```
.TITLE      '     <ZAPPLE 2-K Monitor, QX-10 Version 2.X - MAY 1982>'
.SBTTL      /        Copyright 1978-83 by APPLEZAP CORP./
;     <<< ZAPPLE 2-K MONITOR SYSTEM >>>
;           by
;        * APPLEZAP *
;
;     WRITTEN by Roger W. Amidon
;
VER   == 4
REV   == 5
MOD   == 0
;
.NAME ZAPPLE
.PROGID     ZAPPLE,VER,REV*10+MOD
.ENTRY      ZAPPLE
.IDENT      ZAPPLE
;
BASE  == .
;
.BTYP == 0 ;\"What style boot? (0=2.0 19K, 1=2.0 16k, 2=1.18 16k) "
;
.CPM  == 0  ;NOT FOR CPM
;
INTBUG      == 0  ;RUN W/INTERRUPTS OFF CONDX ASSEMBLY
;
.DEFINE     SEND=[RST   4]
;
      .IFN  .BTYP-2,[
      .EXTERN     XICP
]
      .EXTERN     ICP,PIOS
      .EXTERN     DSIZE,DRIVER
      .INTERN     TRICK
;
USER  = BASE+0F00H
;
;
STOP  == 80H           ;STOP KEY CODE
;
KEYST == 12H
KEYDAT      == KEYST-2
;
INTCM0      == 8
INTCM1      == INTCM0+1
;
INTCS0      == 0CH
INTCS1      == INTCS0+1
;
INTVEC      == BASE+0F20H
;
KBUF  == BASE+0F60H
MAX   == 32       ;KEYBOARD BUFFER SIZE
;
TTYS  = 13H ;\"TTY STATUS PORT? "
```

```
TTYD  = 11H ;\"TTY DATA PORT? "
TTYRDA      = 0   ;\"RDA BIT? "
TTYTBE      = 2   ;\"TBE BIT? "
;
;     +++ ASCII CONTROL CHARACTERS +++
;
BS    ==    08H         ;BACK SPACE
HT    ==    09H         ;TAB
CR    ==    0DH         ;CARRIAGE RETURN
LF    ==    0AH         ;LINE FEED
FF    ==    0CH         ;FORM FEED
BEL   ==    07H         ;BELL
DC1   ==    11H         ;DC CONTROLS
CLR   ==    1AH         ;CLEAR SCREEN
DEL   ==    7FH         ;DELETE
;
;     <I/O CONFIGURATION MASKS>
;
CMSK  = 11111100B ;CONSOLE DEVICE
RMSK  = 11110011B ;STORAGE DEVICE (IN)
PMSK  = 11001111B ;STORAGE DEVICE (OUT)
LMSK  = 00111111B ;LIST DEVICE
;
;
;-CONSOLE CONFIGURATION
CTTY  = 0   ;RS232 PORT
CCRT  = 1   ;C.R.T.
BATCH = 2   ;READER FOR INPUT, LIST FOR OUTPUT
CUSE  = 3   ;USER DEFINED
;
;-STORAGE INPUT CONFIGURATION
RTTY  = 0   ;RS232 SERIAL PORT
RPTR  = 4   ;QX-10 KEYBOARD
RCAS  = 8   ;HIGH-SPEED RDR (EXTERNAL ROUTINE)
RUSER = 0CH ;USER DEFINED
;
;-STORAGE OUTPUT CONFIGURATION
PTTY  = 0   ;RS232 SERIAL PORT
PPTP  = 10H ;CRT SCREEN
PCAS  = 20H ;HIGH-SPEED PUNCH (EXTERNAL ROUTINE)
PUSER = 30H ;USER DEFINED
;
;-LIST DEVICE CONFIGURATION
LTTY  = 0   ;B CHANNEL SERIAL PORT
LCRT  = 40H ;C.R.T. SCREEN
LINE  = 80H ;LINE PRINTER (EXTERNAL ROUTINE)
LUSER = 0C0H      ;USER DEFINED
;
;
RSTRT = 28H       ;RST 5 (LOCATION FOR TRAP)
RSTOP = 0EFH            ;RESTART 5 OPCODE
;
;     VECTORS FOR USER DEFINED ROUTINES
;
```

```
.LOC  USER
;
CILOC:      .BLKB 3     ;CONSOLE INPUT
COLOC:      .BLKB 3     ;CONSOLE OUTPUT
RPTPL:      .BLKB 3     ;HIGH-SPEED READER (*NOT USED)
RULOC:      .BLKB 3     ;USER DEFINED STORAGE (INPUT)
PTPL: .BLKB 3     ;HIGH-SPEED PUNCH (*NOT USED)
PULOC:      .BLKB 3     ;USER DEFINED STORAGE (OUTPUT)
LNLOC:      .BLKB 3     ;LINE PRINTER (*INTERNAL ROUTINE)
LULOC:      .BLKB 3     ;USER DEFINED PRINTER
CSLOC:      .BLKB 3     ;CONSOLE INPUT STATUS ROUTINE
STPVEC:     .BLKB 3     ;STOP KEY VECTOR
;
ZVINIT      == RPTPL          ;PLUGGABLE VECTOR
;
J =.
;
;     PROGRAM CODE BEGINS HERE
;
.LOC  BASE
;
ZAPPLE:     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.
;
      JMP   CI    ;CONSOLE INPUT
      JMP   RI    ;READER INPUT
      JMP   CO    ;CONSOLE OUTPUT
      JMP   PO    ;PUNCH OUTPUT
      JMP   LO    ;LIST OUTPUT
      JMP   CSTS  ;CONSOLE STATUS
IOCHK:      JMP   GETIO ;I/O CHECK
      JMP   IOSET ;I/O SET
      JMP   MEMCK ;MEMORY LIMIT CHECK
;
;
; 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.
;
TRAP: PUSH  PSW   ;SAVE USER'S ACCUMULATOR
      LDAI        ;GET INTERUPT CONDITION
      DI          ;SHUT OFF INTERUPTS
      XTHL        ;HL=USER ACC, SP=USER HL
      PUSH  D     ;PUSH ALL REGISTERS
      PUSH  B
      PUSH  H     ;USER'S ACC.
      PUSH  PSW   ;GET INTERUPT CONDITION
      POP   B     ;IN C REG.
      LDAR        ;GET REFRESH REG.
      ANI   7FH   ;CLEAR BIT 7
      BIT   2,C   ;TEST PARITY FLAG
      JRZ   .+4   ;INTERUPTS WERE OFF
      ORI   80H   ;ELSE SET BIT 7
      STAR        ;AND SAVE IN "R" REG.
      LXI   D,EXIT+100H
      LXI   H,10  ;GO UP 10 BYTES IN STACK
      DAD   SP
      MVI   B,4   ;PICK OFF REG.
      XCHG        ;HL=MONITOR STACK, DE=USER'S+10
..TR1:      DCX   H
      MOV   M,D   ;SAVE IN WORKAREA
      DCX   H
      MOV   M,E
      POP   D
      DJNZ  ..TR1
      POP   B
      DCX   B     ;ADJUST P.C. VALUE
      SPHL        ;SET MONITOR STACK
      LXI   H,TLOCX
      DAD   SP
      CALL  ..TR6 ;TEST FOR A SET TRAP
      INX   H
      INX   H
```

```
      CNZ   ..TR6 ;TEST FOR 2nd TRAP
      JRZ   ..TR2
      INX   B     ;NO TRAPS SET, RE-ADJUST P.C.
..TR2:      LXI   H,LLOCX
      DAD   SP
      MOV   M,E   ;STORE USER H&L
      INX   H
      MOV   M,D
      INX   H
      INX   H
      INX   H
      MOV   M,C   ;AND USER P.C.
      INX   H
      MOV   M,B
      LXI   H,TLOCX
      DAD   SP
      PUSH  B
      LXI   B,200H
..TR3:      MOV   E,M   ;REPLACE BYTES TAKEN FOR TRAP
      MOV   M,C   ;ZERO OUT STORAGE AREA
      INX   H
      MOV   D,M
      MOV   M,C
      INX   H
      MOV   A,E
      ORA   D     ;DO NOTHING IF ZERO
      JRZ   ..TR4
      MOV   A,M
      STAX  D     ;STORE BYTE
..TR4:      INX   H     ;SAME THING
      DJNZ  ..TR3 ; FOR OTHER BREAKPOINT
      EXAF        ;GET ALTERNATE SET OF REG.'S
      EXX
      XTHL        ;AND STORE IN WORKSPACE
      PUSH  D
      PUSH  B
      PUSH  PSW
      PUSH  X
      PUSH  Y
      LDAR        ;GET REFRESH BYTE
      MOV   C,A
      LDAI        ;GET INTERUPT VECTOR BYTE
      MOV   B,A
      PUSH  B     ;SAVE
;
; AT THIS POINT WE MAY ALLOW INTERRUPTS. SEE IF
; THEY WERE ON WHEN WE GOT HERE, AND ALLOW THEM
; TO BE ON AGAIN IF THEY WERE.
;
      BIT   7,C   ;WERE THEY ON?
      JRZ   ..OFF ;NO
      EI          ;ALLOW NOW
..OFF:      MVI   C,'@' ;DISPLAY BREAK ADDRESS.
      CALL  CO
```

```
      CALL  LADR
      JMP   START ;BACK TO START
;
..TR6:      MOV   A,M
      SUB   C     ;LOOK FOR A TRAP/MATCH
      INX   H
      RNZ
      MOV   A,M
      SUB   B
RETURN:     RET
;
%TRAP=.-TRAP
;
; *** NOTE ***
;
; IN THE QX-10, WE HAVE A KNOWN ENVIRONMENT, SO THIS
; ROUTINE HAS BEEN VASTLY CUT DOWN.
;
; THIS ROUTINE RETURNS TO A USER
; PROGRAM THE CURRENT TOP OF
; MEMORY VALUE MINUS WORKSPACE
; AREA USED BY THE MONITOR.
;
MEMCK:      LXI   B,0EFFFH    ;HIGH BYTE
      MOV   A,C
      LXI   H,TBLTBL    ;KEYBOARD CONVERSION TABLES
      LXI   D,RPTTBL    ;REPEAT TABLE
      RET
;
.PAGE
;
;
;     ANNOUNCEMENT OF MONITOR NAME & VERSION
;
MSG:  .BYTE CR,LF
      .ASCII      'Zapple QX-10 V'
      .BYTE VER!"0",".",REV!"0",MOD!"0"
MSGL  = .-MSG
;
;     LET US BEGIN
;
BEGIN:      DI          ;START W/ NO INTERRUPTS
      XRA   A
      MOV   L,A
      MOV   H,A
      OUT   20H   ;DESELECT C-MOS RAM
      INR   A
      OUT   1CH   ;DESELECT ROM
;
      LXI   SP,EXIT+100H
      MVI   B,10  ; (20 OF THEM)
..B2: PUSH  H     ; TO ZERO
      DJNZ  ..B2
      SET   7,L   ;SHOW WE USE INTERRUPTS
```

```
      PUSH  H     ;RFRSH & INT
;
; STACK NOW PROPERLY SET. INITIALIZE OTHER DEVICES.
;
COLD: CALL  INIT  ;ALTERED ONCE RUN
;
; MEANWHILE, BACK AT ZAPPLE....
;
RE$UPX:     MVI   B,MSGL      ;SAY HELLO TO THE FOLKS
      CALL  TOM   ;OUTPUT SIGN-ON MSG
START:      LXI   D,START     ;MAIN 'WORK' LOOP
      PUSH  D     ;SET UP A RETURN TO HERE
      CALL  CRLF
      MVI   C,'>'
      CALL  CO
      LXI   H,TBL ;POINT TO INTERNAL TABLE
STAR0:      CALL  TI    ;GET A CONSOLE CHARACTER
      JRZ   STAR0 ;GET ANOTHER IF ZERO
      CPI   ' '
      JRC   STAR0 ;NO PROMPT ON CONTROLS
      SUI   'A'   ;QUALIFY THE CHARACTER
      RC          ;<A
      CPI   'Z'-'A'+1
      JRNC  STAR0 ;>Z
      ADD   A     ;A*2
      ADD   L
      MOV   L,A   ;POINT TO PLACE ON TABLE
      MOV   A,M
      INX   H
      MOV   H,M
      MOV   L,A
      ANA   H     ;TEST FOR UNIMPLIMENTED
      INR   A     ;COMMAND
      JRZ   ERRX
      PCHL        ;GO EXECUTE COMMAND
;
.PAGE
;
;
;           <COMMAND BRANCH TABLE>
;
TBL:
;
.WORD ASSIGN      ;A - ASSIGN I/O
.WORD OFF   ;B - BOOT UP THE TPM-II SYSTEM
.WORD CMOS  ;C - CMOS CLOCK RAM I/O
.WORD DISP  ;D - DISPLAY MEMORY ON CONS. IN HEX
.WORD OFF   ;E - EJECT PRINTER PAGE
.WORD FILL  ;F - FILL MEMORY WITH A CONSTANT
.WORD GOTO  ;G - GOTO [ADDR]<,>BREAKPOINTS (2)
.WORD HEXN  ;H - HEX MATH. <SUM>,<DIFFERENCE>
.WORD OFF   ;I * USER DEFINED, INSERT VECTOR
.WORD OFF   ;J - NON-DESTRUCTIVE MEMORY TEST
.WORD OFF   ;K * USER DEFINED, INSERT VECTOR
```

```
.WORD LECHO ;L - ECHO CONSOLE TO LINE PRINTER
.WORD MOVE  ;M - MOVE BLOCKS OF MEMORY
.WORD OFF   ;N - UNUSED
.WORD OFF   ;O * USER DEFINED, INSERT VECTOR
.WORD PUTA  ;P - 'PUT' ASCII INTO MEMORY.
.WORD QUERY ;Q - QI(N)=DISP. N; QO(N,V)=OUT N,V
.WORD OFF   ;R - READ DISK SECTORS
.WORD SUBS  ;S - SUBSTITUTE &/OR EXAMINE MEMORY
.WORD TYPE  ;T - TYPE MEMORY IN ASCII
.WORD OFF   ;U - disk Unit select
.WORD VERIFY     ;V - COMPARE MEMORY AGAINST MEMORY
.WORD OFF   ;W - WRITE DISK SECTORS
.WORD XAM   ;X - EXAMINE & MODIFY CPU REGISTERS
.WORD WHERE ;Y - FIND SEQUENCE OF BYTES IN MEM.
.WORD SIZE  ;Z - ADDRESS OF LAST R/W LOCATION
;
OFF   = -1
;
.PAGE
;
;
; THE VALUE OF THE 'IOBYT' REPRESENTS THE
; CURRENT CONFIGURATION.  IT IS STRUCTURED THUSLY:
;
; 000000XX - WHERE XX REPRESENTS THE CURRENT CONSOLE. (Keyboard)
; 0000XX00 - WHERE XX REPRESENTS THE CURRENT READER.  (Playback)
; 00XX0000 - WHERE XX REPRESENTS THE CURRENT PUNCH.   (Record)
; XX000000 - WHERE XX REPRESENTS THE CURRENT LISTER.  (Printer)
;
; WHEN XX = 00, THE DEVICE IS ALWAYS THE
; TELEPRINTER.  WHEN XX = 11, THE DEVICE IS ALWAYS
; USER DEFINED.  SEE OPERATORS MANUAL FOR FURTHER
; DETAILS.
;
ASSIGN:     CALL  TI    ;GET DEVICE NAME
      LXI   H,LTBL-1 ;POINT TO DEVICE TABLE
      LXI   D,4   ;4 DEV.
      CALL  ..A3  ;GET DEVICE
      PUSH  B
..A1: CALL  TI    ;SCAN PAST '='
      SUI   '='
      JRNZ  ..A1
      MOV   E,A   ;CLEAR E
      CALL  TI    ;GET NEW ASSIGNMENT
      CALL  ..A3  ;GET IT
      POP   D     ;E=DEVICE
      MVI   H,3   ;SET UP A MASK
      MOV   L,C   ;L=ASSIGNMENT
      MOV   A,E   ;GET DEVICE
..A2: DCR   A     ;DEVICE IN A
      JM    ASET  ;GOT IT
      DAD   H     ;DOUBLE LEFT SHIFT
      DAD   H     ; MASK & ASSIGNMENT
      JMPR  ..A2
```

```
..A3: LXI   B,400H      ;4 DEVICES TO LOOK FOR
..A4: INX   H     ;POINT TO ASSIGNMENT NAME
      CMP   M     ;LOOK FOR PROPER MATCH
      RZ          ;MATCH FOUND
      DAD   D     ;POINT TO NEXT
      INR   C     ;KEEP TRACK OF ASSIGNMENT NMBR
      DJNZ  ..A4
ERRX: JMP   ERROR ;NO MATCH, ERROR
;
GETIO:      LDA   IOBYTE
      RET
;
ASET: XRA   H     ;INVERT FOR AND'ING
      MOV   H,A   ;SAVE IN H
      CALL  GETIO ;GET PRESENT CONFIGURATION
      ANA   H     ;MODIFY ONLY SELECTED DEVICE
      ORA   L     ;'OR' IN NEW BIT PATTERN
      MOV   C,A   ;NEW CONFIGURATION
;
%A=.-ASSIGN
;
; THIS ALLOWS USER PROGRAMS TO MODIFY
; THE I/O CONFIGURATION DYNAMICALLY
; DURING EXECUTION.
;
IOSET:      MVI   A,32
      DI
      CALL  CMOSO
      STA   IOBYTE
      EI
      RET
;
; THESE ROUTINES ARE SPECIAL FOR THE QX-10.
; THEY ALLOW QUERY OF THE CMOS CLOCK RAM AS IF
; IT WAS A SEPARATE SET OF I/O PORTS.
;
; IT USES THE SAME STYLE & FORMAT OF COMMAND
; AS THE "Q" COMMAND.
;
CMOS: CALL  TI
      CPI   "O"
      JRZ   ..OUT
      CPI   "I"
      JRNZ  ERRX
      CALL  EXPR
      POP   B
      MOV   A,C
      CALL  CMOSI
      MOV   E,A
      JMP   BITS
;
..OUT:      CALL  EXPC
      MOV   C,E
      MOV   A,L
```

```
;
CMOSO:      OUT   3DH
      MOV   A,C   ;NEW VALUE PASSED IN C REG
      OUT   3CH   ; & SAVED IN CLOCK RAM
      RET
;
CMOSI:      OUT   3DH
      IN    3CH
      RET
;
%C=.-CMOS
;
LECHO:      LXI   H,ECHO$
      CALL  TI
      CPI   "E"
      JRZ   ..E
      SUI   "D"
      JRNZ  ERRX
..E:  MOV   M,A
      RET
;
%L=.-LECHO
;
; THIS DISPLAYS THE CONTENTS OF MEMORY IN BASE HEX
; WITH THE STARTING LOCATION ON EACH LINE.(BETWEEN
; THE TWO PARAMETERS GIVEN).
;
DISP: MVI   A,16  ;SET TO DEFAULT
      CALL  EXPC  ;GET DISPLAY RANGE
..D0: CALL  LFADRX      ;CRLF & PRINT ADDR.
      PUSH  H
..D1: CALL  BLK   ;SPACE OVER
      MOV   A,M
      CALL  LBYTE
      CALL  HILO  ;RANGE CHECK
      JRC   ..D2
      DJNZ  ..D1
..D2: POP   H
      CALL  BLK
      CALL  BLK
      LDA   PARAM3
      MOV   B,A   ;RESET COUNTER
      CALL  TYPASC
      RC          ;DONE
      JMPR  ..D0  ;TIME TO CRLF
;
%D=.-DISP
;
%E=.-.
;
; 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.)
;
FILL: CALL  EXP3  ;GET 3 PARAMETERS
..F:  MOV   M,C   ;STORE THE BYTE
      CALL  HILO
      JRNC  ..F
      POP   D     ;RESTORE STACK
      JMP   START ; IN CASE OF ACCIDENTS
;
%F=.-FILL
;
; 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.
;
GOTO: CALL  PCHK  ;GET A POSSIBLE ADDRESS
      JRZ   ..G0  ;DELIMETER ENTERED
      CALL  EXF   ;GET ONE EXPRESSION
      POP   D
      LXI   H,PLOC      ;PLACE ADDRESS IN 'P' LOCATION
      DAD   SP
      MOV   M,D   ;HIGH BYTE
      DCX   H
      MOV   M,E   ;LOW BYTE
..G0: CPI   CR    ;SEE IF LAST CHARACTER WAS CR
      JRZ   ..G4  ;YES, LEAVE
      MVI   D,2   ;TWO BREAKPOINTS MAX
      LXI   H,TLOC      ;POINT TO TRAP STORAGE
      DAD   SP
..G1: PUSH  H     ;SAVE STORAGE POINTER
      CALL  EXPR  ;GET A TRAP ADDRESS
      POP   B     ;TRAP ADDR.
      POP   H     ;STORAGE
      PUSH  PSW   ;SAVE DELIMETER
      MOV   A,B   ;LOOK AT TRAP ADDR
      ORA   C
      JRZ   ..G2  ;DON'T SET A TRAP AT 0
      MOV   M,C   ;SAVE BKPT ADDR
      INX   H
```

```
      MOV   M,B
      INX   H
      LDAX  B     ;PICK UP INST. BYTE
      MOV   M,A   ;SAVE THAT TOO
      INX   H
      MVI   A,(RSTOP)   ;RESTART OPCODE
      STAX  B     ;SOFTWARE INTERUPT
..G2: POP   PSW   ;LOOK AT DELIMITER
      JRC   ..G3
      DCR   D     ;COUNT BKPTS
      JRNZ  ..G1  ;GET ONE MORE
..G3: MVI   A,JMP ;SET UP JMP INSTRUCTION
      STA   RSTRT ; AT RESTART TRAP LOC.
      LXI   H,TRAP      ; TO MONITOR VECTOR
      SHLD  RSTRT+1
..G4: CALL  CRLF
      POP   D     ;CLEAR SYSTEM RETURN
      POP   D     ;GET 'R' REG.
      PUSH  D     ;RE-ADJUST STACK
      BIT   7,E   ;SEE IF 'EI' OR 'DI'
      LXI   H,INTLOC
      DAD   SP    ;POINT TO OPCODE
      MVI   M,(DI)      ;INITIALIZE TO DI
      JRZ   ..G5
      MVI   M,(EI)      ;ENABLE INTERUPTS
..G5: LXI   H,16H ;FIND 'EXIT' ROUTINE
      DAD   SP    ;UP IN STACK
      PCHL        ;GO SOMEPLACE
;
%G=.-GOTO
      .IFN  0,[
;
; 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
;
TEST: CALL  EXPC  ;GET TWO PARAMS
..T1: MOV   A,M   ;READ A BYTE
      MOV   B,A   ;SAVE IN B REG.
      CMA
      MOV   M,A   ;READ/COMPLIMENT/WRITE
      XRA   M     ; & COMPARE
      MOV   M,B   ;REPLACE BYTE
      JRZ   ..T2  ;SKIP IF ZERO (OK)
      PUSH  D     ;SAVE END POINTER
      MOV   E,A   ;SET-UP TO DISPLAY
      CALL  HLSP  ;PRINT BAD ADDR
```

```
      CALL  BITS  ;PRINT BAD BIT LOC.
      POP   D     ;RESTORE DE
..T2: CALL  HILOX ;RANGE TEST
      JMPR  ..T1
;
%J=.-TEST
;
]
;
; 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>
;
MOVE: CALL  EXP3  ;GET 3 PARAMETERS
..M:  MOV   A,M   ;PICK UP
      STAX  B     ;PUT DOWN
      INX   B     ;MOVE UP
      CALL  HILOX ;CHECK IF DONE
      JMPR  ..M
;
%M=.-MOVE
;
; THIS ALLOWS ENTERING OF ASCII TEXT INTO MEMORY
; FROM THE CONSOLE DEVICE. THE PARITY BIT IS CLEARED,
; AND ALL WILL BE STORED EXCEPT A RUB-OUT (7FH),
; WHICH DELETES THE PREVIOUS CHARACTER, AND A
; CONTROL-D, WHICH TERMINATES THE SEQUENCE. THE
; NEXT ADDRESS WHICH WOULD HAVE BEEN WRITTEN INTO
; IS THEN DISPLAYED ON THE CONSOLE, AND CONTROL
; IS RETURNED TO ZAPPLE.
;
PUTA: CALL  EXPR  ;GET THE STARTING ADDRESS
      CALL  CRLF
      POP   H
..P1: CALL  CI    ;GET A CHARACTER
      CPI   04    ;CONTROL-D (EOT)
      JZ    LFADR ;STOP & PRINT ADDRESS
      CPI   7FH   ;ERASE MISTAKE?
      JRZ   ..P3  ; YES.
      MOV   M,A   ;ELSE STORE IT
      MOV   C,A
      INX   H     ;POINT TO NEXT MEMORY LOCATION
..P2: CALL  CO    ;ECHO IT ON CONSOLE
      JMPR  ..P1  ;AND CONTINUE
..P3: DCX   H     ;BACK UP POINTER
      MOV   C,M   ;PICK UP OLD CHAR.
      JMPR  ..P2  ;AND DISPLAY IT
;
%P=.-PUTA
;
```

```
%R=.-.
;
%W=.-.
;
; 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
; 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.
;
SUBS: CALL  EXPR  ;GET STARTING ADDR.
      POP   H
      RC
..S0: MOV   A,M
      CALL  LBYTE ;DISPLAY THE BYTE
      CALL  COPCK ;MODIFY?
      RC          ; NO, ALL DONE
      JRZ   ..S1  ;DON'T MODIFY
      CPI   '_'   ;BACKUP?
      JRZ   ..S3
      PUSH  H     ;SAVE POINTER
      CALL  EXF   ;GET NEW VALUE
      POP   D     ;VALUE IN E
      POP   H
      MOV   M,E   ;MODIFY
      RC          ;DONE
..S1: INX   H
..S2: MOV   A,L   ;SEE IF TIME TO CRLF
      ANI   7
      CZ    LFADR ;TIME TO CRLF
      JMPR  ..S0
..S3: DCX   H     ;DECREMENT POINTER
      JMPR  ..S2  ;AND PRINT DATA THERE.
;
%S=.-SUBS
;
; 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.
;
TYPE: MVI   A,64  ;SET DEFAULT
      CALL  EXPC  ;GET RANGE
```

```
..T0: CALL  LFADRX      ;DISPLAY ADDRESS
      CALL  TYPASC
      RC          ;DONE
      JMPR  ..T0  ;MORE TO GO
;
TYPASC:     MOV   A,M
      ANI   7FH   ;KILL PARITY BIT
      CPI   ' '   ;RANGE TEST
      JRNC  ..T0  ;=>SPACE
      MVI   A,'.' ;REPLACE NON-PRINTING
..T0: MOV   C,A   ;SEND IT
      CALL  CO
      CALL  HILOX ;MORE TO GO?
      DJNZ  TYPASC      ;SEE IF TIME TO CRLF
      RET         ; YES.
;
%T=.-TYPE
;
; 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>
;
VERIFY:     CALL  EXP3  ;GET 3 PARAMETERS
..V0: LDAX  B
      PUSH  B     ;SAVE POINTER
      MOV   B,M   ;GET MEMORY
      CMP   B     ;MATCH?
      CNZ   CERR  ;DISPLAY ERRORS
..V1: POP   B     ;RESTORE ADDR
      INX   B
      CALL  HILOX
      JMPR  ..V0
;
%V=.-VERIFY
;
;     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]
;
XAM:  CALL  PCHK
      LXI   H,ACTBL
      JRC   ..X6  ;FULL REG. DISPLAY
      CPI   "'"   ;SEE IF PRIMES WANTED
      JRNZ  ..X0
      LXI   H,PRMTB
      CALL  PCHK
      JRC   ..X6  ;FULL REG. DISPLAY
..X0: CMP   M     ;TEST FOR REGISTER NAME
      JRZ   ..X1
      BIT   7,M   ;SEE IF END OF TABLE
      JNZ   ERROR
      INX   H
      INX   H
      JMPR  ..X0
..X1: CALL  BLK
..X2: CALL  ..X8  ;PRINT REGISTER VALUE
..X3: CALL  COPCK ;MODIFY?
      JRZ   ..X5  ;SKIP TO NEXT REG.
      PUSH  H
      PUSH  B
      CALL  EXF   ;GET NEW VALUE
      POP   H
      POP   B     ;OLD B
      PUSH  PSW   ;SAVE DELIMITER
      MOV   A,L
      STAX  D
      BIT   7,B   ;SEE IF 8 BIT OR 16 BIT REG.
      JRZ   ..X4  ;8 BIT
      INX   D
      MOV   A,H   ;HIGH BYTE OF 16 BIT REG.
      STAX  D
..X4: POP   PSW   ;GET DELIMITER
      POP   H     ;RESTORE TABLE POINTER
..X5: RC          ;CR ENTERED, ALL DONE
      BIT   7,M   ;SEE IF END OF TABLE
      RNZ         ;RETURN IF SO
      JMPR  ..X2
..X6: CALL  CRLF
..X7: CALL  BLK
      MOV   C,M
      CALL  CO
      MVI   C,'='
      CALL  CO
      CALL  ..X8  ;PRINT REG. VALUE
      BIT   7,M   ;END OF TABLE?
      JRZ   ..X7  ; NO
      BIT   6,M   ;SEE IF EI/DI
      RNZ         ;ALL DONE
      DCX   D     ;GET "R" REG. VALUE
      LDAX  D
      RAL         ;TEST BIT 7
```

```
      RNC         ;INTERUPTS DISABLED
      MVI   C,'*' ;ELSE ENABLED
      JMPR  CO    ;PRINT '*' AND RETURN
..X8: INX   H     ;POINT TO REG. DISPLACEMENT
      MOV   A,M   ;GET IT
      INX   H     ;POINT TO NEXT IN TABLE
      XCHG        ;SAVE IN DE
      MOV   B,A   ;SAVE FOR FLAGS
      ANI   3FH   ;CLEAN UP FOR OFFSET
      MOV   L,A
      MVI   H,0
      DAD   SP
      INX   H     ;ADJUST FOR THE
      INX   H     ; RETURN ON STACK
      BIT   6,B   ;TEST FOR "M"
      JRZ   ..X9  ; NO
      MOV   A,M   ;GET "M" POINTER
      DCX   H
      MOV   L,M
      MOV   H,A   ;GOT IT
..X9: MOV   A,M   ;GET A BYTE
      CALL  LBYTE ;PRINT REG. VALUE
      XCHG        ;RESTORE TABLE POINTER
      BIT   7,B   ;SINGLE OR DOUBLE?
      RZ          ;SINGLE
      DCX   D
      LDAX  D
      JMP   LBYTE ;PRINT IT & RETURN
;
%X=.-XAM
;
; 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.
;
; "Y'IS THERE..."
;
WHERE:      LXI   H,0   ;COUNT SEARCH BYTES
      MOV   C,L   ;IN C
      DAD   SP    ;GET CURRENT SP
      DCX   H     ;-1
      XCHG        ;SAVE IN DE
..Y0: CALL  EXPR  ;GET A MATCH BYTE
      POP   H     ;IN L
      MOV   H,L   ;STICK IN HIGH BYTE
      PUSH  H     ;PUT IT IN STACK
      INX   SP    ;ADJUST STACK
      INR   C     ;COUNT UP
```

```
      JRNC  ..Y0  ;MORE TO GO
      MOV   A,C   ;GET BYTE COUNT IN A
      PUSH  PSW   ;SAVE COUNT IN SP
      PUSH  D     ;MATCH STRING POINTER
      LXI   B,0   ;STARTING ADDRESS
      PUSH  B     ;SAVE THAT POINTER
..Y1: CALL  CRLF
..Y2: POP   H     ;HL=SEARCH ADDR
      POP   D     ;D=SEARCH BYTE POINTER
      LDAX  D     ;GET FIRST MATCH VALUE
      CCIR        ;COMPARE, INCR., & REPEAT
      JPO   ..DONE      ;ODD PARITY=DONE
      POP   PSW   ;RESET COUNT
      PUSH  PSW   ;SAVE IT AGAIN
      PUSH  D     ;SAVE POINTERS
      PUSH  H
..Y3: DCR   A
      JRZ   ..Y5  ;FOUND ALL
..Y4: EXAF        ;SAVE THE COUNT
      DCX   D     ;LOOK AT NEXT MATCH
      LDAX  D
      CMP   M     ;TEST NEXT
      JRNZ  ..Y2  ;NO MATCH
      INX   H     ;BUMP POINTERS
      EXAF        ;GET COUNT AGAIN
      JMPR  ..Y3  ;TEST NEXT MATCH
..Y5: POP   H
      PUSH  H
      DCX   H
      PUSH  B     ;SAVE SEARCH COUNT LIMIT
      CALL  LADR  ;TELL CONSOLE
      POP   B     ;RESTORE
      JMPR  ..Y1  ;DO IT AGAIN
..DONE:     XCHG        ;GET STACK BACK
      INX   H
      SPHL        ;STACK RESTORED
      RET
;
%Y=.-WHERE
;
; THIS ROUTINE WILL RETURN THE
; CURRENT VALUE OF THE HIGHEST
; READ/WRITE MEMORY LOCATION THAT
; IS AVAILABLE ON THE SYSTEM.
;
SIZE: LXI   H,0EFFFH
;
%Z=.-SIZE
;
; ESTABLISH THE THIRD PARAMETER
;
LFADRX:     LDA   PARAM3
      MOV   B,A   ;RESET B=A'
;
```

```
; CRLF BEFORE HLSP ROUTINE
;
LFADR:      CALL  CRLF
;
; PRINT THE CURRENT VALUE OF H&L,
; AND A SPACE.
;
HLSP: CALL  LADR
;
; PRINT A SPACE ON THE CONSOLE
;
BLK:  MVI   C,' '
;
; THIS IS THE MAIN CONSOLE
; OUTPUT ROUTINE.
;
CO:   CALL  CRTX
      LDA   ECHO$
      ORA   A     ;ECHO TO LP SELECTED?
      CNZ   LO
      MOV   A,C
      RET
;
CRTX: CALL  GETIO
      ANI   # CMSK
      JRNZ  CO0
TTYSND:     CALL  TTYOUT           ;SEND THE CHARACTER
;
; WE ARE ADDING A FILL ABILITY HERE FOR
; SLOW TTY TYPE DEVICES.
;
      CPI   FF    ;WAS THIS A FORM FEED?
      JRZ   ..FF  ;YUP
      CPI   LF    ;WAS THIS A LINE FEED?
      RNZ         ;NO, SPLIT
;
; APPEND NULLS TO FF's & LF's
;
..FF: PUSH  B
      MVI   A,37
      CALL  CMOSI
      ORA   A
      JRZ   ..BYE ;NONE NEEDED
      MOV   B,A   ;NUMBER OF NULLS
      MVI   C,0
..LP: CALL  TTYOUT
      DJNZ  ..LP  ;SEND THEM
..BYE:      POP   B
      MOV   A,C
      RET
;
; TELEPRINTER CONFIGURATION
; I/O DRIVER.
;
```

```
TTYOUT:     MVI   A,10H
      OUT   TTYS  ;RESET STATUS FLAGS
      IN    TTYS  ;PORT B STATUS
      BIT   5,A   ;CLEAR TO SEND?
      JRZ   TTYOUT
      BIT   TTYTBE,A
      JRZ   TTYOUT
      MOV   A,C
      OUT   TTYD
      RET
;
      .IFE  1,[
NWPAG:      MVI   C,12  ;FORM FEED TO LIST DEVICE
      JMPR  LO
]
;
CO0:  DCR   A     ;CCRT?
      JZ    CRTOUT
      DCR   A     ;BATCH?
      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".
;
LO:   CALL  GETIO
      ANI   # LMSK
      JRZ   TTYSND
      CPI   LCRT
      JZ    COLOC
      CPI   LINE
      JNZ   LULOC ;EXTERNAL DRIVER
;
;     PARALLEL PORT DRIVER.....
;
      IN    15H
      ANI   18H   ;MAKE SURE OK
      SUI   18H
      RZ          ;NOT THERE?
..WAIT:     IN    15H
      BIT   5,A
      JRNZ  ..WAIT
      MOV   A,C
      OUT   14H   ;SENT.....
;
; NOW, STROBE IT IN
;
      XRA   A
      OUT   17H
      INR   A
```

```
      OUT   17H
      MOV   A,C
      RET
;
; PUNCH OUTPUT DRIVER ROUTINE
;
PO:   CALL  GETIO
      ANI   # PMSK
      JRZ   TTYOUT      ;PUNCH=PORT B
      CPI   PPTP  ;VIDEO OUT?
      JRZ   CRTX
      JMP   PULOC ;USER VECTOR
;
; 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>
;
QUERY:      CALL  TI
      CPI   'O'
      JRZ   QU0
      CPI   'I'
      JRNZ  ERROR
      CALL  EXPR
      POP   B
      INP   E
BITS: MVI   B,8   ;DISPLAY 8 BITS
      CALL  BLK
..Q2: SLAR  E
      MVI   A,'0' >1
      ADC   A     ;MAKE "0" OR "1"
      MOV   C,A
      CALL  CO
      DJNZ  ..Q2
      JMPR  CRLF  ;CRLF & RETURN
QU0:  CALL  EXPC
      MOV   C,L   ;L=PORT
      OUTP  E     ;E=DATA
      RET
;
%Q=.-QUERY
;
; 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.
;
TOM:  LXI   H,MSG
TOM1: MOV   C,M   ;GET A CHARACTER
```

```
      INX   H     ;MOVE POINTER
      CALL  CO    ;OUTPUT IT
      DJNZ  TOM1  ;KEEP GOING TILL B=0
;
ABORT:      CALL  CSTS  ;SEE IF AN ABORT REQUEST
      ORA   A     ; IS WAITING.
      RZ          ;NO
..WT: CALL  CI
      CPI   13H   ;CONTROL-S?
      JRZ   ..WT  ;YES, PAUSE
      CPI   5     ;CONTROL-E?
      RNZ
;
; SYSTEM ERROR ROUTINE. THIS
; WILL RESTORE THE SYSTEM AFTER
; A SYSTEM ERROR HAS BEEN TAKEN.
; THE I/O CONFIGURATION IS NOT
; AFFECTED.
;
ERROR:      LXI   SP,EXIT+(100H-16H)
      MVI   C,'*' ;ANNOUNCE ERROR
      CALL  CO
      JMP   START ;BACK TO WORK
;
.PAGE
;
;     <SUBROUTINES>
;
; THIS SUBOUTINE IS USED TO DISPLAY THE
; CURRENT LOCATION OF THE 'M' REGISTER POINTER (HL),
; THE VALUE AT THAT LOCATION (IN B), AND THE CONTENTS
; OF THE ACCUMULATOR.
;
CERR: PUSH  PSW   ;SAVE ACC.
      CALL  HLSP  ;DISPLAY H&L
      MOV   A,B
      CALL  LBYTE ;PRINT 'M'
      CALL  BLK   ;SPACE OVER
      POP   PSW
      CALL  LBYTE ;PRINT ACC.
;
; CONSOLE CARRIAGE RETURN &
; LINE FEED ROUTINE.
;
CRLF: PUSH  H     ;SAVE HL
      PUSH  B     ;AND BC
      MVI   B,2   ;CRLF LENGTH
      CALL  TOM   ;SEND CRLF
      POP   B     ;RESTORE THINGS
      POP   H
      RET
;
COEXLF:     CALL  CO
;
```

```
EXLF: CALL  EXPR
      POP   B     ;RETURN RESULT
      JMPR  CRLF
;
; 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.
;
EXPR: CALL  TI    ;GET SOMETHING FROM CONSOLE
EXF:  LXI   H,0   ;INITIALIZE HL TO ZERO
..EX1:      MOV   B,A   ;SAVE IT
      CALL  NIBBLE      ;CONVERT ASCII TO HEX.
      JRC   ..EX2 ;ILLEGAL CHARACTER DETECTED
      DAD   H     ;MULTIPLY BY 16
      DAD   H
      DAD   H
      DAD   H
      ORA   L     ;OR IN THE SINGLE NIBBLE
      MOV   L,A
      CALL  TI
      JMPR  ..EX1 ;GET SOME MORE
..EX2:      XTHL        ;SAVE UP IN STACK
      PUSH  H     ;REPLACE THE RETURN
      MOV   A,B   ;TEST THE DELIMITER
      CALL  QCHK
      JRNZ  ERROR ;SOMETHING WRONG
      RET         ;ELSE RETURN
;
; GET THREE PARAMETERS.
; THE THIRD ONE IS OPTIONAL.
;
; AND CRLF.
;
EXPC: STA   PARAM3      ;SAVE ANY DEFAULT 3rd PARAM
      CALL  EXPR  ;GET 1st. PARAMETER
ERRC: JRC   ERROR ;CR TOO SOON
      CALL  EXPR
      POP   D     ;GET 2nd.
      POP   H     ;GET 1st.
      JRC   ..P   ;CARRY SET=2 PARAMETERS
      PUSH  H     ;SAVE 1st.
      CALL  EXPR  ;GET ONE MORE
      POP   B     ;GET 3rd.
      POP   H     ;GET 1st.
      MOV   A,C   ;TEST 3rd. LSB
      ORA   A     ;ZERO?
      JRZ   ..P   ;USE DEFAULT
      STA   PARAM3      ;UPDATE DEFAULT
      ORA   A     ;CARRY CLEAR=3 PARAMETERS
..P:  PUSH  PSW   ;SAVE FLAGS
```

```
      CALL  CRLF  ;DO CRLF
      POP   PSW
      RET
;
; GET 3 PARAMETERS ONLY
; AND CRLF.
;
EXP3: CALL  EXPC  ;GET 3
      JRC   ERRC  ;I SAID 3
      RET
;
; CONVERT HEX TO ASCII
;
CONV: ANI   0FH   ;LOW NIBBLE ONLY
      ADI   90H
      DAA
      ACI   40H
      DAA
      MOV   C,A
      RET
;
; TEST THE CURRENT CONSOLE'S
; KEYBOARD FOR A KEY-PRESS.
; RETURN TRUE (0FFH IN A REG)
; IF THERE IS A CHARACTER
; WAITING FROM THE INPUT DEVICE.
;
CSTS: CALL  GETIO
      ANI   # CMSK
      JRNZ  ..CS0
; GET STATUS OF "B" PORT (TTY)
..TTY:      IN    TTYS  ;PORT B STATUS
      ANI   01    ;RDA
      RZ
      JMPR  RETFF
;
..CS0:      DCR   A     ;CCRT
      JRZ   KEYSTS
..CS2:      DCR   A     ;BATCH
      JRZ   ..TTY ;USUALLY 0, WE NEED FOR "BYE"
      JMP   CSLOC ;USED DEFINED VECTOR
;
;
; GET STATUS OF "A" PORT (CRT KBD)
;
KEYSTS:
      .IFN  INTBUG,[
      LDAI        ;SEE IF INTERRUPTS ARE ON
      JPO   KOFF  ;NO, DO OLD FASHIONED WAY
][
      EI          ;INSURE INTERRUPTS ON
]
      LDA   KBUF
      ORA   A
```

```
      RZ
      DI
;
; HERE WE ARE TRYING TO HANDLE THE ^STOP STUFF
;
      PUSH  H
      LXI   H,KEYST$
      BIT   6,M   ;BLANKED NOW?
      JRZ   ..NOBL      ;NO
      RES   6,M   ;SHOW UNBLANKED
      MVI   A,0FH ;UNBLANK COMMAND
      JMPR  ..DOBL
;
..NOBL:     LDA   KBUF+1      ;GET THE WAITING KEY
      CPI   STOP  ;IS IT A STOP?
      JRNZ  ..NSTP      ;NO
      BIT   1,M   ;CONTROL-STOP?
      JRZ   ..NSTP      ;NO
;
; AT THIS POINT, ^STOP WAITING. BLANK THE SCREEN
;
      SET   6,M   ;SHOW BLANKED SCREEN
      MVI   A,0CH ;BLANK COMMAND
..DOBL:     POP   H
      CALL  ..BLNK
      JMPR  KEYSTS      ;DO IT AGAIN
;
..BLNK:     PUSH  PSW
..WT: IN    38H
      ANI   20H
      JRZ   ..WT
      POP   PSW
      OUT   39H
      JMP   KYSUCK      ;REMOVE THIS KEY
;
..NSTP:     POP   H
      EI
;
RETFF:      ORI   -1    ;SOMETHING
      RET
;
      .IFN  INTBUG,[
KOFF: LDA   KCSTS$      ;IF CHARACTER HAS COME IN
      ORA   A
      RNZ
      IN    KEYST
      ANI   01    ;ANYTHING WAITING?
      CNZ   GTREAL      ;TEST IT
      JRZ   ..OFF ;NOT REAL
      STA   KCHR$ ;SAVE IT HERE
      SUI   STOP  ;IS THIS THE STOP KEY?
      MVI   A,-1
      STA   KCSTS$
      JZ    STPVEC
```

```
      RET
;
..OFF:      XRA   A     ;INSURE ZERO
      RET
]
;
; RANGE TESTING ROUTINES.
; CARRY SET INDICATES RANGE EXCEEDED.
;
HILOX:      CALL  HILO
      RNC         ;OK
      POP   D     ;RETURN ONE LEVEL BACK
      RET
;
HILO: INX   H     ;INCREMENT HL
      MOV   A,H   ;TEST FOR CROSSING 64K BORDER
      ORA   L
      STC         ;CARRY SET=STOP
      RZ          ;YES, BORDER CROSSED
      MOV   A,E   ;NOW, TEST HL VS. DE
      SUB   L
      MOV   A,D
      SBB   H
      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
;
HEXN: CALL  EXPC
      PUSH  H     ;SAVE HL FOR LATER
      DAD   D     ;GET SUM
      CALL  HLSP  ;PRINT IT
      POP   H     ;THIS IS LATER
      ORA   A     ;CLEAR CARRY
      DSBC  D     ;GET DIFFERENCE & PRINT IT
;
%H=.-HEXN
;
; PRINT H&L ON CONSOLE
;
LADR: MOV   A,H
      CALL  LBYTE
      MOV   A,L
LBYTE:      PUSH  PSW
      RRC
      RRC
```

```
      RRC
      RRC
      CALL  ..2
      POP   PSW
..2:  CALL  CONV
      JMP   CO
;
; THIS WILL CONVERT AN ASCII NUMBER
; IN "A" TO A HEX NIBBLE. IF THE CHAR
; IS NOT WITHIN HEX RANGE, IT RETURNS
; WITH THE CARRY SET.
;
NIBBLE:     SUI   '0'   ;QUALIFY & CONVERT
      RC          ;<0
      CPI   'G'-'0'     ;>F?
      CMC         ;PERVERT CARRY
      RC
      CPI   10    ;NMBR?
      CMC         ;PERVERT AGAIN
      RNC         ;RETURN CLEAN
      SUI 'A'-'9'-1     ;ADJUST
      CPI   0AH   ;FILTER ":" THRU "@"
      RET
;
COPCK:      MVI   C,'-'
      CALL  CO
;
PCHK: CALL  TI
;
; TEST FOR DELIMITERS
;
QCHK: CPI   ' '   ;RETURN ZERO IF DELIMITER
      RZ
      CPI   ','
      RZ
      CPI   CR    ;RETURN W/CARRY SET IF CR
      STC
      RZ
      CMC         ;ELSE NON-ZERO, NO CARRY
      RET
;
%U=.-.
;
%N=.-.
;
; MAIN CONSOLE INPUT ROUTINE
;
CI:   CALL  GETIO
      ANI   # CMSK
      JRNZ  CI1
;
; RS-232C ROUTINE
;
TTYIN:      IN    TTYS  ;PORT B STATUS
```

```
      BIT   TTYRDA,A
      JRZ   TTYIN
      IN    TTYD
      RET
;
CI1:  DCR   A     ;CONSOLE=CRT?
      JRZ   CRTIN
CI2:  DCR   A     ;BATCH?
      JNZ   CILOC ;NO, MUST BE USER DEFINED
;
RI:   CALL  GETIO
      ANI   #RMSK
      JRZ   TTYIN ;SERIAL CHANNEL
      CPI   RPTR  ;CRT CHANNEL?
      JNZ   RULOC ;USER VECTOR
;
; C.R.T. INPUT ROUTINE
;
CRTIN:
      .IFN  INTBUG,[
      LDAI        ;SEE IF INTERRUPTS ARE ON
      JPO   ..OFF ;NO, DO OLD FASHIONED WAY
][
      EI          ;INSURE INTERRUPTS ON
]
..WT: CALL  KEYSTS
      ORA   A     ;IS THERE ANYTHING?
      JRZ   ..WT  ;WAIT FOR IT
;
      DI          ;SHUT OFF FOR NOW
;
KYSUCK:     PUSH  H
      PUSH  D
      PUSH  B
      LXI   H,KBUF
      MOV   A,M   ;GET COUNT
      ANI   1FH   ;SAFETY MEASURE
      ADD   A     ;*2
      MOV   C,A   ;COUNT IN A
      MVI   B,0
      DCR   M     ;ONE LESS
      INX   H
      MOV   D,H
      MOV   E,L   ;FOR MOVE
      MOV   A,M   ;GET ACTUAL ASCII CHARACTER
      INX   H
      EXAF        ;STICK FLAGS INTO A'
      MOV   A,M
      EXAF
      INX   H
      LDIR
      POP   B
      POP   D
      POP   H
```

```
      EI          ;ALLOW INTERRUPTS AGAIN
      RET
;
      .IFN  INTBUG,[
; IN CASE INTERRUPTS WENT OFF
;
..OFF:      PUSH  H
      LXI   H,KCSTS$
      MOV   A,M
      MVI   M,0
      ORA   A
      INX   H
      MOV   A,M
      POP   H
      RNZ
      IN    KEYST
      ANI   01    ;KEYBOARD CHARACTER?
;
; HERE, WE GO TO ..OFF BECAUSE THERE IS A REMOTE
; POSSIBILITY THAT THE "LDAI" INSTRUCTION LIED.
;
      JRZ   ..OFF ;NO, WAIT
      CALL  GTREAL      ;GET A REAL CHARACTER
      JRZ   ..OFF
      CPI   STOP  ;IS THIS THE STOP KEY?
      RNZ
      JMP   STPVEC
]
;
; GET REAL WILL TEST THE KEY PRESS FOR A
; REAL ASCII OR A LOCAL FUNCTION (CTL, SHFT, ETC)
; AND RETURN WITH THE ZERO FLAG INDICATING THE
; VALIDITY OF THE CHARACTER (AND CHARACTER IN "A")
;
; IF ZERO FLAG SET (TRUE), THE CHARACTER IS FALSE.
; IF ZERO FLAG CLR (NON ZERO), CHARACTER IS REAL.
;
GTREAL:     IN    KEYST ;TEST FOR DATA AVAIL
      ANI   1
      RZ          ;NOTHING TO DO
      IN    KEYDAT      ;GET IT
      CALL  CHRCVT      ;DO CONVERSION TO ASCII
      ORA   A     ;ILLEGAL CHAR?
      RNZ         ;NO, REAL
      LDA   CHRIN$      ;GET IT AGAIN
      SUI   61H   ;TRUE NULL..?..
      JRZ   ..NULL
      XRA   A     ;MAKE ZERO, ILLEGAL CHAR
      RET
;
..NULL:     CMP   H     ;MAKE FLAGS NON-ZERO
      RET         ;BUT RETURN THE NULL
;
; This routine is the interupt driven console
```

```
; keyboard routine. We use 8 levels of stack. We
; must preserve the stack.
;
KEYSVC:     SSPD  KSTK
      LXI   SP,KSTK
      PUSH  PSW
      PUSH  H
      PUSH  B     ;SAVE CALLER'S REGISTERS
      CALL  GTREAL      ;CHECK IT OUT
      CNZ   PUTKBF      ;IF REAL, PUT IT INTO FIFO
PKBADR      ==    .-2   ;*** THIS IS POINTED TO BY TBLTBL ***
      POP   B
      POP   H     ;RESTORES CALLER'S REG's
      MVI   A,38H ;RESET INTERRUPT
      OUT   KEYST
      MVI   A,20H ;EOI TO 8259
      OUT   8
      JRNZ  ..STOP
      POP   PSW
      LSPD  KSTK
      EI
      RET
;
..STOP:     POP   PSW
      LSPD  KSTK
      JMP   STPVEC
;
; This routine stores the character in the accumulator into
; the keyboard fifo.  If the fifo is full, it is a no-op.
;
PUTKBF:     LXI   H,KBUF
      STA   KCHR$ ;SAVE THE CHARACTER
      CPI   STOP  ;IS THIS THE STOP KEY?
      JRNZ  ..NSTP      ;NO
      LDA   KEYST$      ;SEE IF CONTROL-STOP
      ANI   2
      JRZ   ..FIX ;NO
..NSTP:     MOV   A,M   ;GET CURRENT COUNT
      CPI   MAX-1 ;FULL?
      JRNC  ..FUL ;OH WELL
      INR   M     ;ADD ONE
      INX   H     ;POINT TO FIRST
      ADD   A     ;COUNT*2
      ADD   L     ;INDEX
      MOV   L,A   ;POINT TO SLOT
;     JRNC  ..NC
;     INR   H
..NC: LDA   KCHR$
      MOV   M,A   ;PUT CHARACTER INTO BUFFER
      XRA   A     ;FORCE ZERO
..RNZ:      INX   H
      LDA   KEYST$      ;GET THIS STATUS
      MOV   M,A
      RET
```

```
..FUL:      LXI   H,KBUF+2    ;SET NEXT STATUS
      SET   3,M         ;AS FULL BUFFER
      XRA   A           ;NOT STOP KEY
      RET
;
; WE GET HERE ON THE "STOP" KEY EXCEPT FOR
; ^STOP. "A" WILL BE ZERO WHEN WE GET HERE.
;
..FIX:      INR   A     ;RETURN NON-ZERO
      MOV   M,A   ;COUNT = 1
      INX   H
      MVI   M,STOP
      JMPR  ..RNZ
;
; THIS IS THE INTERNAL KEYBOARD
; HANDLING ROUTINE. IT WILL IGNORE
; RUBOUTS (0FFH) AND BLANKS (00),
; AND IT WILL NOT ECHO CR'S.
; IT CONVERTS LOWER CASE TO UPPER
; CASE FOR THE LOOK-UP OF COMMANDS.
;
; OTHER CHARACTERS ARE ECHOED AS THEY
; ARE RECIEVED.
;
TI:   CALL  CI
      PUSH  H
      LXI   H,IOBYTE
      BIT   0,M   ;TTY?
      POP   H
      JRNZ  ..CRT ;NO
      ANI   7FH   ;KILL PARITY
      RZ
..CRT:      CPI   CR    ;IGNORE CR'S
      RZ
      PUSH  B
      MOV   C,A
      CALL  CO
      MOV   A,C
      POP   B
      CPI   'A'-1 ;CONVERT TO UPPER CASE
      RC
      ANI   05FH
      RET
;
; <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.
;
LTBL:
;
.BYTE 'C'   ;CONSOLE ASSIGNMENTS
```

```
;
.BYTE 'T'   ;CONSOLE=<T>TY (serial RS-232)
.BYTE 'C'   ;CONSOLE=<C>RT (built-in)
.BYTE 'B'   ;CONSOLE=<B>atch
.BYTE 'U'   ;CONSOLE=<U>SER (dynamically defined)
;
;
.BYTE 'R'   ;READER ASSIGNMENTS
;
.BYTE 'T'   ;RTTY SERIAL PORT B
.BYTE 'K'   ;RPTR SERIAL PORT A (keyboard)
.BYTE 'P'   ;RCAS USER
.BYTE 'U'   ;RUSER      USER
;
;
.BYTE 'P'   ;PUNCH ASSIGNMENTS
;
.BYTE 'T'   ;PTTY SERIAL PORT B
.BYTE 'C'   ;PPTP CRT DISPLAY
.BYTE 'P'   ;PCAS USER
.BYTE 'U'   ;PUSER      USER
;
;
.BYTE 'L'   ;LIST ASSIGNMENTS
;
.BYTE 'T'   ;LTTY SERIAL PORT B
.BYTE 'C'   ;LCRT LIST=CRT
.BYTE 'L'   ;LINE PRINTER (PARALLEL)
.BYTE 'U'   ;LUSER      USER
;
; DISPLACEMENTS OF REGISTER
; STORAGE FROM NORMAL STACK
; LOCATION.
;
ENDX:
;
      ALOC  = 15H
      BLOC  = 13H
      CLOC  = 12H
      DLOC  = 11H
      ELOC  = 10H
      FLOC  = 14H
      HLOC  = 30H
      LLOC  = 2FH
      INTLOC      = 2FH
      PLOC  = 34H
      SLOC  = 17H
      TLOC  = 35H
      TLOCX = 25H
      LLOCX = 1FH
;
      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=NORMAL REG. BYTE
;                01XXXXXX=SPECIAL FOR "M" REG.
;                10XXXXXX=WORD
;
ACTBL:           ;NORMAL SET OF REGISTERS (8080)
;          ;PLUS THE INTERUPT REGISTER ("I")
;
      .BYTE 'A',  ALOC  !0
      .BYTE 'B',  BLOC  !0
      .BYTE 'C',  CLOC  !0
      .BYTE 'D',  DLOC  !0
      .BYTE 'E',  ELOC  !0
      .BYTE 'F',  FLOC  !0
      .BYTE 'H',  HLOC  !0
      .BYTE 'L',  LLOC  !0
      .BYTE 'M',  HLOC  !040H
      .BYTE 'P',  PLOC  !080H
      .BYTE 'S',  SLOC  !080H
      .BYTE 'I',  ILOC  !0
;
      .BYTE 80H
;
PRMTB:           ;ADDITIONAL SET OF REGISTERS (Z-80)
;
      .BYTE 'A',  APLOC !0
      .BYTE 'B',  BPLOC !0
      .BYTE 'C',  CPLOC !0
      .BYTE 'D',  DPLOC !0
      .BYTE 'E',  EPLOC !0
      .BYTE 'F',  FPLOC !0
      .BYTE 'H',  HPLOC !0
      .BYTE 'L',  LPLOC !0
      .BYTE 'M',  HPLOC !040H
```

```
      .BYTE 'X',  XLOC  !080H
      .BYTE 'Y',  YLOC  !080H
      .BYTE 'R',  RLOC  !0
      .BYTE 0C0H
;
; This is the keyboard service stack area.
; It appears that the keyboard service uses
; 8 levels of stack, worse case.
;
      .BYTE [16]0AAH         ;STACK AREA
;
KSTK: .WORD ERROR            ;STACK SAVER
;
IOBYTE:     .BYTE 0          ;SYSTEM I/O BYTE
;
ECHO$:      .BYTE 0          ;ECHO CONSOLE TO PRINTER FLAG
;
PARAM3:     .BYTE 0          ;STORAGE FOR OPTIONAL PARAMETER
;
CHRIN$:     .BYTE 0
KCSTS$:     .BYTE 0
KCHR$:      .BYTE 0
;
;
      .INSERT    KEYCNVT.INS
;
      .INSERT    KEYTBL.INS
;
; NOTE THAT LABEL "ZZ" IS END OF "Must have" CODE.
;
     ZZ:
;
.INSERT    VID-DRV.INS
;
;     THE FOLLOWING ARE INITIALIZED ON START-UP
;     AND ONLY USED BY CRT DRIVER.
;
%SPOS:      .WORD 0
%XPOS:      .BYTE 0
%YPOS:      .BYTE 0
;
; THIS IS THE REAL END OF ZAPPLE ONCE WE HAVE BOOTED
;
; WE RESERVE FROM 0FD00H-0FDFFH FOR .SPLR. STUFF
; WE RESERVE FROM 0FE00H-0FEFFH FOR .UTIL. STUFF
;
;     **** NOTE -> WE MUST BE BELOW 0FD00H HERE ****
;
.PAGE
.SBTTL      /INITIALIZATION CODE/
;
; NOTE - MUCH OF THIS MAY EVENTUALLY BE IN THE
;     COLD START ROM.
;
```

```
; -- NOTE --
;     WE ARE USING THE CMOS RAM IN THE CLOCK CHIP
;     TO HOLD THE FOLLOWING VALUES:
;
; 32 = IOBYTE
; 33 = UNUSED
; 34 = KEYBOARD STATUS
; 35 = BAUD RATE FOR EXTERNAL SERIAL PORT
; 36 = VIDEO STATUS
; 37 = NULLS ON TTY DEVICE OUTPUT
; 38 = RESERVED FOR SCHEDULER
; 39 = RESERVED FOR VALDOCS
; 40 = MODEM FILE COUNT KEEPER
; AND OTHERS. SEE CMOSDEF.DOC
;
; - WE ARE USING THE 8 BIT EXTERNAL SWITCH AS FOLLOWS:
;
; BIT 0 - IF 0, WE USE CLOCK RAM #32 FOR CONSOLE
;       IF 1, WE FORCE INTERNAL CONSOLE.
;
; BIT 1 - IF 0, WE AUTO BOOT TPM
;       IF 1, WE JUST SIGN ON ZAPPLE
;
;     **** NOTE -> THE FOLLOWING TABLES MAY
;                  BE MOVED TO BOOT ROM.
;
; THE FOLLOWING TABLES ARE BASED ON USING
; A 16X CLOCK RATE, WITH THE DIVIDE CHAIN RUNNING
; AT 2 Mhz.
;
BTBL: .WORD 1135  ;110 (0)
;
      .WORD 924   ;135 (1)
;
      .WORD 832   ;150 (2)
;
      .WORD 416   ;300 (3)
;
      .WORD 208   ;600 (4)
;
      .WORD 104   ;1200 (5)
;
      .WORD 52    ;2400 (6)
;
      .WORD 26    ;4800 (7)
;
      .WORD 13    ;9600 (8)
;
;
SIOTB:      .BYTE 18H   ;RESET
;
      .BYTE 14H   ;WRITE REG 4.
      .BYTE 44H   ;4 - X16, 1 STOP
;
```

```
      .BYTE 03H   ;WRITE REG 3.
      .BYTE 0C1H  ;8 BITS, NO AUTOENAB, RX ENA
;
      .BYTE 005H  ;WRITE REG 5.
      .BYTE 0EAH  ;DTR, RTS=1, 8 BIT, TXENB
;
      .BYTE 002H  ;WRITE REG 2.
      .BYTE 000H  ;BOTH CHANNELS INTERRUPT
;
      .BYTE 001H  ;WRITE REG 1.
      .BYTE 00H   ;NEVER INTERRUPT
;
SIOTL == .-SIOTB
;
INIT: LXI   H,SIOTB     ;COMMAND TABLE
      MVI   B,SIOTL     ;.......LENGTH
      MVI   C,TTYS      ;.......PORT
      OUTIR             ;.......SET
;
; NOW, SET UP CTC PROPERLY FOR SELECTED BAUD
;
      LXI   H,BTBL
      MVI   A,35  ;FOR BAUD RATE CLUE
      CALL  CMOSI
      ANI   0FH   ;IN CASE OF JUNK
      MOV   C,A   ;SAVE IT
      MVI   A,44  ;TO TEMP
      CALL  CMOSO
      ADD   A     ;TWO BYTES PER ENTRY
      MOV   C,A
      MVI   B,0
      DAD   B
      MVI   A,0B6H      ;MODE 3, CTR 2
      OUT   7
      MOV   A,M   ;LOW BYTE OF BAUD
      OUT   6
      INX   H
      MOV   A,M   ;HIGH BYTE OF BAUD
      OUT   6
;
;     NOW SET UP BELL TIMER
;
      MVI   A,32H ;MODE 1, CTR 0
      OUT   3
      MVI   A,128 ;BELL TIMING
      OUT   0
      XRA   A
      OUT   0
;
; WE ARE NOW ALWAYS CLEARING 2 CLOCK RAM LOCATIONS
;
      MOV   C,A         ;C=0
      MVI   A,48
      CALL  CMOSO
```

```
      MVI   A,49
      CALL  CMOSO
;
; SET THE MOTOR TIMEOUT FOR 15-20 SECONDS.
;
      MVI   A,10  ;TIMER ADDRESS IN CMOS CALENDAR
      MVI   C,29H ;SET IT UP
      CALL  CMOSO
;
; INITIALIZE THE HARDWARE HERE FOR INTERRUPTS
;
; FIRST, INITIALIZE THE INTERRUPT CONTROLLER
;
; SET EDGE, 4 BYTE VECTOR, CASCADE, NEED ICW4
      LXI   H,INTVEC + 15H ;VECTOR LOCATION
      MOV   A,L   ;LOW BYTE OF VECTOR ADDR
      OUT   INTCM0      ;MASTER
; HIGH ORDER INTERRUPT VECTOR ADDRESS
      MOV   A,H
      OUT   INTCM1
; SLAVE INTO INT #7
      MVI   A,80H
      OUT   INTCM1
; NO AUTO EOI
      XRA   A
      OUT   INTCM1
; SLAVE IS SHUT OFF AT THE MOMENT, BUT
; SET IT UP ANYWAY.
      LXI   H,INTVEC+(4*8)+15H ;NEXT SLOT
      MOV   A,L
      OUT   INTCS0
; HIGH ORDER NOW
      MOV   A,H
      OUT   INTCS1
; SLAVE I.D. (USING #7)
      MVI   A,7
      OUT   INTCS1
; NO AUTO EOI
      XRA   A
      OUT   INTCS1
; NOW, ONLY ALLOW KEYBOARD FOR NOW
      MVI   A,#10H
      OUT   INTCM1
;
; SET UP SLAVE AS INACTIVE.
      MVI   A,#0
      OUT   INTCS1
;
; WE ARE NOW INITIALIZED.
;
; GET DEFAULT KEYBOARD CONFIG & I/O BYTE
;
      MVI   A,34
      CALL  CMOSI
```

```
      ANI   00110000B ;JUST CAPSLOCK & SHIFT LOCK
      OUT   3CH
      STA   KEYST$     ;INITIALIZE
;
      MVI   A,32  ;IOBYTE
      CALL  CMOSI
      RES   1,A   ;DON'T ALLOW BATCH/USER
      MOV   C,A
      MVI   A,62  ;SEE IF EVER INIT'ED
      CALL  CMOSI
      CPI   0AAH
      JRNZ  ..INT ;NO, FORCE INTERNAL
      IN    18H   ;GET SWITCHES
      BIT   0,A   ;ALLOW RS-232 CONSOLE?
      JRZ   ..MAN ;YES
..INT:      MOV   A,C
      ANI   0FCH  ;FORCE INTERNAL CONSOLE
      ORI   1
      MOV   C,A
;
; WE CAN NOW ENABLE ANY INTERRUPTS. WE WILL USE
; THE 0FF00H PAGE FOR INTERRUPT HANDLING VECTORS.
;
..MAN:      IM0         ;MODE 0
      CALL  IOSET ;INITIALIZE CURRENT CONFIG
                  ;RETURNS WITH INTERUPTS ENABLED
;
RESULT:               ;DISK SCRATCH AREA
      LXI   H,ZVINIT
      SHLD  COLD+1      ;NEVER COME HERE AGAIN
      CALL  VINIT ;INITIALIZE VIDEO
;
; BOOT TPM FROM SYSTEM DISK
;
BOOT: MVI   A,20H ;ON BANK 1
      OUT   18H

      LXI   H,SENDC
      SHLD  21H
      MVI   A,(JMP)
      STA   20H
;
      .IFN  .BTYP-2,[
      LXI   H,XICP
][
      LXI   H,DRIVER
]
      .IFE  .BTYP,[
      LXI   D,10  ;TRK 0 SEC 10
][
      LXI   D,16  ;TRK 0 SEC 16
]
      MOV   B,E   ;B=FULL TRACK
      CALL  ..NXT ;GET FIRST PART
```

```
;
      INR   D     ;TRK 1 SEC 0
      MOV   E,A   ;E=0
      CALL  DRINIT      ;DETERMINE 48/96tpi
      .IFE  .BTYP,[
      MVI   B,18  ;GET 9k WORTH
]
      .IFE  .BTYP-1,[
      MVI   B,7*4 ;GET SECOND PART
]
      .IFE  .BTYP-2,[
      MVI   A,10H ;BACK TO BANK 0
      OUT   18H
      MVI   B,30  ;GET 30 PAGES
]
      CALL  ..NXT
      .IFE  .BTYP,[
      LXI   B,1024
      DAD   B
      MVI   B,2
]
      .IFE  .BTYP-1,[
      LXI   B,1024+256  ;SPLIT BUFFERS + CLIOS BUFFERS
      DAD   B     ;SKIP DATA AREA AFTER CLIOS
      DCR   B     ;B=4
]
      .IFN  .BTYP-2,[
      CALL  ..NXT ;GET PIOS
]
      JMP   PIOS
;
..NXT:      PUSH  B
      PUSH  D
      PUSH  H
      CALL  DSKDRV
      POP   H
      POP   D
      POP   B
      JRZ   ..GO
      POP   B     ;REMOVE RETURN
      CALL  LBYTE ;SHOW ERROR TYPE
      XCHG
      JMP   LFADR
;
..GO: INR   H           ;BUMP DMA
      .IFE  .BTYP,[
      INR   H           ;TWICE IF 512 BYTES/SEC
]
      INR   E           ;BUMP SECTOR
      DJNZ  ..NXT       ;MORE?
      RET
;
%B=.-BOOT
;
```

```
.INSERT    DRIVERX.INS
;
;
EOA:             ;END OF IT ALL
;
.PAGE
.SBTTL     /FIXED LOCATIONS DEFINED/
;
; MEMORY USAGE HERE:
;
;     0FF00H-0FF1BH    = USER I/O JUMP TABLES.
;     0FF1CH-0FF1EH    = STOP KEY VECTOR
;            0FF1FH    = KEYBOARD STATUS BYTE
;     0FF20H-0FF5FH    = INTERRUPT VECTORS
;     0FF60H-0FF9FH    = KEYBOARD BUFFER
;     0FFA0H-0FFFFH    = MONITOR STACK
;
; WE ARE NOW GOING TO DEFINE & PLACE THE CODE
; TO HANDLE THE USER & INTERUPPT VECTORS AT 0FE00H.
; DURING THE "GETTBL" ROUTINE, THEY WILL MAGICALLY
; APPEAR IN THE 0FF00H AREA.
;
      .LOC  ZAPPLE+0E00H
;
      JMP   CRTIN ;CONSOLE IN w/VDRIVER
..CO: JMP   CRTOUT      ;CONSOLE OUT w/VDRIVER
      JMP   VINIT ;VIDEO INITIALIZATION ROUTINE
      JMP   ERROR ;EXTERNAL RS-232 RI VECTOR
      JMP   ERROR ;UNUSED VECTOR
      JMP   ERROR ;EXTERNAL RS-232 PO VECTOR
      JMP   ERROR ;EXTERNAL RS-232 LO VECTOR
      JMP   ..CO+100H ;EXTERNAL PARALLEL LO VECTOR
      JMP   KEYSTS      ;CONSOLE STATUS CHECK w/VDRIVER
      EI          ;STOP KEY VECTOR
      JMP   RETURN
;
KEYST$     == .+100H
;
.PAGE
.SBTTL     /INTERRUPT HANDLING VECTOR ALLOCATION/
;
; INTERRUPT HANDLING STUFF
; JUST KEYBOARD FOR NOW.
;
      .LOC  INTVEC-100H
;
INT0: JMP   ERROR
      .BYTE 0     ;POWER FAIL INTERRUPT
INT1: JMP   ERROR
      .BYTE 1     ;SOFTWARE TIMER #1
INT2: JMP   ERROR
      .BYTE 2     ;OPTION CARD #1
INT3: JMP   ERROR
      .BYTE 3     ;OPTION CARD #2
```

```
INT4: JMP   KEYSVC
      .BYTE 4     ;QX-10 KEYBOARD / RS-232
INT5: JMP   ERROR
      .BYTE 5     ;7220 INTERRUPT
INT6: JMP   ERROR
      .BYTE 6     ;FDC INTERRUPT
INT7: JMP   ERROR
      .BYTE 7     ;(SLAVE - NOT USED)
;
; SLAVE
;
INT8: JMP   ERROR
      .BYTE 8     ;PARALLEL PORT (PRINTER)
INT9: JMP   ERROR
      .BYTE 9     ;OPTION CARD #3
INT10:      JMP   ERROR
      .BYTE 10    ;CALENDAR CLOCK (ALARM)
INT11:      JMP   ERROR
      .BYTE 11    ;OPTION CARD #4
INT12:      JMP   ERROR
      .BYTE 12    ;OPTION CARD #5
INT13:      JMP   ERROR
      .BYTE 13    ;SOFTWARE TIMER #2
INT14:      JMP   ERROR
      .BYTE 14    ;OPTION CARD #6
INT15:      JMP   ERROR
      .BYTE 15    ;OPTION CARD #7
;
      .LOC  ZAPPLE+0E70H     ;MIDDLE OF KEYBOARD BUFFER
;
      .DATE       ;CURRENT ASSEMBLY TIME/DATE
      .TIME
;
      .LOC  ZAPPLE+0EA0H
;
TRICK == .+100H
;
      LXI   H,0F100H
      LXI   D,0F000H
      LXI   B,0E80H
      LDIR
      JMP   ZAPPLE
;
      .LOC  ZAPPLE+0EDCH
;
; THIS IS A SHORT PROGRAM, EXECUTED
; UPON EXECUTING A "GO" COMMAND. IT
; IS PLACED IN THE STACK WORK AREA.
;
EXIT:             ;EXIT ROUTINE LOADS ALL REGISTERS
      POP   B
      MOV   A,C
      NOP         ; -   STAR
      NOP
```

```
      MOV   A,B
      STAI        ;THIS HARDWARE CAN'T SUPPORT IM2
      POP   Y
      POP   X
      POP   PSW
      POP   B
      POP   D
      POP   H
      EXAF
      EXX
      POP   D
      POP   B
      POP   PSW
      POP   H
      SPHL
HLX   == .+101H
      LXI   H,0
      NOP         ;RESERVED FOR ENABLE INTERUPTS
      JMP   0
;
      .WORD 0     ;STORAGE AREA FOR TRAP DATA
      .BYTE 0
      .WORD 0
      .BYTE 0
;
Z:
;
.END  ZAPPLE
```