETHING FROM CONSOLE
053B' 47        EX1:    MOV     B,A     ;SAVE IT
053C' CD 05D3'          CALL    NIBBLE  ;CONVERT ASCII TO HEX.
053F' 3808              JRC     ..EX2   ;ILLEGAL CHARACTER DETECTED
0541' 29                DAD     H       ;MULTIPLY BY 16
```

```
0542' 29                DAD     H
0543' 29                DAD     H
0544' 29                DAD     H
0545' B5                ORA     L       ;OR IN THE SINGLE NIBBLE
0546' 6F                MOV     L,A
0547' 18EF              JMPR    EX0     ;GET SOME MORE
0549' E3        ..EX2:  XTHL            ;SAVE UP IN STACK
054A' E5                PUSH    H       ;REPLACE THE RETURN
054B' 78                MOV     A,B     ;TEST THE DELIMITER
054C' CD 0607'          CALL    QCHK
054F' 3002              JRNC    ..EX3   ;CR ENTERED
0551' 0D                DCR     C       ;SHOULD GO TO ZERO
0552' C8                RZ              ; RETURN IF IT DOES
0553' C2 0452'  ..EX3:  JNZ     ERROR   ;SOMETHING WRONG
0556' 0D                DCR     C       ;DO THIS AGAIN?
0557' 20DC              JRNZ    EXPR    ; YES.
0559' C9                RET             ;ELSE RETURN
055A' 0E01      EXF:    MVI     C,1
055C' 21 0000           LXI     H,0
055F' 18DA              JMPR    EX1
                ;
                ; RANGE TESTING ROUTINES.
                ; CARRY SET INDICATES RANGE EXCEEDED.
                ;
0561' CD 0567'  HILOX:  CALL    HILO
0564' D0                RNC             ;OK
0565' D1                POP     D       ;RETURN ONE LEVEL BACK
0566' C9                RET
                ;
0567' 23        HILO:   INX     H       ;INCREMENT HL
0568' 7C                MOV     A,H     ;TEST FOR CROSSING 64K BORDER
0569' B5                ORA     L
056A' 37                STC             ;CARRY SET=STOP
056B' C8                RZ              ;YES, BORDER CROSSED
056C' 7B                MOV     A,E     ;NOW, TEST HL VS. DE
056D' 95                SUB     L
056E' 7A                MOV     A,D
056F' 9C                SBB     H
0570' C9                RET             ;IF CARRY WAS SET, THEN STOP
                ;
                ;       HEXADECIMAL MATH ROUTINE
                ;
                ; THIS ROUTINE IS USEFUL FOR
                ; DETERMINING RELATIVE JUMP
                ; OFFSETS.  IT RETURNS THE SUM
                ; & DIFFERENCE OF TWO PARAMETERS.
                ;
                ;   H<X>,<Y>
                ;
                ;  X+Y   X-Y
                ;
0571' CD 04FF'  HEXN:   CALL    EXLF
0574' E5                PUSH    H       ;SAVE HL FOR LATER
0575' 19                DAD     D       ;GET SUM
0576' CD 0473'          CALL    HLSP    ;PRINT IT
```

```
0579' E1                POP     H       ;THIS IS LATER
057A' B7                ORA     A       ;CLEAR CARRY
057B' ED52              DSBC    D       ;GET DIFFERENCE & PRINT IT
                ;
                ; PRINT H&L ON CONSOLE
                ;
057D' 7C        LADR:   MOV     A,H
057E' CD 0582'          CALL    LBYTE
0581' 7D                MOV     A,L
0582' F5        LBYTE:  PUSH    PSW
0583' 0F                RRC
0584' 0F                RRC
0585' 0F                RRC
0586' 0F                RRC
0587' CD 058B'          CALL    ..2
058A' F1                POP     PSW
058B' CD 04F5'  ..2:    CALL    CONV
058E' C3 0478'          JMP     CO
                ; THIS ROUTINE SENDS EIGHT RUBOUTS
                ; TO THE PUNCH DEVICE.
                ;
0591' 01 08FF   MARK:   LXI     B,08FFH ;SET-UP B&C
0594' 1803              JMPR    LE0
                ;
                ; THIS ROUTINE SENDS BLANKS TO THE
                ; PUNCH DEVICE.
                ;
0596' 01 4800   LEAD:   LXI     B,4800H ;PRESET FOR SOME NULLS
0599' CD 04B4'  LE0:    CALL    PO
059C' 10FB              DJNZ    LE0
059E' C9                RET
                ;
                ; THIS ROUTINE RETURNS TO A USER
                ; PROGRAM THE CURRENT TOP OF
                ; MEMORY VALUE MINUS WORKSPACE
                ; AREA USED BY THE MONITOR.
                ;
059F' E5        MEMCK:  PUSH    H
05A0' CD 05AC'          CALL    MEMSIZ
05A3' 7D                MOV     A,L
05A4' D63C              SUI     3CH
05A6' 3001              JRNC    ..B
05A8' 25                DCR     H
05A9' 44        ..B:    MOV     B,H
05AA' E1                POP     H
05AB' C9                RET
                ;
                ; THIS IS A CALLED ROUTINE USED
                ; TO CALCULATE THE TOP OF MEMORY
                ; STARTING FROM THE BOTTOM OF
                ; MEMORY, AND SEARCHING UPWARD UNTIL
                ; FIRST R/W MEMORY IS FOUND, AND THEN
                ; CONTINUING UNTIL THE END OF THE R/W
                ; MEMORY. THIS ALLOWS R.O.M. AT ZERO,
                ; AND INSURES A CONTINUOUS MEMORY BLOCK
```

```
                    ; HAS BEEN FOUND.
                    ; IT IS USED BY THE ERROR ROUTINE TO
                    ; RESET THE STACK POINTER AS WELL.
                    ;
05AC' C5            MEMSIZ: PUSH    B
05AD' 01 0000'              LXI     B,BASE  ;POINT TO START OF MONITOR
05B0' 21 FFFF               LXI     H,-1    ;RAM SEARCH STARTING PT.
05B3' 24            ..M0:   INR     H       ;FIRST FIND R/W MEMORY
05B4' 7E                    MOV     A,M
05B5' 2F                    CMA
05B6' 77                    MOV     M,A
05B7' BE                    CMP     M
05B8' 2F                    CMA
05B9' 77                    MOV     M,A
05BA' 20F7                  JRNZ    ..M0
05BC' 24            ..M1:   INR     H       ;R/W FOUND, NOW FIND END
05BD' 7E                    MOV     A,M
05BE' 2F                    CMA
05BF' 77                    MOV     M,A
05C0' BE                    CMP     M
05C1' 2F                    CMA
05C2' 77                    MOV     M,A
05C3' 2004                  JRNZ    ..M2
05C5' 7C                    MOV     A,H     ;TEST FOR MONITOR BORDER
05C6' B8                    CMP     B
05C7' 20F3                  JRNZ    ..M1    ;NOT THERE YET
05C9' 25            ..M2:   DCR     H       ;BACK UP, SUBTRACT WORKSPACE
05CA' 01 FFDD               LXI     B,EXIT-ENDX
05CD' 09                    DAD     B
05CE' C1                    POP     B       ;RESTORE BC
05CF' C9                    RET             ;VALUE IN HL
                    ;
                    ;
05D0' CD 067B'      RIBBLE: CALL    RIX
05D3' D630          NIBBLE: SUI     '0'     ;QUALIFY & CONVERT
05D5' D8                    RC              ;<0
05D6' FE17                  CPI     'G'-'0' ;>F?
05D8' 3F                    CMC             ;PERVERT CARRY
05D9' D8                    RC
05DA' FE0A                  CPI     10      ;NMBR?
05DC' 3F                    CMC             ;PERVERT AGAIN
05DD' D0                    RNC             ;RETURN CLEAN
05DE' D607                  SUI 'A'-'9'-1   ;ADJUST
05E0' FE0A                  CPI     0AH     ;FILTER ":" THRU "@"
05E2' C9                    RET
                    ;
                    ; SEND H&L VALUE TO PUNCH DEVICE
                    ;
05E3' 7C            PADR:   MOV     A,H
05E4' CD 05E8'              CALL    PBYTE
05E7' 7D                    MOV     A,L
                    ;
                    ; PUNCH A SINGLE BYTE
                    ;
05E8' F5            PBYTE:  PUSH    PSW     ;NIBBLE AT A TIME
```

```
05E9' 0F                RRC
05EA' 0F                RRC
05EB' 0F                RRC
05EC' 0F                RRC
05ED' CD 04F5'          CALL    CONV
05F0' CD 04B4'          CALL    PO
05F3' F1                POP     PSW     ;NEXT NIBBLE
05F4' F5                PUSH    PSW     ;SAVE FOR CHECKSUM
05F5' CD 04F5'          CALL    CONV
05F8' CD 04B4'          CALL    PO
05FB' F1                POP     PSW     ;ORIGINAL BYTE HERE
05FC' 82                ADD     D       ;ADDED TO CHECKSUM
05FD' 57                MOV     D,A     ;UPDATE CHECKSUM
05FE' C9                RET
                ;
                ;
05FF' 0E2D      COPCK:  MVI     C,'-'
0601' CD 0478'          CALL    CO
                ;
0604' CD 0736'  PCHK:   CALL    TI
                ;
                ; TEST FOR DELIMITERS
                ;
0607' FE20      QCHK:   CPI     ' '     ;RETURN ZERO IF DELIMITER
0609' C8                RZ
060A' FE2C              CPI     ','
060C' C8                RZ
060D' FE0D              CPI     CR      ;RETURN W/CARRY SET IF CR
060F' 37                STC
0610' C8                RZ
0611' 3F                CMC             ;ELSE NON-ZERO, NO CARRY
0612' C9                RET
                ;
                ; MAIN CONSOLE INPUT ROUTINE
                ;
0613' CD 010B'  CI:     CALL    IOCHK
0616' E603              ANI     # CMSK
0618' 2009              JRNZ    CI1
                ;
                ; TELEPRINTER ROUTINE
                ;
061A' DB00      TTYIN:  IN      TTS
061C' E601              ANI     TTYDA
061E' 20FA              JRNZ    TTYIN
0620' DB01              IN      TTI
0622' C9                RET
                ;
0623' 3D        CI1:    DCR     A       ;CONSOLE=CRT?
0624' 2009              JRNZ    CI2
                ;
                ; C.R.T. INPUT ROUTINE
                ;
0626' DB04      CRTIN:  IN      CRTS
0628' E601              ANI     CRTDA
062A' 20FA              JRNZ    CRTIN
```

```
062C' DB05              IN      CRTI
062E' C9                RET
                ;
062F' 3D        CI2:    DCR     A       ;BATCH?
0630' C2 0800'          JNZ     CILOC   ;NO, MUST BE USER DEFINED
                ;
                ;
                ; READER INPUT ROUTINE, WITH
                ; TIME-OUT DELAY. INCLUDES
                ; PULSING OF HARDWARE PORT
                ; TO INDICATE REQUEST FOR
                ; READER DATA.
                ;
0633' E5        RI:     PUSH    H
0634' CD 010B'          CALL    IOCHK
0637' E60C              ANI     # RMSK
0639' 2F                CMA
063A' D303              OUT     RCP     ;PULSE READER CONTROL PORT
063C' 2F                CMA             ;CLEAR IT
063D' D303              OUT     RCP
063F' 201B              JRNZ    RI3     ;NOT TTY
0641' 67                MOV     H,A     ;CLEAR FOR TIME-OUT TEST
0642' DB00      RI0:    IN      TTS
0644' E601              ANI     TTYDA
0646' 280F              JRZ     RI2
0648' C5                PUSH    B
0649' 0600              MVI     B,0
064B' E3        DL0:    XTHL            ;WASTE TIME
064C' E3                XTHL            ;FOR DELAY
064D' 10FC              DJNZ    DL0
064F' C1                POP     B
0650' 25                DCR     H
0651' 20EF              JRNZ    RI0
0653' AF        RI1:    XRA     A
0654' 37                STC
0655' E1                POP     H
0656' C9                RET
0657' DB01      RI2:    IN      TTI
0659' B7        RID:    ORA     A
065A' E1                POP     H
065B' C9                RET
065C' FE08      RI3:    CPI     RCAS
065E' 2012              JRNZ    RI6
0660' DBFF              IN      SWITCH  ;READ INITIAL SENSE CONDX.
0662' 6F                MOV     L,A
0663' DBFF      RI4:    IN      SWITCH  ;SEE IF SW. ALTERED
0665' BD                CMP     L
0666' 20EB              JRNZ    RI1     ;ABORT IF SO
0668' DB06              IN      RCSS
066A' E601              ANI     RCSDA   ;DATA YET?
066C' 20F5              JRNZ    RI4     ;KEEP LOOKING
066E' DB07      RI5:    IN      RCSD
0670' 18E7              JMPR    RID
0672' E1        RI6:    POP     H
0673' FE04              CPI     RPTR
```

```
0675' CA 0806'          JZ      RPTPL   ;EXTERNAL ROUTINE
0678' C3 0809'          JMP     RULOC   ;USER VECTOR
                ;
                ; THIS ROUTINE GETS READER INPUT
                ; AND KILLS THE PARITY BIT.
                ;
067B' CD 0462'  RIX:    CALL    RIFF
067E' E67F              ANI     7FH
0680' C9                RET
                ;
                ; THIS ROUTINE READS A BINARY FILE
                ; IMAGE, IN THE FORM AS PUNCHED IN
                ; THE "U" (UNLOAD) COMMAND.  IT TAKES
                ; ONE PARAMETER, WHICH IS THE STARTING
                ; ADDRESS OF THE LOAD, AND WILL PRINT
                ; THE LAST ADDRESS(+1) LOADED ON THE
                ; CONSOLE DEVICE.
                ;
0681' CD 0533'  LOAD:   CALL    EXPR1   ;INITIAL LOAD ADDRESS
0684' E1                POP     H
0685' CD 0504'          CALL    CRLF
0688' 16FF              MVI     D,0FFH  ;START-OF-FILE TAG
068A' 0604      ..L0:   MVI     B,4     ;FIND AT LEAST FOUR 0FFH'S
068C' CD 0462'  ..L1:   CALL    RIFF
068F' 20F9              JRNZ    ..L0
0691' 10F9              DJNZ    ..L1
0693' CD 0462'  ..L2:   CALL    RIFF    ;4 FOUND, NOW WAIT FOR NON-0FFH
0696' 28FB              JRZ     ..L2
0698' 77                MOV     M,A     ;FIRST REAL DATA BYTE
0699' 3E07              MVI     A,BELL  ;TELL TTY
069B' D301              OUT     TTO
069D' 23        ..L3:   INX     H
069E' CD 0462'          CALL    RIFF
06A1' 2803              JRZ     ..EL    ;POSSIBLE END OF FILE
06A3' 77                MOV     M,A
06A4' 18F7              JMPR    ..L3
06A6' 1E01      ..EL:   MVI     E,1     ;INITIALIZE
06A8' CD 0462'  ..EL0:  CALL    RIFF
06AB' 2009              JRNZ    ..EL1
06AD' 1C                INR     E       ;COUNT QUES
06AE' 3E07              MVI     A,MAX   ;LOOK FOR EOF
06B0' BB                CMP     E       ;FOUND MAX?
06B1' 20F5              JRNZ    ..EL0   ;NOPE
06B3' C3 057D'          JMP     LADR    ;YEP, PRINT END ADDR
06B6' 72        ..EL1:  MOV     M,D
06B7' 23                INX     H
06B8' 1D                DCR     E       ;RESTORE
06B9' 20FB              JRNZ    ..EL1
06BB' 77                MOV     M,A     ;REAL BYTE
06BC' 18DF              JMPR    ..L3
                ;
                ; THIS IS THE BREAKPOINT "TRAP" HANDLING
                ; ROUTINE.  ALL USER REGISTERS ARE SAVED
                ; FOR DISPLAY PURPOSES, AND THE CONTENTS
                ; ARE RESTORED WHEN EXECUTING A "GO" (G)
```

```
                        ; COMMAND.
                        ;
06BE' E5                RESTART: PUSH   H       ;PUSH ALL REGISTERS
06BF' D5                         PUSH   D
06C0' C5                         PUSH   B
06C1' F5                         PUSH   PSW
06C2' CD 05AC'                   CALL   MEMSIZ  ;GET MONITOR'S STACK VALUE
06C5' EB                         XCHG
06C6' 21 000A                    LXI H, 10      ;GO UP 10 BYTES IN STACK
06C9' 39                         DAD    SP
06CA' 0604                       MVI    B,4     ;PICK OFF REG.
06CC' EB                         XCHG
06CD' 2B                ..R0:    DCX    H
06CE' 72                         MOV    M,D     ;SAVE IN WORKAREA
06CF' 2B                         DCX    H
06D0' 73                         MOV    M,E
06D1' D1                         POP    D
06D2' 10F9                       DJNZ   ..R0
06D4' C1                         POP    B
06D5' 0B                         DCX    B       ;ADJUST P.C. VALUE
06D6' F9                         SPHL           ;SET MONITOR STACK
06D7' 21 0025                    LXI H, TLOCX
06DA' 39                         DAD    SP
06DB' 7E                         MOV    A,M
06DC' 91                         SUB    C       ;LOOK FOR A TRAP/MATCH
06DD' 23                         INX    H
06DE' 2004                       JRNZ   ..R1
06E0' 7E                         MOV    A,M
06E1' 90                         SUB    B
06E2' 280C                       JRZ    ..R3    ;NO TRAP HERE
06E4' 23                ..R1:    INX    H
06E5' 23                         INX    H
06E6' 7E                         MOV    A,M
06E7' 91                         SUB    C       ;TEST FOR 2ND TRAP
06E8' 2005                       JRNZ   ..R2
06EA' 23                         INX    H
06EB' 7E                         MOV    A,M
06EC' 90                         SUB    B
06ED' 2801                       JRZ    ..R3
06EF' 03                ..R2:    INX    B       ;NO TRAPS SET, RE-ADJUST P.C.
06F0' 21 0020           ..R3:    LXI H, LLOCX
06F3' 39                         DAD    SP
06F4' 73                         MOV    M,E     ;STORE USER H&L
06F5' 23                         INX    H
06F6' 72                         MOV    M,D
06F7' 23                         INX    H
06F8' 23                         INX    H
06F9' 71                         MOV    M,C     ;AND USER P.C.
06FA' 23                         INX    H
06FB' 70                         MOV    M,B
06FC' C5                         PUSH   B
06FD' 0E40                       MVI    C,'@'   ;DISPLAY BREAK ADDRESS.
06FF' CD 0478'                   CALL   CO
0702' E1                         POP    H
0703' CD 057D'                   CALL   LADR
```

```
0706' 21 0025            LXI H,  TLOCX
0709' 39                 DAD     SP
070A' 01 0200            LXI     B,200H
070D' 5E          ..R4:  MOV     E,M      ;REPLACE BYTES TAKEN FOR TRAP
070E' 71                 MOV     M,C      ;ZERO OUT STORAGE AREA
070F' 23                 INX     H
0710' 56                 MOV     D,M
0711' 71                 MOV     M,C
0712' 23                 INX     H
0713' 7B                 MOV     A,E
0714' B2                 ORA     D        ;DO NOTHING IF ZERO
0715' 2802               JRZ     ..R5
0717' 7E                 MOV     A,M
0718' 12                 STAX    D        ;STORE BYTE
0719' 23          ..R5:  INX     H        ;SAME THING
071A' 10F1               DJNZ    ..R4     ;FOR OTHER BREAKPOINT
071C' 08                 EXAF             ;GET ALTERNATE SET OF REG.'S
071D' D9                 EXX
071E' E5                 PUSH    H        ;AND STORE IN WORKSPACE
071F' D5                 PUSH    D
0720' C5                 PUSH    B
0721' F5                 PUSH    PSW
0722' DDE5               PUSH    X
0724' FDE5               PUSH    Y
0726' ED57               LDAI             ;GET INTERUPT VECTOR BYTE
0728' 47                 MOV     B,A
0729' ED5F               LDAR             ;GET REFRESH BYTE
072B' 4F                 MOV     C,A
072C' C5                 PUSH    B        ;SAVE
072D' C3 0062'           JMP     START    ;BACK TO START
                  ;
                  ; THIS IS THE INTERNAL KEYBOARD
                  ; HANDLING ROUTINE. IT WILL IGNORE
                  ; RUBOUTS (0FFH) AND BLANKS (00),
                  ; AND IT WILL NOT ECHO CR'S & N'S.
                  ; (NO N'S FOR THE "NULL" COMMAND).
                  ; IT CONVERTS LOWER CASE TO UPPER
                  ; CASE FOR THE LOOK-UP OF COMMANDS.
                  ;
                  ; OTHER CHARACTERS ARE ECHOED AS THEY
                  ; ARE RECIEVED.
                  ;
0730' CD 0613'    KI:    CALL    CI       ;GET CHARACTER FROM CONSOLE
0733' E67F               ANI     7FH      ;CLEAR PARITY BIT
0735' C9                 RET
                  ;
0736' CD 0730'    TI:    CALL    KI
0739' 3C                 INR     A        ;IGNORE RUBOUTS
073A' F8                 RM
073B' 3D                 DCR     A        ;IGNORE NULLS
073C' C8                 RZ
073D' FE4E               CPI     'N'      ;IGNORE N'S FOR NULL CMND
073F' C8                 RZ
0740' FE6E               CPI     'n'
0742' 2810               JRZ     ..T
```

```
0744' FE0D              CPI     CR      ;IGNORE CR'S
0746' C8                RZ
0747' C5                PUSH    B
0748' 4F                MOV     C,A
0749' CD 0478'          CALL    CO
074C' 79                MOV     A,C
074D' C1                POP     B
074E' FE40              CPI     'A'-1   ;CONVERT TO UPPER CASE
0750' D8                RC
0751' FE7B              CPI     'z'+1
0753' D0                RNC
0754' E65F      ..T:    ANI     05FH
0756' C9                RET
                ;
                ; THIS ROUTINE ALLOWS EXAMINATION OF
                ; ANY INPUT PORT, OR THE SENDING OF
                ; ANY VALUE TO ANY OUTPUT PORT.
                ;
                ; QO<N>,<V>[CR]
                ;       OUTPUT TO PORT <N>, THE VALUE <V>
                ;
                ; QI<N>[CR]
                ;       DISPLAY THE PORT <N>
                ;
0757' CD 0736'  QUERY:  CALL    TI
075A' FE4F              CPI     'O'
075C' 281C              JRZ     QU0
075E' FE49              CPI     'I'
0760' C2 0452'          JNZ     ERROR
0763' CD 0533'          CALL    EXPR1
0766' C1                POP     B
0767' ED58              INP     E
0769' 0608      BITS:   MVI     B,8     ;DISPLAY 8 BITS
076B' CD 0476'          CALL    BLK
076E' CB23      ..Q2:   SLAR    E
0770' 3E18              MVI     A,'0' >1
0772' 8F                ADC     A       ;MAKE "0" OR "1"
0773' 4F                MOV     C,A
0774' CD 0478'          CALL    CO
0777' 10F5              DJNZ    ..Q2
0779' C9                RET
077A' CD 0535'  QU0:    CALL    EXPR
077D' D1                POP     D
077E' C1                POP     B
077F' ED59              OUTP    E
0781' C9                RET
                ;
                ; THIS ROUTINE VERIFIES THE CONTENTS
                ; OF ONE MEMORY BLOCK WITH ANOTHER.
                ;
                ; V<ADDR1>,<ADDR2>,<ADDR3>
                ;       VERIFY FROM <1> THRU <2> WITH
                ; THE CONTENTS OF MEMORY BEGINNING AT <3>
                ;
0782' CD 0528'  VERIFY: CALL    EXPR3   ;GET 3 PARAMETERS
```

```
0785' 0A          VERI0:  LDAX    B
0786' BE                  CMP     M
0787' 2805                JRZ     ..B
0789' C5                  PUSH    B
078A' CD 014B'            CALL    CERR    ;DISPLAY ERRORS
078D' C1                  POP     B
078E' 03          ..B:    INX     B
078F' CD 0561'            CALL    HILOX
0792' 18F1                JMPR    VERI0
                  ;
                  ; <SYSTEM I/O LOOK-UP TABLE>
                  ;
                  ; THE FIRST CHARACTER IS THE DEVICE NAME
                  ; (ONE LETTER) AND THE NEXT FOUR ARE THE
                  ; NAMES OF THE FOUR POSSIBLE DRIVERS TO BE
                  ; ASSIGNED.
                  ;
0794'             LTBL:
0794' 43          .BYTE   'C'       ;CONSOLE ASSIGNMENTS
0795' 54          .BYTE   'T'       ;CTTY   T=TELEPRINTER
0796' 43          .BYTE   'C'       ;CCRT   C=CRT (VIDEO MONITOR)
0797' 42          .BYTE   'B'       ;BATCH= COMMANDS FROM READER
0798' 55          .BYTE   'U'       ;CUSE   USER
                  ;
0799' 52          .BYTE   'R'       ;READER ASSIGNMENTS
079A' 54          .BYTE   'T'       ;RTTY
079B' 50          .BYTE   'P'       ;RPTR   P=PAPER TAPE
079C' 43          .BYTE   'C'       ;RCAS   C=CASSETTE
079D' 55          .BYTE   'U'       ;RUSER  USER
                  ;
079E' 50          .BYTE   'P'       ;PUNCH ASSIGNMENTS
079F' 54          .BYTE   'T'       ;PTTY
07A0' 50          .BYTE   'P'       ;PPTP
07A1' 43          .BYTE   'C'       ;PCAS   C=CASSETTE
07A2' 55          .BYTE   'U'       ;PUSER  USER
                  ;
07A3' 4C          .BYTE   'L'       ;LIST ASSIGNMENTS
07A4' 54          .BYTE   'T'       ;LTTY   LIST=TELEPRINTER
07A5' 43          .BYTE   'C'       ;LCRT   LIST=CRT
07A6' 4C          .BYTE   'L'       ;LINE PRINTER
07A7' 55          .BYTE   'U'       ;LUSER  USER
                  ;
                  ;
                  ; THIS IS A SHORT PROGRAM, EXECUTED
                  ; UPON EXECUTING A "GO" COMMAND. IT
                  ; IS PLACED IN THE WORK AREA WHEN
                  ; THE MONITOR IS INITIALIZED, AS IT
                  ; REQUIRES RAM FOR PROPER OPERATION.
                  ;
07A8'             EXIT:             ;EXIT ROUTINE (LOADS ALL REGISTERS)
07A8' C1                  POP     B
07A9' 79                  MOV     A,C
07AA' ED4F                STAR
07AC' 78                  MOV     A,B
07AD' ED47                STAI
```

```
07AF' FDE1              POP     Y
07B1' DDE1              POP     X
07B3' F1                POP     PSW
07B4' C1                POP     B
07B5' D1                POP     D
07B6' E1                POP     H
07B7' 08                EXAF
07B8' D9                EXX
07B9' D1                POP     D
07BA' C1                POP     B
07BB' F1                POP     PSW
07BC' E1                POP     H
07BD' F9                SPHL
07BE' 00                NOP             ;RESERVED FOR ENABLE INTERUPTS
07BF' 21 0000           LXI     H,0
07C2' C3 0000           JMP     0
                ;
07C5' 0000              .WORD   0       ;STORAGE AREA FOR TRAP DATA
07C7' 00                .BYTE   0
07C8' 0000              .WORD   0
07CA' 00                .BYTE   0
                ;
                ; DISPLACEMENTS OF REGISTER
                ; STORAGE FROM NORMAL STACK
                ; LOCATION.
                ;
07CB'           ENDX:
                ;
                        ALOC    = 15H
                        BLOC    = 13H
                        CLOC    = 12H
                        DLOC    = 11H
                        ELOC    = 10H
                        FLOC    = 14H
                        HLOC    = 31H
                        LLOC    = 30H
                        PLOC    = 34H
                        SLOC    = 17H
                        TLOC    = 35H
                        TLOCX   = 25H
                        LLOCX   = 20H
                ;
                        APLOC   = 09H
                        BPLOC   = 0BH
                        CPLOC   = 0AH
                        DPLOC   = 0DH
                        EPLOC   = 0CH
                        FPLOC   = 08H
                        HPLOC   = 0FH
                        LPLOC   = 0EH
                        XLOC    = 07
                        YLOC    = 05
                        RLOC    = 02
                        ILOC    = 03
                ;
```

```
                        ;
                        ;
                        ; THIS IS THE TABLE USED TO DETERMINE
                        ; A VALID REGISTER IDENTIFIER, AND IT'S
                        ; DISPLACEMENT FROM THE STACK POINTER.
                        ;
                        ; POSITION ONE= REGISTER NAME, WITH BIT 7 INDICATING
                        ; END OF TABLE.
                        ;
                        ; POSITION TWO= BIAS FROM CURRENT STACK LEVEL OR'ED
                        ; WITH A TWO-BIT FLAG.  00XXXXXX=BYTE
                        ;                       10XXXXXX=WORD
                        ;                       11XXXXXX=SPECIAL FOR "M" REG.
                        ;
07CB'                   ACTBL:          ;NORMAL SET OF REGISTERS (8080)
                        ;               ;PLUS THE INTERUPT REGISTER ("I")
                        ;
07CB' 4115                      .BYTE   'A',    ALOC    !0
07CD' 4213                      .BYTE   'B',    BLOC    !0
07CF' 4312                      .BYTE   'C',    CLOC    !0
07D1' 4411                      .BYTE   'D',    DLOC    !0
07D3' 4510                      .BYTE   'E',    ELOC    !0
07D5' 4614                      .BYTE   'F',    FLOC    !0
07D7' 4831                      .BYTE   'H',    HLOC    !0
07D9' 4C30                      .BYTE   'L',    LLOC    !0
07DB' 4DF1                      .BYTE   'M',    HLOC    !0C0H
07DD' 50B4                      .BYTE   'P',    PLOC    !080H
07DF' 5397                      .BYTE   'S',    SLOC    !080H
07E1' 4903                      .BYTE   'I',    ILOC    !0
                        ;
07E3'                           .BLKB    4
                        ;
07E7'                   PRMTB:  ;ADDITIONAL SET OF REGISTERS (Z-80)
                        ;
07E7' 4109                      .BYTE   'A',    APLOC   !0
07E9' 420B                      .BYTE   'B',    BPLOC   !0
07EB' 430A                      .BYTE   'C',    CPLOC   !0
07ED' 440D                      .BYTE   'D',    DPLOC   !0
07EF' 450C                      .BYTE   'E',    EPLOC   !0
07F1' 4608                      .BYTE   'F',    FPLOC   !0
07F3' 480F                      .BYTE   'H',    HPLOC   !0
07F5' 4C0E                      .BYTE   'L',    LPLOC   !0
07F7' 4DCF                      .BYTE   'M',    HPLOC   !0C0H
07F9' 5887                      .BYTE   'X',    XLOC    !080H
07FB' 5985                      .BYTE   'Y',    YLOC    !080H
07FD' 5202                      .BYTE   'R',    RLOC    !0
07FF' 80                        .BYTE   80H
                        ;
0800'                           Z:
                                ;END OF PROGRAM
                        ;
                        ;
                        ;
0000'                   .END    BASE
```

+++++ SYMBOL TABLE +++++

ACTBL	07CB'	AHEAD	0045'	ALOC	0015	APLOC	0009
ASSIGN	00BE'	BASE	0000'	BATCH	0002	BEGIN	0031'
BELL	0007	BITS	0769'	BLK	0476'	BLOC	0013
BPLOC	000B	BYE	010F'	CCHK	044C'	CCRT	0001
CERR	014B'	CI	0613'	CI1	0623'	CI2	062F'
CILOC	0800'	CLOC	0012	CMSK	00FC	CO	0478'
CO0	0489'	CO1	0496'	COLOC	0803'	COMP	013C'
CONV	04F5'	COPCK	05FF'	CPLOC	000A	CR	000D
CRLF	0504'	CRTBE	0080	CRTDA	0001	CRTI	0005
CRTIN	0626'	CRTO	0005	CRTOUT	048C'	CRTS	0004
CS0	0517'	CS1	051C'	CS2	0520'	CS3	0523'
CSLOC	0818'	CSTS	050C'	CTTY	0000	CUSE	0003
DISP	015D'	DL0	064B'	DLOC	0011	DONE	0272'
DONE2	0359'	DPLOC	000D	ELOC	0010	ENDX	07CB'
EOF	0174'	EPLOC	000C	ERR2	026F'	ERR3	0269'
ERROR	0452'	EX0	0538'	EX1	053B'	EXF	055A'
EXIT	07A8'	EXLF	04FF'	EXPR	0535'	EXPR1	0533'
EXPR3	0528'	FALSE	0000	FIL	0000	FILL	0190'
FIND	0332'	FINDC	032D'	FLOC	0014	FOUND	0341'
FPLOC	0008	GOTO	019D'	HELLO	005D'	HEXN	0571'
HILO	0567'	HILOX	0561'	HLOC	0031	HLSP	0473'
HPLOC	000F	ILOC	0003	IOBYT	0002	IOCHK	010B'
IOSET	0106'	J	0821'	KI	0730'	LADR	057D'
LBYTE	0582'	LCRT	0040	LE0	0599'	LEAD	0596'
LF	000A	LFADR	0470'	LINE	0080	LLOC	0030
LLOCX	0020	LMSK	003F	LNLOC	0812'	LO	049A'
LOAD	0681'	LOD0	0228'	LOD4	0264'	LODCB	029E'
LODR	027E'	LPLOC	000E	LTBL	0794'	LTTY	0000
LULOC	0815'	LUSER	00C0	MARK	0591'	MAX	0007
MEMCK	059F'	MEMSIZ	05AC'	MOVE	0209'	MSG	0021'
MSGL	0010	NIBBLE	05D3'	NN	0000	NULL	04E9'
PADR	05E3'	PBYTE	05E8'	PCAS	0020	PCASO	0007
PCASS	0006	PCHK	0604'	PCSBE	0080	PEOL	04AD'
PLOC	0034	PMSK	00CF	PO	04B4'	PO0	04BF'
PO1	04C9'	PPTP	0010	PRMTB	07E7'	PTPL	080C'
PTTY	0000	PULOC	080F'	PUSER	0030	PUTA	011D'
QCHK	0607'	QU0	077A'	QUERY	0757'	RCAS	0008
RCP	0003	RCSD	0007	RCSDA	0001	RCSS	0006
READ	0214'	RESTAR	06BE'	RI	0633'	RI0	0642'
RI1	0653'	RI2	0657'	RI3	065C'	RI4	0663'
RI5	066E'	RI6	0672'	RIBBLE	05D0'	RID	0659'
RIFF	0462'	RIX	067B'	RLOC	0002	RMSK	00F3
RPTPL	0806'	RPTR	0004	RST7	0038	RTTY	0000
RUB	00FF	RULOC	0809'	RUSER	000C	SBYTE	02AE'
SENSE	0003	SIZE	0469'	SLOC	0017	STAR0	006E'
START	0062'	STKIT	005A'	STORE	02C1'	SUBS	02CD'
SWITCH	00FF	TBL	008A'	TELL	034F'	TEST	01EB'
TI	0736'	TLOC	0035	TLOCX	0025	TOM	043D'
TOM1	0440'	TRAP	001E'	TRUE	FFFF	TTI	0001
TTO	0001	TTS	0000	TTYBE	0080	TTYDA	0001
TTYIN	061A'	TTYOUT	047F'	TYPE	02F6'	UNLD	04D1'
USER	0800'	VERI0	0785'	VERIFY	0782'	WAIT	04EC'
WHERE	0316'	WRITE	035E'	XAM	039E'	XLOC	0007
YLOC	0005	Z	0800'				

NO PROGRAM ERRORS