THE ZAP MONITOR

A. FEATURES

The ZAP Monitor is a 1K version of TDL's 2K ZAPPLE Monitor. It is relocatable (can be placed anywhere in memory), expandable ("modules" of additional commands can be tacked on at the end, like cars on a freight train.), and quite powerful as a system executive.

The expandable feature should be of great interest to the user. Since it is designed in a modular fashion, and since the ZAPPLE is its direct parent, this monitor features tremendous expandability - either of routines generated by the user, or by routines provided by Technical Design Labs. Several "modules" which will be of great interest include powerful "breakpoint", "search" and "register display" commands. Paper tapes of these modules will be available from TDL in the early fall. (Contact us for the latest word on availability.)

B. LOADING PROCEDURE

The loading procedure is presented on the following two pages exactly as it was prepared on the computer.
.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.
/
APPENDIX A. SUPPORT PROGRAMS FOR RELOCATING BOOT LOADER, V3.2
UART STYLE BOOT LOADER Routines

.TLIST
;
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 2 ;MOVE POINTER
0027 77 MOV M, A ;SAVE DATA
0028 20F6 JRNZ ..RDR ;GET MORE DATA OR
002A E9 PCHL ; GO TO LOADER
;
; ALTAIR SICO 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

; PT CO 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 ALT AIR 2S10.
;
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
.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 LAH

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                JNZ  ..RDR ;WALK OVER LEADER
0026 2D                  DCR  L ;MOVE POINTER
0027 77                  MOV  M,A ;SAVE DATA
0028 20F6                JNZ  ..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

;


.TLIST

; 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,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 DAX 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 ;CAN'T GO TO ZERO
016C B5 ORA L
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
0199  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  DD8E00   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
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 boundary. Running the monitor on other than a page border sounds a little pointless in any case.

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

The following are the commands and operating symbols of the ZAP Monitor.

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>DISPLAY COMMAND - this command displays the contents of memory in base hex. Memory is displayed 16 bytes per line, with the starting address of the line given as the first information on the line. In use, first the command is given, then the starting address, the ending address and a carriage return. The form is: D000,FFF(cr). (This would display memory from 000 to FFF.)</td>
</tr>
<tr>
<td>E</td>
<td>END OF FILE - this command outputs the end of file pattern for the checksum loader. It is used after punching a block of memory with a &quot;W&quot; command. An address parameter for the End of File may be given. For use, when the file being dumped is finished, type: E(cr).</td>
</tr>
<tr>
<td>F</td>
<td>FILL - This command fills a block of memory with a specific value. It is handy for initializing a block to a specific value (such as for tests, zeroing memory when starting up, etc.) In use, first the command, then the starting address, ending address, and the value to be entered, followed by a carriage return. The form is FL000,1FFF,AA(cr). This would fill the block 1000 to 1FFF with AA.</td>
</tr>
<tr>
<td>G</td>
<td>GOTO - this command causes the processor to go to the specific address named and start executing. If a Return command is included in the program, the processor may jump back to the monitor after execution of the program. (RETURN is C9 hex). To use, the command is followed by the address chosen to execute from and a carriage return. The form is: G2FD4(cr). The processor will goto address 2FD4 and execute.</td>
</tr>
</tbody>
</table>
MEMORY TEST - this is a "hard" memory test which will locate bad bits and represent them in their binary form. It is not meant to be the definitive memory test, but rather serves as an aid. It can also serve to very quickly locate accidentally or mistakenly protected areas of memory. It is non destructive of the memory contained in the area being examined.

In use, the command is followed by starting and ending addresses. A read/complement/write is executed and if any errors are found, the bad address will be printed followed by the binary representation of the bit pattern. The form is: J000,FF(cr). If address AA were bad on its fourth bit, the processor will print back AA 00010000, the "1" representing the bad bit found.

LOAD A BINARY FILE - This reads a binary file, either from cassette or tape. The form is: L000 (cr). This would load a binary file starting at address 000. To use, enter the command and the starting address, type carriage return, and start the reader with nulls on the tape.

MOVE COMMAND - this command can move a block of memory from one location to another. This command should be used with some caution as careless placing could "smash" memory locations containing wanted data.

To use, type M followed by the starting address of the memory block to be moved, the ending address of the block to be moved, and the starting address of the new location. The form is: M000,AA,CC. This would move the block of memory starting at location 000 and extending to location AA up to location CC.

NULL - this command may be used to print nulls on paper tape as a leader. To use simply type N - and nulls will be punched.
Q

OUTPUT OR DISPLAY FROM/TO I/O PORTS - this command instructs the processor where to look for or where to send data to. To use, enter the command, indicating whether the processor is to input or output, name the port, and name the value to be output, if you are outputting. The form is: QO∅, AA or QI∅. The first would output an AA to port ∅, the second would input from port zero.

R

READ CHECKSUMMED HEX FILE - this command reads the check-summed hex files for both the normal Intel format and the TDL relocating format. On both files, a "bias" (a shift in the address) may be added which will allow 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 may wrap around. When used with the TDL relocating assembler, it allows generating a program to execute anywhere, and to be stored anywhere else in memory. When loading a relocatable file, an additional parameter may be added which represents the actual execution address desired. This may also be any location in memory.

To use, with a normal file, type R(cr) and start the reader.
With a relocating file, the following examples should clarify the use of bias.

R(cr) = ∅ bias, ∅ execution address
Rl(cr) = 1 bias, ∅ execution address
R,1(cr) = ∅ bias, 1 execution address
Rl,1(cr) = 1 bias, 1 execution address

S

SINGLE BYTE INSPECT AND MODIFY - this command allows single bytes of memory to be examined and modified or not as the user desires.

To use, give the command followed by an address and push the space bar - the data at that address will be displayed followed by a "-". If you wish to change the data at that address, simply type in the new data in hex and press the space bar. The old data will be replaced, and then the next byte of data will appear. If you wish to retain the old data,
simply press the space bar and the next byte will appear. Typing a carriage return ends the sequence.

BINARY DUMP - 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 checksum. The format which will be generated is a leader, 8-0FFH's, and a trailer. The rub-outs are called file ques and are detected and counted to determine the start and end of files.
To use, type the command followed by the starting and ending addresses, start the reader and (cr). The form is: U00,FF(start reader - cr). This would generate a binary tape in the above format of the core contained in memory location 00 to FF.

HEX DUMP - this routine dumps memory in the standard Intel-style hex file format. The start and end parameters are required and the End of File should be separately generated with the "E" command.
To use, enter the command, starting address, ending address, start the reader, (cr). When dump finished, type E(cr) to generate end of file. The form is: W00,FF(start punch - cr) ----E(cr). (N here is optional).

TOP OF MEMORY - this command locates and names the top byte of RAM in the system. It does not include the space the monitor is occupying. Simply type Z - no (cr) is needed. The top of memory will be displayed in hex.

HEXDECIMAL MATH - this command allows hex addition and subtraction to be executed.
To use, type H, and the two hex figures to be added and subtracted. The form is: H00,11(cr). The computer will print out first the hex sum and then the hex difference, in hex.

This concludes the command set of the ZAP Monitor.

In addition to these commands there are two symbols which you will observe. The first is an *, which is an error message. The second is a > (greater than) which is a prompter basically saying "OK, continue...".
To interrupt a routine such as a D or J command, just type a CONTROL C. This ends the routine.

D. ZPU FINAL CHECKOUT USING MONITOR

Assembly and electrical checkout of the ZPU was conducted elsewhere. However, only operation will show if the ZPU is actually operating correctly. The monitor is the best means of achieving this. Load the monitor as per the preceding instructions, and experiment with its various commands. The FILL and DISPLAY, plus MOVE and J commands provide good exercise for the processor and if they seem to function normally, all is probably well.

E. SOURCE LISTING

The following pages are an "off the printer" copy of the ZAP Monitor source code. It is provided for your understanding, plus as an invitation to experiment with Z-80 programming which can be quite exciting given 696 opcodes.
<< ZAP 1-K MONITOR SYSTEM >>

by

TECHNICAL DESIGN LABS, INC.
RESEARCH PARK
PRINCETON, NEW JERSEY 08540

COPYRIGHT JAN. 1977 TDL INC.

ASSEMBLED by Roger Amidon

.PREL #THIS MONITOR SUPPLIED IN RELOCATING FORMAT

LENGTH = 2 #SIZE OF THIS MONITOR

.TITLE "<Zap Monitor, Version 2.0, Jan. 16 1977>"
.SBTIL # Copyright 1977 by TECHNICAL DESIGN LABS, INC./

# I/O DEVICES

#TELEPRINTER

0001 TTI = 1 #DATA IN PORT
0001 TTD = 1 #DATA OUT PORT
0000 TTS = 0 #STATUS PORT (IN)
0001 TTYDA = 1 #DATA AVAILABLE MASK BIT
0080 TTYBE = 80H #CTRL BUFFER EMPTY MASK

0003 RCP = 3 #READER CONTROL PORT.
#THIS PORT IS PULSED ONCE
#FOR EACH READER REQUEST
#TO SUPPORT A CONTROLLED
#READER.

<CONSTANTS>

0000 I = 0 #I' REG. VALUE
0000 FALSE = 0 #ISN'T SO
FFFF TRUE = # FALSE #IT IS SO
0000 CR = ODH #ASCII CARRIAGE RETURN
0000 LF = OAH #ASCII LINE FEED
0007 BELL = 7 #DING
00FF RUB = OFFH #RUB OUT
0000 FILL = CO #FILL CHARACTERS AFTER CRLF
0007 MAX = 7 #NUMBER OF QUEST IN EOF

PROGRAM CODE BEGINS HERE

0000 C3 0308 ZAP: JMP BEGIN #GO AROUND VECTORS
# GET MEMORY SIZE,
# AND CONTINUE AHEAD
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.

'STS'-
RETURNS TRUE (OFFH IN 'A' REG.) IF THERE IS
SOMETHING WAITING, AND ZERO (00) IF NOT.

'IOCH'-
RETURNS WITH THE CURRENT I/O CONFIGURATION
BYTE IN 'A' REGISTER.

'IOSET'-
I/O CANNOT BE MODIFIED IN THIS 1K VERSION

'MEMCH'-
RETURNS WITH THE HIGHEST ALLOWED USER
MEMORY LOCATION. 'B'=HIGH BYTE, 'A'=LOW.

'TRAP'-
THIS IS THE 'BREAKPOINT' ENTRY POINT.
NOT USED IN THE 1K VERSION, GOES TO THE
'ERROR' ROUTINE TO RESET THE MONITOR'S
STACK.

<table>
<thead>
<tr>
<th>Address</th>
<th>C3 0374</th>
<th>C3 037D</th>
<th>C3 0222</th>
<th>C3 0233</th>
<th>C3 0222</th>
<th>C3 0282</th>
<th>3E00</th>
<th>0397</th>
<th>017</th>
<th>0017</th>
<th>0018</th>
<th>001B</th>
<th>001E</th>
<th>0021</th>
<th>0022</th>
<th>0024</th>
<th>0027</th>
</tr>
</thead>
<tbody>
<tr>
<td>0003</td>
<td>JMP CI</td>
<td>JMP RI</td>
<td>JMP CO</td>
<td>JMP PO</td>
<td>JMP CO</td>
<td>JMP CSTS</td>
<td>MVI A,0</td>
<td>I/O CHECK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0015</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0017</td>
<td>I/OSET</td>
<td>RET</td>
<td>I/OSET</td>
<td>I/OSET</td>
<td>I/OSET</td>
<td>I/OSET</td>
<td>CALL MEMSIZ</td>
<td>RE-ESTABLISH A STACK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0018</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0021</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0022</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0027</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
; MONITOR NAME & VERSION
; 0029' 0DDA000000 MSG: .BYTE CR,LF,FIL,FIL,FIL
002E' 5A61702056 .ASCII 'Zap V'
0033' 322E30 .ASCII '2.0'

; 0034' MSGL = .-MSG
; 0036' 0038' STACK = .-2 ; A FAKE STACK TO GET STARTED
; 0038' F9 AHEAD: .WORD AHEAD ; AFTER MEMORY SIZE
; 0039' 060D F9 AHEAD: SPHL ; SET TRUE STACK
; 003B' CD 01F2' MVI B,MSGL ; SAY HELLO TO THE FOLKS
; 003E' 0E3E' CALL TOD ; OUTPUT SIGN-ON MSG
; 0040' 21 003E' START: MVI C,<> ; PROMPT CHARACTER
; 0040' 003E' CALL LXI H,START ; MAIN 'WORK' LOOP
; 0044' 0E44' PUSH H ; SET UP A RETURN TO HERE
; 0047' CD 0278' CALL CRLF
; 0049' CD 022E' CALL CALL
; 004A' CD 030C' STAKH: CALL TI ; GET A CONSOLE CHARACTER
004D' E67F ANI 7FH ; IGNORE NULLS
004F' 2BF9 CALL JRZ STAR ; GET ANOTHER
0051' OE02 MVI C,2 ; SET-UP C REG.
0053' FE44 CALL CPI 'D' ; SEE IF 'DISPLAY COMMAND
0055' 2017 JRNZ EOF

; 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 MAY.
; 0057' CD 0273' DISP: CALL E3LF ; GET DISPLAY RANGE
; 005A' CD 021A' .DO: CALL LFADDR ; CRLF & PRINT ADDR.
; 005D' CD 0220' .DI: CALL BLK ; SPACE OVER
; 0060' 7E MOV A,M
; 0061' CD 02E3' CALL CALL LBYTE
; 0064' CD 02BD' CALL HILOX ; RANGE CHECK
; 0067' 7D MOV A,L
; 0068' E60F CALL CALL ANI OFH ; SEE IF TIME TO CRLF
; 006A' 20F1 JRNZ ..DI
; 006C' 18EC JMPR ..DO

; 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,
; WHICH WILL BE INCLUDED IN THE END FILE.
006E' FE45 EOF: CALL CPI 'E' ; SEE IF 'EOF'
0070' 201A JRNZ FILL
0072' CD 0296' CALL EXPRI ; GET OPTIONAL ADDR.
0075' CD 022E' CALL PEDL ; CRLF TO PUNCH
0078' OE3A MVI C,<> ; FILE MARKER Cue
007A' CD 0233' CALL PO
007D AF XRA A \#ZERO LENGTH
007E CD 034D CALL PBYTE
0081 E1 POP H
0082 CD 0348 CALL PADR \#PUNCH OPTIONAL ADDR.
0085 AF XRA A \#FILE TYPE=0
0086 CD 034D CALL PBYTE \#PUNCH IT
0089 C3 025F JMP NULL \#TRAILER & RETURN

THIS COMMAND WILL FILL A BLOCK OF MEMORY WITH A VALUE. IE: FO,FFF,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.)

008C FE46 FILL: CPI 'F' \#SEE IF 'FILL'
008E 200C JRNZ GOTO \#GET PARAMETERS
0090 CD 0288 CALL EXPR3 \#STORE THE BYTES
0093 71 ..F: MOV M,C \#GET 3 PARAMETERS
0094 CD 02C3 CALL HILO \#STORE THE BYTES
0096 30FA JRNZ ..F \#GET 3 PARAMETERS
0097 30FA POP D \#RESTORE STACK
0099 D1 JMPR START \#IN CASE OF ACCIDENTS

THIS COMMAND ALLOWS EXECUTION OF ANOTHER PROGRAM.

009C FE47 GOTO: CPI 'G' \#SEE IF 'GOTO'
009E 2006 JRNZ TEST \#GET AN ADDRESS TO GO TO
00A0 CD 0296 CALL EXPR1 \#EXECUTE
00A3 C3 0278 JMP CRLF \#EXECUTE

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> O4" WHERE, IN THIS PARTICULAR EXAMPLE, BIT 2 IS THE BAD BIT.
BIT LOCATION OF THE FAILURE IS EASILY DETERMINED. NON-R/W MEMORY WILL DISPLAY \#ADDR> FF (ALL BITS BAD)

00A6 FE4A TEST: CPI 'J' \#SEE IF 'TEST'
00A8 2018 JRNZ MOVE
00AA CD 0273 CALL EXLF \#GET TWO PARAMS
00AD 7E MOV A,M \#READ A BYTE
00AE 47 MOV B,A \#SAVE IN B REG.
00AF 2F CMA
00B0 77 MOV M,A \#READ/COMPLEMENT/WRITE & COMPARE
00B1 AE XRA M \#SKIP IF ZERO (OK)
00B2 280B JRZ ..T2 \#SAVE BAD BYTE
00B4 08 EXAF \#PRINT BAD ADDR
00B5 CD 021D CALL HLSP
TDL Z80 RELOCATING ASSEMBLER VERSION 1.2
<Zap Monitor, Version 2.0, Jan. 16 1977>
Copyright 1977 by TECHNICAL DESIGN LABS, INC.

0088 08 EXAF GET BAD BYTE BACK
008B 08 CALL LBYTE PRINT IT
008C 08 CALL CRLF
008F 70 .T2 MOV M, B REPLACE BYTE
0090 CD 02E3 CALL HILOX RANGE TEST
0093 18E8 JMPR ..I1

; 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>

00C5 FE4D MOVE CPI 'M' SEE IF 'MOVE'
00C7 2008 JRNZ READ
00C9 0D 02B8 CALL EXPR3 GET 3 PARAMETERS
00CC 7E ..M MOV A,M PICK UP
00CD 02 STAX B PUT DOWN
00CE 03 INX B MOVE UP
00CF CD 02B8 CALL HILOX CHECK IF DONE
00D2 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<ADDR>[CR] = <1> BIAS, 0 EXECUTION ADDR.
; R,<ADDR>[CR] = 0 BIAS, <1> EXECUTION ADDR.
; R<ADDR1>,,<ADDR2>[CR] = <1> BIAS, 2 EXECUTION ADDR.

00D4 FE52 READ CPI 'R' SEE IF 'READ' COMMAND
00D6 C2 017C JRNZ SUBS
00D9 CD 0296 CALL EXPR1 GET BIAS, IF ANY
00DC 78 MOV A,B LOOK AT DELIMITER
00DD C60D SUI CR ALL DONE?
00DF 47 MOV B,A SET UP RELOCATION OF 0
00E0 1F MOV C,A IF CR ENTERED
00E1 DF PDP D BIAS AMOUNT
00E2 2804 JRZ ..RO CR ENTERED
<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>00E4'</td>
<td>CD 0296'</td>
<td>CALL EXPI $GET RELOCATION</td>
</tr>
<tr>
<td>00E7'</td>
<td>C1</td>
<td>POP  B $ACTUAL RELOCATION VALUE</td>
</tr>
<tr>
<td>00E8'</td>
<td>EB</td>
<td>XCHG</td>
</tr>
<tr>
<td>00E9'</td>
<td>D9</td>
<td>EXX $HL'=BIAS, BC'=RELOCATION</td>
</tr>
<tr>
<td>00EA'</td>
<td>CD 0278'</td>
<td>CALL CRLF</td>
</tr>
<tr>
<td>00ED'</td>
<td>CD 020C'</td>
<td>CALL RIFF $GET A CHARACTER</td>
</tr>
<tr>
<td>00F0'</td>
<td>E67F</td>
<td>ANI  7FH $KILL PARITY BIT</td>
</tr>
<tr>
<td>00F2'</td>
<td>D63A</td>
<td>SUI</td>
</tr>
<tr>
<td>00F4'</td>
<td>47</td>
<td>MOV  B,A $SAVE CUE CLUE</td>
</tr>
<tr>
<td>00F5'</td>
<td>E6FE</td>
<td>ANI  OFEH $KILL BIT O</td>
</tr>
<tr>
<td>00F7'</td>
<td>20F4</td>
<td>JRNZ LODO $NO, KEEP LOOKING</td>
</tr>
<tr>
<td>00F9'</td>
<td>57</td>
<td>MOV  D,A $ZERO CHECKSUM</td>
</tr>
<tr>
<td>00FA'</td>
<td>CD 0162'</td>
<td>CALL SBYTE $GET FILE LENGTH</td>
</tr>
<tr>
<td>00FD'</td>
<td>5F</td>
<td>MOV  E,A $SAVE IN E REG.</td>
</tr>
<tr>
<td>00FE'</td>
<td>CD 0162'</td>
<td>CALL SBYTE $GET LOAD MSB</td>
</tr>
<tr>
<td>0101'</td>
<td>F5</td>
<td>PUSH PSW $SAVE IT</td>
</tr>
<tr>
<td>0102'</td>
<td>CD 0162'</td>
<td>CALL SBYTE $GET LOAD LSB</td>
</tr>
<tr>
<td>0105'</td>
<td>D9</td>
<td>EXX $CHANGE GEARS</td>
</tr>
<tr>
<td>0106'</td>
<td>D1</td>
<td>POP  D $RECOVER MSB</td>
</tr>
<tr>
<td>0107'</td>
<td>5F</td>
<td>MOV  E,A $FULL LOAD ADDR</td>
</tr>
<tr>
<td>0108'</td>
<td>C5</td>
<td>PUSH B $BC'=RELOCATION</td>
</tr>
<tr>
<td>0109'</td>
<td>D5</td>
<td>PUSH D $DE'=LOAD ADDR</td>
</tr>
<tr>
<td>010A'</td>
<td>E5</td>
<td>PUSH H $HL'=BIAS</td>
</tr>
<tr>
<td>010B'</td>
<td>19</td>
<td>DAD  D $BIAS+LOAD</td>
</tr>
<tr>
<td>010C'</td>
<td>E3</td>
<td>XTHL $RESTORE HL'</td>
</tr>
<tr>
<td>010D'</td>
<td>DDE1</td>
<td>POP  X $X=BIAS+LOAD</td>
</tr>
<tr>
<td>010F'</td>
<td>D9</td>
<td>EXX $DOWNSHIFT</td>
</tr>
<tr>
<td>0110'</td>
<td>E1</td>
<td>POP  H $HL=LOAD ADDR</td>
</tr>
<tr>
<td>0111'</td>
<td>CD 0162'</td>
<td>CALL SBYTE $GET FILE TYPE</td>
</tr>
<tr>
<td>0114'</td>
<td>3D</td>
<td>DCR  A $1=REL. FILE, 0=ABS.</td>
</tr>
<tr>
<td>0115'</td>
<td>78</td>
<td>MQV  A,B $SAVE CUE BIT</td>
</tr>
<tr>
<td>0116'</td>
<td>C1</td>
<td>POP  B $BC=RELOCATION</td>
</tr>
<tr>
<td>0117'</td>
<td>2003</td>
<td>JRNZ ..A $ABSOLUTE FILE</td>
</tr>
<tr>
<td>0119'</td>
<td>09</td>
<td>DAD  B $ELSE RELOCATE</td>
</tr>
<tr>
<td>011A'</td>
<td>DD09</td>
<td>DADX B $BOOTH X &amp; HL</td>
</tr>
<tr>
<td>011C'</td>
<td>1C</td>
<td>..A INR E $TEST LENGTH</td>
</tr>
<tr>
<td>011D'</td>
<td>1D</td>
<td>DCR  E $0=DONE</td>
</tr>
<tr>
<td>011E'</td>
<td>C8</td>
<td>RZ</td>
</tr>
<tr>
<td>011F'</td>
<td>3D</td>
<td>DCR  A $TEST CUE</td>
</tr>
<tr>
<td>0120'</td>
<td>2810</td>
<td>JRZ  LODR $RELATIVE</td>
</tr>
<tr>
<td>0122'</td>
<td>CD 0162'</td>
<td>CALL SBYTE $NEXT</td>
</tr>
<tr>
<td>0125'</td>
<td>CD 0175'</td>
<td>CALL SSTORE $STORE IT</td>
</tr>
<tr>
<td>0128'</td>
<td>20F8</td>
<td>JRNZ ..L1 $MORE COMM</td>
</tr>
<tr>
<td>012A'</td>
<td>CD 0162'</td>
<td>CALL SBYTE $GET CHECKSUM</td>
</tr>
<tr>
<td>012D'</td>
<td>28BE</td>
<td>JRZ  LODO $GOOD CHECKSUM</td>
</tr>
<tr>
<td>012F'</td>
<td>C3 001E'</td>
<td>JMP  ERROR $BAD, ABORT</td>
</tr>
<tr>
<td>0132'</td>
<td>2E01</td>
<td>MVI  L,1 $SET-UP BIT COUNTER</td>
</tr>
<tr>
<td>0134'</td>
<td>CD 0152'</td>
<td>CALL LODCB $GET THE BIT</td>
</tr>
<tr>
<td>0137'</td>
<td>3807</td>
<td>JRC  ..L3 $DOUBLE BIT</td>
</tr>
<tr>
<td>0139'</td>
<td>CD 0175'</td>
<td>CALL STORE $WRITE IT</td>
</tr>
<tr>
<td>013C'</td>
<td>20F6</td>
<td>JRNZ ..L1</td>
</tr>
<tr>
<td>013E'</td>
<td>18EA</td>
<td>JMPR LOD4 $TEST CHECKSUM</td>
</tr>
<tr>
<td>0140'</td>
<td>4F</td>
<td>MOV  C,A $SAVE LOW BYTE</td>
</tr>
<tr>
<td>0141'</td>
<td>CD 0152'</td>
<td>CALL LODCB $NEXT CONTROL BIT</td>
</tr>
<tr>
<td>0144'</td>
<td>47</td>
<td>MOV  B,A $SAVE HIGH BYTE</td>
</tr>
</tbody>
</table>
0145' D9  EXX  GET RELOCATION
0146' C5  PUSH B  INTO HL
0147' D9  EXX  RELOCATE
0148' E3  XTHL  LOW BYE
0149' 09  DAD B  STORE IT
014A' 7D  MOV A,L  STORE HL
014B' CD 0175' .CALL SIORE  RESTORE HL
014C' 7C  MOV A,H  HIGH BYTE
014D' E1  POP H  COUNT BITS
014E' 1BE7  JMPR .L5  DO THIS AGAIN
0150' 20D  DCR L  MORE LEFT
0152' 2007  JRNZ .LC1  GET NEXT
0153' 2012' CALL SBYTE  GET NEXT
0155' CD 0162'  DCR E  COUNT BYTES
015B' 67  MOV H,A  SAVE THE BITS
015C' 2E08  MOV L,8  B BITS/BYTE
015A' CD 0162'  .LC1: CALL SBYTE  GET A DATA BYTE
015F' CB42  SLAR H  TEST NEXT BIT
0160' C9  RET  PRESERVE BC
0163' CD 0333'  SBYTE: PUSH B  GET A CONVERTED ASCII CHAR.
0166' 07  RLC  MOVE IT TO HIGH NIBBLE
0167' 07  RLC  SAVE IT
0168' 07  RLC  GET OTHER HALF
0169' 07  RLC  MADE WHILE
016A' 4F  MOV C,A  SAVE AGAIN IN C
016B' CD 0333'  CALL RIBBLE
016C' B1  ORA C  UPDATE CHECKSUM
016F' 4F  MOV C,A  NEW CHECKSUM
0170' 82  ADD D  CONVERTED BYTE
0171' 57  MOV D,A  WRITE TO MEMORY
0172' 79  MOV A,C  ADVANCE POINTER
0173' C1  POP B  COUNT DOWN
0174' C9  RET
0175' DD7700  SIORE: MOV O(X),A
0176' DD23  INX X
0177' 1D  DCR E  COUNT DOWN
0178' C9  RET

; THIS ROUTINE ALLOWS BOTH INSPECTION & 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.
017C' FE53  SUBS:  CPI  'S'  $SEE IF 'SUBSTITUTE'
017E' 202E  JRNZ WRITE
0180' CD 0296'  CALL EXPR1  $GET STARTING ADDR.
0183' E1  POP H
0184' 7E  MOV A,M
0185' CD 02E3'  CALL LBYTE  $DISPLAY THE BYTE
0188' CD 0360'  CALL COPCK  $MODIFY?
018B' D8  RC  $NO, ALL DONE
018C' 2814  JRZ  'S1'  $DON'T MODIFY
018E' FE5F  CPI 'L'  $BACKUP?
0190' 2819  JRZ  'S2'
0192' E5  PUSH H  $SAVE POINTER
0193' 0E01  MVI C,1
0195' 21 0000  LXI H,0
019B' CD 029E'  CALL E+1  $GET NEW VALUE
019F' FE0D  CPI CR  $VALUE IN E
019E' 73  MOV M,E  $MODIFY
01A3' 7D  MOV A,L  $SEE IF TIME TO CRLF
01A4' E607  ANI 7
01A6' CC 021A'  CZ LFADR  $TIME TO CRLF
01AF' 18D9  JMPR 'S0'
01AC' 2B  DCX H  $DECREMENT POINTER
01AD' 18E5  JMPR 'S3'  $AND PRINT DATA THERE.

; ; THIS ROUTINE DUMPS MEMORY IN THE STANDARD ; INTEL HEX-FILE FORMA1. A START & END ; PARAMETER IS REQUIRED. AT THE CONCLUSION ; OF THE DUMP, AN "END OF FILE" SHOULD BE ; GENERATED WITH THE "E" COMMAND.

01AE' FE57  WRITE:  CPI 'L'  $SEE IF 'WRITE' COMMAND
01B0' 2061  JRNZ SIZE
01B2' CD 0273'  CALL EXLF  $GET TWO PARAMETERS
01B5' CD 0374'  CALL CI  $PAUSE FOR PUNCH-ON
01BB' CD 022C'  CALL PE0L  $CRLF TO PUNCH
01BB' 01 003A  LXI B,'$'  $START-OF-FILE CUE
01BE' CD 0233'  CALL TO  $PUNCH IT
01C1' D5  PUSH D  $SAVE
01C2' E5  PUSH H  $ POINTERS
01C3' 04  INR B  $CALCULATE FILE LENGTH
01C4' CD 02C3'  CALL HILO
01C7' 3B24  JRC 'W4'  $SHORT FILE
01C9' 3E18  MVI A,24  $24 BYTES PER FILE
01CB' 90  SUB B  $ENOUGH YET?  $NO.
01CC' 20F5  JRNZ 'W1'  $GET START ADDR BACK.
01CE' E1  POP H  $SEND THE BLOCK
01CF' CD 01D5'  CALL 'W2'
01D2' D1  POP D  $RESTORE END OF FILE POINTER
01D3' 18E3  JMPR 'W0'  $KEEP GOING
TDL Z80 RELOCATING ASSEMBLER VERSION 1.2
<Zap Monitor, Version 2.0, Jan. 16 1977>
Copyright 1977 by TECHNICAL DESIGN LABS, INC.

01D5  57   ....W2:  MOV  D,A  ; INITIALIZE CHECKSUM
01D6  78   MOV  A,B  ; FILE LENGTH
01D7  CD 034D  CALL  PBYTE  ; PUNCH IT
01DA  CD 0348  CALL  PADDR  ; PUNCH ADDRESS
01DD  AF  XRA  A  ; FILE TYPE=0
01DE  CD 034D  CALL  PBYTE  ; PUNCH IT
01E1  7E   ....W3:  MOV  A,M  ; GET A DATA BYTE
01E2  CD 034D  CALL  PBYTE  ; PUNCH IT
01E5  23   INX  H  ; POINT TO NEXT BYTE
01E6  10F9  DJNZ  ....W3  ; DECREMENT FILE COUNT
01E8  AF  XRA  A  
01E9  92   SUB  D  ; CALCULATE CHECKSUM
01EA  C3 034D  JMP  PBYTE  ; PUNCH IT, RETURN
01ED  E1   ....W4:  POP  H  ; CLEAR STACK
01EE  D1  POP  D  ; OF POINTERS
01EF  AF  XRA  A  ; SET-UP A
01F0  1BE3  JMPR  ....W2  ; FINISH UP & RETURN

; THIS IS A MESSAGE OUTPUT ROUTINE.
; IT IS USED BY THE SIGN-ON AND CRLF.
; POINTER IS IN HL (WHEN ENTERED AT
; TOML1) AND LENGTH IN B REG.

01F2  21 0029  ; TOML:  LXI  H, MSG
01F5  4E  ; TOML1:  MOV  C,M  ; GET A CHARACTER
01F6  23   INX  H  ; MOVE POINTER
01F7  CD 0222  CALL  CD  ; OUTPUT IT
01FA  10F9  DJNZ  TOML1  ; KEEP GOING TILL B=0
01FC  CD 0282  CALL  CSTS  ; SEE IF AN ABORT REQUEST
01FF  B7  QRA  A  ; WAITING.
0200  C8  RZ  ; NO.

; SEE IF CONTROL-C IS WAITING
; ABORT IF SO.

0201  CD 0374  CALL  CI
0204  E67F  ANI  7FH  ; KILL PARITY BIT
0206  FE03  CPI  3  ; CONTROL-C?
0208  CO  RNZ
0209  C3 001E  ERRX:  JMP  ERROR

; THIS GETS A READER CHARACTER,
; AND COMPARES IT WITH 'D' REG.
; IT ABORTS ON AN 'OUT-OF-DATA'
; CONDITION.

020C  CD 037D  RIFF:  CALL  RI  ; GET READER CHARACTER
020F  38F8  JRC  ERRX  ; ABORT ON CARRY
0211  BA  CMP  D  ; TEST D
0212  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.

0213' FE5A SIZE: CPI 'Z' ;SEE IF 'SIZE' COMMAND
0215' 2026 JRNZ UNLD
0217' CD 0313' CALL MEMSIZ ;GET THE VALUE

; CRLF BEFORE HLSP ROUTINE
021A' CD 0278' LFADR: CALL CRLF
; PRINT THE CURRENT VALUE OF H&L,
; AND A SPACE.
021D' CD 02DE' HLSP: CALL LADR
; PRINT A SPACE ON THE CONSOLE
0220' 0E20 BLK: MVI C,''
; THIS IS THE MAIN CONSOLE
; OUTPUT ROUTINE.
; TELEPRINTER CONFIGURATION
; I/O DRIVER.
0222' DB00 CO: IN TTS
0224' E680 ANI TTYBE
0226' 20FA JRNZ CO
0228' 79 MOV A,C
022A' D301 OUT TIO
022B' C9 RET
; SEND CRLF TO PUNCH DEVICE
022C' 0E00 PEOL: MVI C,CR
022E' CD 0233' CALL PO
0231' 0EOA MVI C,LF
; THIS IS THE 'PUNCH' OUTPUT
; DRIVER. IT IS SET UP FOR THE
; TTY PORTS, BUT MAY BE MODIFIED
; FOR ANOTHER PORT, FOR TRUE
; SEPARATION OF THE CONSOLE
; AND READER/PUNCH DEVICES.
; (I.E. - PORT 6 & 7 FOR CASSETTE, ETC.)

0233' DB00 PO: IN TTS ;STATUS PORT
0235' E680 ANI TTYBE ;TRANSMITTER BUFFER EMPTY?
0237' 20FA  JRNZ  PO  ;IF NOT, LOOP.
0239' 79  MOV  A.C  ;GET CHARACTER TO OUTPUT
023A' D301  OUT  TIO  ;TO DATA PORT
023C' C9  RET  ;DONE

; 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.

; U<A1>, <A2>{CR}
; PUNCHES FROM <A1> THRU <A2>

023D' FE55  UNLD:  CPI  'U'  ;SEE IF 'UNLOAD' COMMAND
023F' 201A  JRNZ  NULLX
0241' CD 0273'  CALL  EXLF  ;GET TWO PARAMETERS
0244' CD 0374'  CALL  CI  ;PAUSE FOR PUNCH-ON (ITY)
0247' CD 02F6'  CALL  LEAD  ;PUNCH LEADER
024A' CD 02F1'  CALL  MARK  ;PUNCH FILE MARKER
024D' 4E  ;U:
024E' CD 0233'  CALL  PO  ;PUNCH IT
0251' CD 02C3'  CALL  HILO  ;SEE IF DONE
0254' 30F7  JRNZ  ..U
0256' CD 02F1'  CALL  MARK  ;PUNCH END FILE MARKER
0259' 1804  JMPR  NULL

; THIS PUNCHES NULLS (LEADER/TRAILER).
; IT RETURNS "QUIET"

025B' FE4E  NULLX:  CPI  'N'  ;SEE IF 'NULL'
025D' 206E  JRNZ  HEXN
025F' CD 02F6'  NULL:  CALL  LEAD  ;PUNCH NULLS
0262' C3 004A'  JMP  SIARO  ;RETURN QUIET

; CONVERT HEX TO ASCII

0265' 0F  CBYTE:  RRC
0266' 0F  RRC
0267' 0F  RRC
0268' 0F  RRC

0269' E60F  CONV:  ANI  OFH  ;LOW NIBBLE ONLY
026B' C690  ADI  9OH
026D' 27  DAA
026E' CE40  ACI  40H
0270' 27  DAA
0271' 4F  MOV  C,A
0272' C9  RET

; GET TWO PARAMETERS, PLACE
; THEM IN DE & HL, AND THEN
; CRLF.

0273' CD 0298'  EXLF: CALL  EXPR
0276' D1       POP  D
0277' 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).

0278' E5       CRLF: PUSH  H ;SAVE HL
0279' C5       PUSH  B ; & BC
027A' 0604     MVI  B,4 ;CRLF LENGTH (SET FOR 2 FILLS)
027C' CD 01F2' CALL  TOM ;SEND CRLF
027F' C1       POP  B
0280' E1       POP  H
0281' C9       RET

; TEST THE CONSOLE'S
; KEYBOARD FOR A KEY-PRESS.
; RETURN TRUE (OFFH IN A REG)
; IF THERE IS A CHARACTER
; WAITING.

0282' DB00     CSTS: IN   TIS
0284' E601     ANI  TTYDA
0286' 3E00     MVI  A,FALSE
0288' C0       RNZ   ;MAY NEED PATCHING***
0289' 2F       CMA   ;IF DIFFERENT I/D USED
028A' C9       RET

; GET THREE PARAMETERS AND
; CRLF.

028B' 0C       EXPR3: INR  C
028C' CD 0298' CALL  EXPR
028F' CD 0278' CALL  CRLF
0292' C1       POP  B
0293' D1       POP  D
0294' E1       POP  H
0295' C9       RET

; GET ONE PARAMETER.
; NO CRLF.

0296' 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 ONE 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.

0298  21 0000  EXPR  LXI  H,O  $INITIALIZE HL TO ZERO
029B  CD 03DC  EX0  CALL  TI  $GET SOMETHING FROM CONSOLE
029E  47  MOV  B,A  $SAVE IT
029F  CD 0338  EX1  CALL  NIBBLE  $CONVERT ASCII TO HEX.
02A2  3B08  JRC  ..EX2  $ILLEGAL CHARACTER DETECTED
02A4  29  DAD  H  $MULTIPLY BY 16
02A5  29  DAD  H
02A6  29  DAD  H
02A7  29  DAD  H
02A8  B5  ORA  L  $OR IN THE SINGLE NIBBLE
02A9  6F  MOV  L,A
02AA  18EF  JMPR  EJO  $GET SOME MORE
02AC  E3  ..EX2  XTHL  $SAVE UP IN STACK
02AD  E5  PUSH  H  $REPLACE THE RETURN
02AE  78  MOV  A,B  $TEST THE DELIMITER
02AF  CD 0368  CALL  QCHK
02B2  3002  JRNC  ..EX3  $DELIMITER ENTERED?
02B4  0D  DCR  C  $CR, SHOULD GO TO ZERO
02B5  C8  RZ  $RETURN IF IT DOES
02B6  C2 001E  ..EX3  JNZ  ERROR  $SOMETHING WRONG
02B9  CD  DCR  C  $DO THIS AGAIN?
02BA  20DC  JRNZ  EXPR  $YES.
02BC  C9  RET  $ELSE RETURN

RANGE TESTING ROUTINES.
CARRY SET INDICATES RANGE EXCEEDED.

02BD  CD 02C3  HILOX  CALL  HILO  $OK
02C0  D0  RNC  $RETURN ONE LEVEL BACK
02C1  D1  POP  D
02C2  C9  RET

02C3  23  HILO  INX  H  $INCREMENT HL
02C4  7C  MOV  A,H  $TEST FOR CROSSING 64K BORDER
02C5  B5  ORA  L
02C6  37  STC  $CARRY SET=STOP
02C7  C8  RZ  $YES, BORDER CROSSED
02C8  7B  MOV  A,E  $NOW, TEST HL VS. DE
02C9  95  SUB  L
02CA  7A  MOV  A,D
02CB  9C  SBB  H
02CC  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>, <Y>

X+Y, X-Y

02CD FE48
02CF C2 039C
02D2 CD 0273
02D5 E5
02D6 19
02D7 CD 021D
02DA E1
02DB B7
02DC ED52

HEXN: CPI H $SEE IF HEX MATH
       JNZ LOAD $SAVE HL FOR LATER
       CALL ELF $GET SUM
       PUSH H $PRINT IT
       DAD D $THIS IS LATER
       HLSP $CLEAR CARRY
       POP H $GET DIFFERENCE & PRINT IT

; PRINT H&L ON CONSOLE

02DE 7C
02DF CD 02E3
02E2 7D
02E3 F5
02E4 CD 0265
02E7 CD 0222
02EA F1
02EB CD 0269
02EE C3 0222

LADR: MOV A,H
       CALL LBYTE
       MOV A,L
       PUSH PSW
       CALL CBYTE
       CALL CO
       POP PSW
       CALL CONV
       JMP CO

; THIS ROUTINE SENDS EIGHT RUBOUTS
; TO THE PUNCH DEVICE.

02F1 01 08FF
02F4 1803

MARK: LXI B,08FF $SET-UP B&C
       JMPR LEO

; THIS ROUTINE SENDS BLANKS TO THE
; PUNCH DEVICE.

02F6 01 4800
02F9 CD 0233
02FC 10FB
02FE C9

LEAD: LXI B,4800H $PRESET FOR SOME NULLS
       CALL PO
       DJNZ LEO
       RET

; THIS ROUTINE RETURNS TO A USER
; PROGRAM THE CURRENT TOP OF
; MEMORY VALUE MINUS WORKSPACE
; AREA USED BY THE MONITOR.

02FF E5
0300 CD 0313
0303 44
0304 3EC0
0306 E1
0307 C9

MEMCK: PUSH H
       CALL MEMSIZ
       MOV B,H
       MVI A,OCOH $LEAVE SOME ROOM FOR STACK
       POP H
       RET
\# WE BEGIN IN THE MIDDLE......

0308' 3E00 BEGIN: MVI A, I ; INITIAL 'I' REG. VALUE
030A' ED47 STAI ; NEEDED IF USING INTERRUPT.
030C' AF XRA A ; CLEAR READER CONTROL
030D' D303 OUT RCP ; PORT.
030F' 31 0034' LXI SP, STACK ; SET UP A FAKE STACK
0312' 06 .BYTE (MVI) ; SKIP OVER PUSH

\# 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.

0313' C5 MEMSZ: PUSH B
0314' 01 0000' LXI B, ZAP ; POINT TO START OF MONITOR
0317' 21 FFFF' LXI H, -1 ; RAM SEARCH STARTING PT. -1
031A' 24 ..MO: INR H ; FIRST FIND R/W MEMORY
031B' 7E MOV A, M
031C' 2F CM A
031D' 77 MOV M, A
031E' BE CMP M
031F' 2F CMA
0320' 77 MOV M, A
0321' 20F7 JRNZ ..MO
0323' 24 ..M1: INR H ; R/W FOUND, NOW FIND END
0324' 7E MOV A, M
0325' 2F CM A
0326' 77 MOV M, A
0327' BE CMP M
0328' 2F CMA
0329' 77 MOV M, A
032A' 2004 JRNZ ..M2
032C' 7C MOV A, H ; TEST FOR MONITOR BORDER
032D' BB CMP B
032E' 20F3 JRNZ ..M1 ; NOT THERE YET
0330' 25 ..M2: DCR H ; BACK UP
0331' C1 POP B
0332' C9 RET ; VALUE IN HL

\# THIS GETS A READER CHARACTER, AND
\# CONVERTS IT FROM ASCII TO HEX.

0333' CD 020C' RIBBLE: CALL RIFF
0336' E67F ANI 7FH
TDL Z80 RELOCATING ASSEMBLER VERSION 1.2
<Zap Monitor, Version 2.0, Jan. 16 1977>
Copyright 1977 by TECHNICAL DESIGN LABS, INC.

0338 D630 NIBBLE: SUI 'O' ;QUALIFY & CONVERT
033A D8 RC 'O'<
033B FE17 CPI 'G'-1';>F?
033D 3F CMC ;PERVERT CARRY
033E D8 RC
033F FEOA CPI 10 ;NMVR?
0341 3F CMC ;PERVERT AGAIN
0342 D0 RNC ;RETURN CLEAN
0343 D607 SUI 'A'-9'-1';ADJUST
0345 FEOA CPI 10 ;FILTER "0" THRU "9"
0347 C9 RET

;SEND H&L VALUE TO PUNCH DEVICE

0348 7C PADDR MOV A,H
0349 CD 034D CALL PBYTE
034C 7D MOV A,L

;PUNCH A SINGLE BYTE

034D F5 PBYTE: PUSH PSW ;NIBBLE AT A TIME
034E CD 0265 CALL CBYTE
0351 CD 0233 CALL P0
0354 F1 POP PSW ;NEXT NIBBLE
0355 F5 PUSH PSW ;SAVE FOR CHECKSUM
0356 CD 0269 CALL CONV
0359 CD 0233 CALL P0
035C F1 POP PSW ;ORIGINAL BYTE HERE
035D 82 ADD D ;ADDED TO CHECKSUM
035E 57 MOV D,A ;UPDATE CHECKSUM
035F C9 RET

0360 0E2D COPCK: MVI C,;-
0362 CD 0222 CALL CO
0365 CD 03DC CALL TI

;TEST FOR DELIMITERS

0368 FE20 QCHK: CPI ;;RETURN ZERO IF DELIMITER
036A C8 RZ
036B FE2C CPI ;;
036D C8 RZ
036E FEOD CPI CR ;RETURN W/CARRY SET IF CR
0370 37 STC
0371 C8 RZ
0372 3F CMC ;ELSE NON-ZERO, NO CARRY
0373 C9 RET

;MAIN CONSOLE INPUT ROUTINE

0374 DB00 CI: IN ITS
0376 E601 ANI ITYDA
0378 20FA JRNZ CI
037A DB01 IN TII
037C C9	RET

$ READER INPUT ROUTINE, WITH
$ TIME-OUT DELAY. INCLUDES
$ PULSING OF HARDWARE PORT
$ TO INDICATE REQUEST FOR
$ READER DATA.

$ THIS MAY BE ALTERED TO ANY
$ I/O PORT CONFIGURATION TO ENABLE
$ SEPARATE READER/PUNCH DEVICE.

037D E5	RI:	PUSH	H
037E 3EFF	MVI	A,OFFH ;MAY BE ALTERED TO SUIT
0380 D303	OUT	RCP ;PULSE READER CONTROL PORT
0382 AF	XRA	A ;CLEAR IT
0383 D303	OUT	RCP
0385 67	MOV	H,A ;CLEAR FOR TIME-OUT TEST
0386 D800	RIO:	IN	TIS ;MAY BE MODIFIED ***
0388 E601	ANI	ITYDA ;BUT ALWAYS USE 'ANI'
038A 280C	JRZ	R12 ;TO CLEAR CARRY
038C C5	PUSH	B
038D 06FF	MVI	B,OFFH ;SHORTEN FOR HIGH-SPEED DEVICE
038E E3	DLO:	XTHL ;WASTE TIME
0390 E3	XTHL ;FOR DELAY
0391 10FC	DJNZ	DLO
0393 C1	POP	B
0394 25	DCR	H
0395 20EF	JRNZ	RIO
0397 37	R11:	STC ;*NOTE: CARRY SET TO INDICATE
0398 DB01	R12:	IN	TI1 ;NO DATA.
039A E1	R1D:	POP	H
039B 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.

039C FE4C	LOAD*	CPI	'L' ;SEE IF 'LOAD' COMMAND
039E 205F	JRNZ	NEXT
03A0 CD 0296	CALL	EXPR1 ;INITIAL LOAD ADDRESS
03A3 E1	POP	H
03A4 CD 0278	CALL	CRLF
03A7 16FF	MVI	D,OFFH ;START-OF-FILE TAG
03A9 0604 ..LO:	MVI	B,4 ;FIND AT LEAST FOUR OFFH'S
03AB CD 020C ..L1: CALL	RIFF
03AE 20F9	JRNZ	..LO
03B0 10F9	DJNZ	..L1
03B2 CD 020C ..L2: CALL	RIFF ;4 FOUND, NOW WAIT FOR NON-OFFH
03B5 28FB	JRZ	..L2
03B7' 77      MOV  M,A  ;FIRST REAL DATA BYTE
03B8' 3E07    MVI  A,BELL  ;TELL TTY
03BA' D301    OUT  TTO
03BC' 23      ..L3:   INX  H
03BD' CD 020C' CALL  RIFF
03C0' 2803    JRZ  ..EL  ;POSSIBLE END OF FILE
03C2' 77      MOV  M,A
03C3' 18F7    JMPR  ..L3
03C5' 0601    ..EL:   MVI  B,1  ;INITIALIZE
03C7' CD 020C' ..ELO: CALL  RIFF
03CA' 2009    JRNZ  ..EL1
03CC' 04      INR  B  ;COUNT QUES
03CD' 3E07    MVI  A,MAX  ;LOOK FOR EOF
03CF' B8      CMP  B  ;FOUND MAX2
03D0' 20F5    JRNZ  ..ELO  ;NOPE
03D2' C3 020E' JMP  LADR  ;YEP, PRINT END ADDR
03D5' 72      ..EL1: MOV  M,D
03D6' 23      INX  H
03D7' 10FC    DJNZ  ..EL1
03D9' 77      MOV  M,A  ;REAL BYTE
03DA' 18E0    JMPR  ..L3

; THIS IS THE INTERNAL KEYBOARD
; HANDLING ROUTINE. IT WILL IGNORE
; RUBOUTS (OFFH) 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 RECEIVED.

03DC' CD 0374' TI:   CALL  CI
03DF' E67F    ANI  7FH  ;KILL PARITY BIT
03E1' 3C      INR  A  ;IGNORE RUBOUTS
03E2' F8      RM
03E3' 3D      DCR  A  ;IGNORE NULLS
03E4' C8      ZR
03E5' FE4E    CPI  'N'  ;IGNORE N'S FOR NULL CMND
03E7' C8      ZR
03E8' FE6E    CPI  'N'
03EA' 2810    JRZ  ..I
03EC' FE0D    CPI  CR  ;IGNORE CR'S
03EE' C8      ZR
03EF' C5      PUSH  B
03F0' 4F      MOV  C,A
03F1' CD 0222' CALL  CO
03F4' 79      MOV  A,C
03F5' C1      POP  B
03F6' FE40    CPI  'A'-1  ;CONVERT TO UPPER CASE
03F8' D8      RC
03F9' FE7B    CPI  'Z'+1
03FB' DO      RNC
TDL Z80 RELOCATING ASSEMBLER VERSION 1.2
<Zap Monitor, Version 2.0, Jan. 16 1977>
Copyright 1977 by TECHNICAL DESIGN LABS, INC.

03FC' E65F .T: ANI 05FH
03FE' C9       RET

03FF' C9       NEXT: RET  *ADDITIONAL COMMANDS
                 *MAYBE TESTED FROM HERE,
                 *AND THE MONITOR EXTENDED
                 *FROM BEYOND THIS POINT.

0400' Z:       *END OF PROGRAM

0000' .END ZAP

+++++ SYMBOL TABLE ++++

AHEAD 0038' BEGIN 0308' BELL 0007  BLK 0220'
CBY TE 0265' CI 0374' CO 0222' CONV 0269'
COP CK 0360' CR 000D  CTRL 0278' CSTS 0282'
DISP 0057' DLO 038F' EOF 006E' ERROR 001E'
ERRX 0209' EXO 029B' EXI 029E' EXLF 0273'
EXPR 0298' EXPR 0296' EXPRESS 028B' FALSE 0000
FIL 0000' FILL 008C' GOTO 009C' HEXN 02CD'
HILO 023' HILUX 023D' HLSP 021D' I 0000
IOSET 0017' LADR 02DE' LBY TE 023' LEO 02F9'
LEAD 02F6' LENGTH 0400' LF 000A  LFADL 021A'
LOAD 039C' LODO 00ED' LOD 012A' LODCB 0152'
LODR 0132' MARK 02F1' MAX 0007  MEMP 02FF'
MEMSZ 0313' MOVE 00C5' MSG 0029' MSGL 00D'
NEXT 03FF' NIBBLE 0338' NULL 025F' NULX 025B'
PADR 0348' PBYTE 034D' PDEL 022C' PO 0233'
QCHK 0368' RCP 0003 READ 0004  RI 037D'
RIO 0386' RII 0397' R12 0398  RIBBLE 0333'
RID 039A' RIFF 020C' RUB 00FF  SBYTE 0162'
SIZE 0213' STACK 0034' START 004A' START 003E'
SIDRE 0175' SUBS 017C' SESC 00A6' SI 03DC'
TOM 01F2' TOM 01F5' TRUE 00FF  TTI 0001
TID 0001' TTS 0000  TTYBE 0080  TTYDA 0001
UNLD 023D' WRITE 01AE' Z 0400' ZAP 0000'

NO PROGRAM ERRORS