```
;       8080.ASM  This is an 8080 (only) monitor program for an Altair/IMSAI system.
;       It is a simple monitor for the S100 8080 CPU board by Josh (aka Crusty OMO)
;       It resided in 1 27C256 PROM at F000H. However the board initilizes with the
;       onboard ROM residing at 0H.  The first thing the code does is relocate itself
;       to RAM at F000H. It then disables the ROM at 0h so RAM from 0 to EFFFH is available
;       for CPM etc.  Note the code has an ORG ath F000H. The first 3 bytes are hard wired
;       for its initial 0H ROM location.
;
;       Note this is a "quick & dirty" rework of the MASTER.Z80 monitor. Currently only the
;       basic monitor commands are implemented. Booting CPM is not yet done.
;
;       Assemble with Digital Research's MAC Assembler
;       Use:- MAC 8080.asm
;
;       The monitor is in two sections. The F0000H-F7FFH is for typical display
;       move memory type functions.  The second portion starts at F0800H and contains
;       a series of CPM BIOS compatible jumps. For compatability with some of my old
;       CPM V1.4 software these locations should not be changed. You can easily build
;       around them.  The second section (after the above BIOS jumps section) contains
;       CPM boot loader code and other more specilized stuff.
;
;       To assemble under windows...
;       Load Altair.EXE
;       do cpm3
;       I:
;       Submit 8080
;       8080.HEX is written back to the same windows as altair.exe is in.
;
;       Programming an EEPROM for the Z80 Board with the VP-290 Programmer
;       Using an Intel D273256 EPROM:-
;       For monitor at F000H-F1FFFH
;       Buffer Address 0000
;       From File address F000H
;       This will put the code (up to 8K) in the 8K EEPROM.
;       It can be seen/edited at 0000H in the ROM

;       Recent History...
;       V1.0    29/6/2013             Initial conversion from my Z80's Master.Z80 code, (V4.8)
;       V1.1    7/29/2014             Relocating version for S100 8080 board
;       V1.2    8/1/2014              Removed the MASTER.Z80 IDE routines
;
;
;
SCROLL  EQU     01H                   ;Set scrool direction UP.
BELL    EQU     07H
SPACE   EQU     20H
TAB     EQU     09H                   ;TAB ACROSS (8 SPACES FOR SD-BOARD)
CR      EQU     0DH
LF      EQU     0AH
FF      EQU     0CH
QUIT    EQU     11H                   ;Turns off any screen enhancements (flashing, underline etc).
NO$ENHANCEMENT EQU 17H                ;Turns off whatever is on
FAST    EQU     10H                   ;High speed scrool
ESC     EQU     1BH
DELETE  EQU     7FH
```

```
BACKS   EQU     08H
CLEAR   EQU     1AH                     ;TO CLEAR SCREEN
RST7    EQU     38H                     ;RST 7 (LOCATION FOR TRAP)
IOBYTE  EQU     0EFH                    ;IOBYTE (SEE BELOW)
NN      EQU     0H                      ;[I] INITIAL VALUE
;

STARTCPM EQU    100H                    ;LOCATION WHERE CPM WILL BE PLACED FOR COMOV BOOT
STARTDOS EQU    100H                    ;LOCATION WHERE MSDOS WILL BE PLACED FOR COMOV BOOT
FFILE$SIZE EQU 9000h/512                ;SIZE OF 5MSDOS20.COM IN 512 BYTE SECTORS
STACK   EQU     0EFF0H                  ;Will hard wire the stack to this location since the board will always have RAM here.
;
;
;IOBYTE = SENSE SWITCHES AT PORT 0EFH
;
; BIT MAP OF PORT 0EFH:- X X X X  X X X X      (11111111=NORMAL CONFIG)
;                        | | | |  | | | |..For Z80/8080 Monitor      0=CONSOLE DATA TO PRINTER ALSO
;                        | | | |  | | |....For 8086 Monitor,  0=Force MSDOS Consol output to CGA/VGA Board instead of Propeller board
;                        | | | |  | |......For 8086 Monitor,  0=Do not initilize extra ROMS
;                        | | | |  |........Unused
;                        | | | |...........Unused
;                        | | |.............For Z80/8080 Monitor      0=ALL Consol I/O via ACIA Serial port on S100Computers Serial-IO Board
;                        | |                                        Remember the serial port is currently set for 19.2K, 1 Stop,NP
;                        | |...............For CPM3,          0=Force reformat of Memory disk upon any CPM3 coMOV re-boot
;                        |.................For CPM3,          0=Write protect Memory disk for CPM3
;
;                                       For 8086 Monitor,       0=Prevent doing a JMPF to 500H after 8086 reset (to CPM86 boot)
;                                                               Normally a test is made to see if the CPM86 Boot is already in RAM at 500H
;                                                               If it is, a 8086 reset will bypass the monitor and go directly there.
;                                                               (see Init: in 8086 Monitor)
;                                       Note if 00xxxxxx,       This will force 8086 hardware diagnostic test. (See code at FFFF0H in the 8086 monitor)
;
;
;-------------- SD SYSTEMS VIDIO BOARD FOR CONSOLE INPUT & OUTPUT
CONSOL$STATUS  EQU     0H
CONSOL$IN      EQU     01H
CONSOL$OUT     EQU     01H

;--------------- PORTS ON 8080 CPU BOARD ITSELF ----------------------------------------

BOARD$PORTS    EQU     60H             ;Note overlap with Lomas Video board and Tarbell casette board!
COM1$BASE      EQU     BOARD$PORTS+7H ;8250 8 registers base
COM2$BASE      EQU     BOARD$PORTS+0FH         ;8250 8 registers base
SD$CARD$SPI    EQU     BOARD$PORTS+10H         ;SD Card shift register
SD$CARD        EQU     BOARD$PORTS+11H         ;LSB Controls selecting the SD Card. 0 selects card and if JP5 1-2 kills ROM
PARALLEL$RD    EQU     BOARD$PORTS+18H         ;8212 Read
PARALLEL$WR    EQU     BOARD$PORTS+18H         ;8212 Write


;-------------- THIS IS MY PORT TO OUTPUT DATA TO HP 4050T LASAR PRINTER  (8PIO Board)

PRINTER$STATUS EQU     5               ;IN, HP PARRELL PORT
PRINTER$OUT    EQU     5               ;OUT
PRINTER$STROBE EQU     4               ;OUT
DIAG$LEDS      EQU     5               ;OUT (Will use this port initially for diagnostic LED display)
```

```
;-------------- S100Computers I/O BOARD PORT ASSIGNMENTS  (A0-AC)

BCTL            EQU     0A0H            ;CHANNEL B CONTROL PORT ASSIGNMENTS OF THE ZILOG SCC CHIP
ACTL            EQU     0A1H            ;CHANNEL A CONTROL
BDTA            EQU     0A2H            ;CHANNEL B DATA
ADTA            EQU     0A3H            ;CHANNEL A DATA

PortA$8255      EQU     0A8H            ;A port of 8255 ;<--- Adjust as necessary
PortB$8255      EQU     0A9H            ;B port of 8255
PortC$8255      EQU     0AAH            ;C Port of 8255
PortCtrl$8255   EQU     0ABH            ;8255 configuration port
AinBout8255cfg EQU      10011000b       ;Set 8255 ports:- A input, B output,

USB$DATA        EQU     0ACH            ;PORT ASSIGNEMENT FOR DLP-USB Controller chip
USB$STATUS      EQU     0AAH            ;Status port for USB port (Port C of 8255, bits 6,7)
USB$RXE         EQU     80H             ;If Bit 7 = 0, data available to recieve by S-100 Computer
USB$TXE         EQU     40H             ;If Bit 6 = 0 data CAN be written for transmission to PC


;-------------- S100Computers MSDOS Support Board PORT ASSIGNMENTS

CMOS$PORT       EQU     70H             ;Base Port for CMOS Chip on MSDOS Support Board
MASTER$PIC$PORT         EQU     20h             ;Hardware port the 8259A (two ports 20H & 21H)

MasterICW1      equ     00010111B       ;EDGE triggered, 4 bytes, single Master,ICW4 needed
MasterICW2      equ     8H              ;Base address for 8259A Int Table (IBM-PC uses 8X4 = 20H)
MasterICW3      equ     0H              ;No slave
MasterICW4      equ     00000011B       ;No special mode, non buffer, Auto EOI, 8086.  ;<<<<,


;--------------- PORTS FOR FOR Z80/WD2793 FDC Board

S100$DATA$A     EQU     10H             ;IN,  S100 Data port to GET data to from FDC Board
S100$DATA$B     EQU     10H             ;OUT, S100 Data port to SEND data to FDC Board
S100$STATUS$A   EQU     11H             ;Status port for A
S100$STATUS$B   EQU     11H             ;Status port for B
RESET$ZFDC$PORT EQU     13H             ;Port to reset ZFDC Z80 CPU.

STATUS$DELAY    EQU     5               ;Time-out for waiting for ZFDC Board handshake signal (~0.5 seconds @ 10MHz)
DIRECTION$BIT   EQU     7               ;Bits for the ZFDC flags 0 = IN, 1 = OUT
DATA$IN$RDY     EQU     0               ;Bit for data available from ZFDC board
DATA$OUT$RDY    EQU     1               ;Bit for data can be sent to ZFDC board
STD8IBM         EQU     1               ;IBM 8" SDSS Diak
NO$ERRORS$FLAG EQU      0               ;No Errors flag for previous cmd, sent back to S-100 BIOS


;Commands to the ZFDC Board:-

CMD$RESET$ZFDC EQU      3H              ;Reset the WD2793 chip and Board software
CMD$SET$FORMAT EQU      4H              ;This will select a specified drive and assign a disk format table to that drive
CMD$SET$DRIVE   EQU     5H              ;This will select a specified drive (0,1,2,3)
CMD$SET$TRACK   EQU     7H              ;This will set head request to a specified track
CMD$SET$SIDE    EQU     8H              ;This will set side request to a specified side
CMD$SET$SECTOR EQU      9H              ;This will set sector request to a specified sector
```

```
CMD$SET$HOME   EQU     0AH             ;This will set head request to Track 0 of CURRENT drive
CMD$STEP$IN    EQU     0BH             ;Step head in one track of CURRENT drive
CMD$SEEK$TRACK EQU     0EH             ;Seek to track to (IY+DRIVE$TRACK) with the track verify bit set on CURRENT drive/format
CMD$READ$SECTOR        EQU     10H             ;Read data from the CURRENT sector (on current track,side,drive).
CMD$HANDSHAKE  EQU     21H             ;Handshake command only sent during board initilization/testing
CMD$RD$MULTI$SECTOR EQU 29H            ;Read data from multiple sectors starting at the CURRENT sector (on current track,side,drive).


;-------------- PORT(S) TO SWITCH MASTER/SLAVE(S)

IOBYTE  EQU     0EFH                    ;IOBYTE (SEE ABOVE)
SW$TMAX EQU     0EEH                    ;OUTPUT BIT 0 HIGH FROM THIS PORT LOWERS DMA0* ON THE SMB_V2 (SWITCH IN THE 8086 FAMILY of boards)
                                        ;OUTPUT BIT 1 HIGH FROM THIS PORT LOWERS DMA1* ON THE SMB_V2 (SWITCH IN THE 68000 CPU Board)
                                        ;OUTPUT BIT 2 HIGH FROM THIS PORT LOWERS DMA2* ON THE SMB_V2
                                        ;OUTPUT BIT 3 HIGH FROM THIS PORT LOWERS DMA3* ON THE SMB_V2
SW$TMA0 EQU     0EDH                    ;INPUT FROM THIS PORT LOWERS DMA0* (SWITCHES IN THE 8088,8086,80286 or 80386 boards)
SW68K   EQU     0ECH                    ;INPUT FROM THIS PORT SWITCHES IN THE 68000 CPU Board with the (V1) SMB


;-------------- VERSAFLOPPY-II        FLOPPY DISK CONTROLLER COMMANDS ETC.

X       EQU     50H                     ;BASE PORT FOR 1791
RSET    EQU     X+0                     ;CONTROLLER RESET ADDRESS
SELECT  EQU     X+3                     ;DRIVE SELECT PORT
VERSA$STATUS    EQU     X+4             ;STATUS PORT
TRACK   EQU     X+5                     ;TRACK PORT
SECTOR  EQU     X+6                     ;SECTOR PORT
DATA    EQU     X+7                     ;DATA PORT
CMD     EQU     X+4                     ;COMMAND PORT

CIOBYTE EQU     03H
CDISK   EQU     04H
ZERO$L  EQU     08H                     ;Some of my CPM Loader's needs these to be zero!
ZERO$H  EQU     09H                     ;(The Non Banked version of CPM3). Need to later see why

@TADDR  EQU     40H
@UNIT   EQU     42H                     ;NEW @UNIT BYTE
@SCTR   EQU     43H                     ;SECTOR         (compatible with my oMOV CPM2.2 Versafloppy BIOS)
@TRK    EQU     44H                     ;TRACK
@NREC   EQU     45H                     ;# OF SECTORS
@ERMASK EQU     46H                     ;ERROR MASK
@ERSTAT EQU     47H                     ;ERROR FLAG STORE
@IDSV   EQU     48H                     ;6 BYTES (USED FOR TRACK ID COMMAND)
@CMDSV  EQU     4EH                     ;COMMAND SAVE
@SPSV   EQU     4FH                     ;SP SAVE
TEMP2   EQU     51H                     ;2 BYTE TEMP RECORD
@SIDE   EQU     51H                     ;SIDE STORE FOR MSDOS DISK
@COUNT  EQU     53H                     ;SECTORS/TRACK for BOOT (Currently unused)
@UNITCK EQU     55H                     ;OMOV @UNIT BYTE
@RSEEK  EQU     56H                     ;NBR OF RESEEKS
@RTRY   EQU     57H                     ;NBR OF RTRYS
ADRIVE  EQU     58H                     ;STORE OF A: DRIVE DENSITY ETC TYPE
BDRIVE  EQU     59H                     ;STORE OF B: DRIVE TYPE
@FDCTYPE EQU    5BH                     ;0FFH = ZFDC FDC Board Boot, else Versafloppy II FDC Boot,
@SEC$SIZE EQU   5CH                     ;Byte count of a sector fot loader
@SSTACK EQU     80H                     ;SYSTEM STACK
```

```
COLD    EQU     80H                     ;COMOV START ADDRESS FOR CPM FLOPPY (ONLY) BOOT LOADER

RDACMD  EQU     0C0H                    ;READ ADDRESS CODE
RDCMD   EQU     088H                    ;READ SECTOR CODE
WRCMD   EQU     0A8H                    ;WRITE SECTOR CODE
WRTCMD  EQU     0F4H                    ;WRITE TRACK CODE
RSCMD   EQU     008H                    ;RESTORE COMMAND (Note 3 Ms seek)
SKNCMD  EQU     018H                    ;SEEK NO VERIFY
FSKCMD  EQU     01CH                    ;FLOPPY SEEK COMAND
RSVCMD  EQU     00CH                    ;RESTORE WITH VERIFY COMMAND
MSKCMD  EQU     01FH                    ;MINI FLOPPY SEEK COMMAND

SRMASK  EQU     0FEH                    ;SECTOR READ ERROR BITS MASK

STDSDT  EQU     26                      ;STANDARD 8" 26 SECTORS/TRACK
STDDDT  EQU     50                      ;STANDARD DD 8" 50 SECTORS/TRACK
NBYTES  EQU     128                     ;BYTES/SECTOR
NTRKS   EQU     77                      ;TRACKS/DISK

;-------------- S100Computers IDE HARD DISK CONTROLLER COMMANDS ETC.

IDEAport        EQU     030H            ;lower 8 bits of IDE interface
IDEBport        EQU     031H            ;upper 8 bits of IDE interface
IDECport        EQU     032H            ;control lines for IDE interface
IDECtrl         EQU     033H            ;8255 configuration port
IDEDrivePort    EQU     034H            ;To select the 1st or 2nd CF card/drive (Not used with this monitor)

IDE$Reset$Delay         EQU     020H            ;Time delay for reset/initilization (~60 uS, with 10MHz Z80, 2 I/O wait states)

CPM$ADDRESS     EQU     100H            ;Will place the CPMMOVR.COM Loader here with
                                        ;CPMMOVR.COM will ALWAYS be on TRK 0,SEC2, (LBA Mode)
SEC$COUNT       EQU     12              ;CPMMOVR.COM requires (currently) 10, 512 byte sectors
                                        ;Add extra just in case
RDcfg8255       EQU     10010010B       ;Set 8255 IDECport out, IDEAport/B input
WRcfg8255       EQU     10000000B       ;Set all three 8255 ports output
;
IDEa0line       EQU     01H             ;direct from 8255 to IDE interface
IDEa1line       EQU     02H             ;direct from 8255 to IDE interface
IDEa2line       EQU     04H             ;direct from 8255 to IDE interface
IDEcs0line      EQU     08H             ;inverter between 8255 and IDE interface
IDEcs1line      EQU     10H             ;inverter between 8255 and IDE interface
IDEwrline       EQU     20H             ;inverter between 8255 and IDE interface
IDErdline       EQU     40H             ;inverter between 8255 and IDE interface
IDEreset        EQU     80H             ;inverter between 8255 and IDE interface
;
;Symbolic constants for the IDE Drive registers, which makes the
;code more readable than always specifying the address pins
;
REGdata         EQU     08H             ;IDEcs0line
REGerr          EQU     09H             ;IDEcs0line + IDEa0line
REGcnt          EQU     0AH             ;IDEcs0line + IDEa1line
REGsector       EQU     0BH             ;IDEcs0line + IDEa1line + IDEa0line
REGcyLSB        EQU     0CH             ;IDEcs0line + IDEa2line
REGcyMSB        EQU     0DH             ;IDEcs0line + IDEa2line + IDEa0line
REGshd          EQU     0EH             ;IDEcs0line + IDEa2line + IDEa1line          ;(0EH)
REGCMD          EQU     0FH             ;IDEcs0line + IDEa2line + IDEa1line + IDEa0line      ;(0FH)
```

```
REGstatus      EQU     0FH             ;IDEcs0line + IDEa2line + IDEa1line + IDEa0line
REGcontrol     EQU     16H             ;IDEcs1line + IDEa2line + IDEa1line
REGastatus     EQU     17H             ;IDEcs1line + IDEa2line + IDEa1line + IDEa0line

;IDE CMD Constants.  These shouMOV never change.
CMDrecal       EQU     10H
CMDread        EQU     20H
CMDwrite       EQU     30H
CMDinit        EQU     91H
CMDid          EQU     0ECH
CMDdownspin    EQU     0E0H
CMDupspin      EQU     0E1H
;
; IDE Status Register:
;  bit 7: Busy 1=busy, 0=not busy
;  bit 6: Ready 1=ready for CMD, 0=not ready yet
;  bit 5: DF   1=fault occured insIDE drive
;  bit 4: DSC  1=seek complete
;  bit 3: DRQ  1=data request ready, 0=not ready to xfer yet
;  bit 2: CORR 1=correctable error occured
;  bit 1: IDX  vendor specific
;  bit 0: ERR  1=error occured
;
;-------------------------------------------------------------------------
;
;CONNECTIONS TO Z80-MONB.Z80 :-
;
;BASE   EQU     0F000H                  ;Start or EPROM Location (Assume a 2732 or top half of a 28C64)
BASE    EQU     04000H                  ;<---- For testing monitor at 2000H in RAM with CPM/SID

        ORG     BASE                    ;<--------<<<<<< LOCATION OF START OF MONITOR (First part)
VERSA   EQU     BASE+800H               ;<--------<<<<<< LOCATION OF FLOPPY BIOS (For MOV Software)
                                        ;               NOTE MUST INSURE NO OVERFLOW OF THE FIRST
                                        ;               PART OR THIS MONITOR INTO THIS BIOS AREA

;PROGRAM CODE BEGINS HERE
;FIRST A JUMP TABLE FOR ALL JUMPS INTO THE MONITOR. NOTE THESE CANNOT BE
;CHANGED. WHERE POSSIBLE ZAPPLE FORMAT IS USED.

;       JMP     MOVEROM                 ;MOVE ROM UP TO HIGH RAM LOCATION
;       DB      0C3H, 0B0H, 00EH        ;<------------JUMP must be from 0H in RAM
        JMP     ZAPPLE                  ;<---- For testing monitor at 2000H in RAM with CPM/SID

ZCI:    JMP     CI                      ;CONSOL INPUT
ZRI:    JMP     SERIAL$IN               ;READER INPUT = Modem Input for Now
ZCO:    JMP     CO                      ;CONSOL OUTPUT
ZPOO:   JMP     SERIAL$OUT              ;PUNCH OUTPUT = Modem Output for Now
ZLO:    JMP     LO                      ;LIST OUTPUT
ZCSTS:  JMP     CSTS                    ;CONSOL STATUS
ZMEMCK: JMP     MEMSIZ                  ;GET HIGHEST RAM RETURNS IT IN [HL]
ZTRAP:  JMP     TRAP                    ;ERROR TRAP ADDRESS
ZSTART: JMP     START                   ;JUMP TO MONITOR DO NOT RESET HARDWARE (ALWAYS xx1BH)
ZTALK:  JMP     SPEAKOUT                ;SEND AN ASCII CHARACTER TO TALKER (One at a time)
ZTALKS: JMP     SPEAKER$CTS             ;STATUS FOR SPEECH CTS Line (V-Stamp CTS low when ready)
ZDELAY: JMP     DELAY                   ;SOFTWARE DELAY LENGTH IN [A]
ZLSTAT: JMP     LSTAT                   ;LIST STATUS
```

```
ZONLIST:JMP    ONLIST               ;INITILIZE LIST DEVICE
ZOFFLIST:JMP   OFLIST               ;TURN OFF LIST DEVICE
ZTIME:  JMP    PRINT$TIME           ;PUT TIME ON CRT @ CURSOR POSITION
ZDATE:JMP      PRINT$DATE           ;PRINT DATE ON CRT @ CURSOR POSITION
ZSPEAK$:JMP    SPEAK$               ;SEND ASCII STRING TO TALKER [HL] UP TO '$'
ZSERIAL$OUT:
       JMP     SERIAL$OUT           ;OUT TO ZILOG SCC SERIAL PORT
ZSERIAL$IN:
       JMP     SERIAL$IN            ;INPUT FROM ZILOG SCC SERIAL PORT
ZSERIAL$STAT:
       JMP     SERIAL$STAT          ;STATUS FROM ZILOG SCC SERIAL PORT
ZLOADER:JMP    LOADER               ;LOAD IN CPM IMAGE ON TRACKS 0 & 1 (VIA FLOPPY BOOT LOADER ON DISK SECTOR 1)
ZPMSG0: JMP    TOM                  ;DISPLAY STRING ON CONSOL [HL]=START ADD. [B]=LENGTH
ZPMSG$: JMP    PRINT$STRING         ;DISPLAY STRING ON CONSOL [HL]=START ADD. '$'=END
ZHLSP:  JMP    HLSP                 ;DISPLAY [HL] ON CONSOL THEN ONE SPACE
ZBITS:  JMP    BITS1                ;DISPLAY 8 BITS OF [A] ON CONSOL
ZLBYTE: JMP    LBYTE                ;DISPLAY [A] ON CONSOL
ZHEXSP: JMP    HEXSP                ;PUT 16 BIT PARAMETERS ON STACK FROM CONSOL, [C]=PARAMETER #
ZCRLF:  JMP    CRLF                 ;SEND CRLF TO CONSOL
ZHILO:  JMP    HILO                 ;RANGE CHECK (INC [HL], IF HL=DE THEN SET CARRY)
ZCONV:  JMP    CONV                 ;CONVERT HEX IN [A] TO ASCII IN [A]
ZDOS    JMP    DOS                  ;LOAD MSDOS FROM 5" DRIVE D:
ZPCHK:  JMP    PCHK                 ;INPUT FROM CONSOL & TEST FOR DELIMITERS RET {Z} IF
                                    ;SPACE OR , RET {C} IF A CR ELSE NON ZERO NON CARRY
VFLOPPY JMP    VBOOT                ;BOOT UP CPM-80 FROM VERSAFLOPPY II FDC
ZHARD:  JMP    HBOOTCPM             ;BOOT UP CPM-80 FROM HARD DISK
ZPRDY:  JMP    PRDY                 ;PUNCH READY CHECK
ZRSTAT: JMP    RSTAT                ;READER STATUS
ZCCHK:  JMP    CCHK                 ;CHECK FOR ^S & ESC AT KEYBOARD
ZFLOPPY JMP    ZBOOT                ;BOOT UP CPM-80 FROM ZFDC FDC
;
;       NOTE TABLE MUST BE WITHIN 0-FFH BOUNDRY
;
;COMMAND BRANCH TABLE

TBL:    DW  FLUSH      ; "@"  SEND FF to LaserJet printer
        DW  MEMMAP     ; "A"  DISPLAY A MAP OF MEMORY
        DW  SWITCH$68K  ; "B"  SWITCH CONTROL TO 68000 CPU
        DW  ZBOOT      ; "C"  BOOT IN CP/M FROM 8" DISK WITH WITH ZFDC FDC
        DW  DISP       ; "D"  DISPLAY MEMORY (IN HEX & ASCII)
        DW  ECHO       ; "E"  ECHO CHAR IN TO CHAR OUT
        DW  FILL       ; "F"  FILL MEMORY WITH A CONSTANT
        DW  GOTO       ; "G"  GO TO [ADDRESS]
        DW  SHOW$DATE  ; "H"  SHOW CURRENT DATE
        DW  SHOW$TIME  ; "I"  SHOW CURRENT TIME
        DW  RAMTEST    ; "J"  NON-DESTRUCTIVE MEMORY TEST
        DW  KCMD       ; "K"  DISPLAY THE LIST OF MONITOR COMMANDS
        DW  VBOOT      ; "L"  BOOT IN CP/M FROM 8" DISK WITH VERSAFLOPPY II FDC
        DW  MOVE       ; "M"  MOVE BLOCK OF MEMORY (START,FINISH,DESTINATION)
        DW  BEGIN      ; "N"  SPARE
        DW  START      ; "O"  SPARE
        DW  HBOOTCPM   ; "P"  BOOT IN CPM FROM SD CARD (Not done yet)
        DW  QUERY      ; "Q"  QUERY PORT (IN OR OUT)
        DW  INPORTS    ; "R"  Read ALL Input Ports
        DW  SUBS       ; "S"  SUBSTITUTE &/OR EXAMINE MEMORY
        DW  TYPE       ; "T"  TYPE ASCII PRESENT IN MEMORY
```

```
        DW  BEGIN       ; "U"  SPARE
        DW  VERIFY      ; "V"  COMPARE MEMORY
        DW  SWITCH$8086 ; "W"  INPUT Port ED (switched in 8086/80286)
        DW  START       ; "X"  BOOT IN MSDOS FROM HARD DISK (Not done yet)
        DW  BEGIN       ; "Y"  SPARE
        DW  SIZE        ; "Z"  FIND HIGHEST R/W RAM

;
;--------------------------------------------------------------------------
;
BEGIN:                              ;Initilize hardware
        XRA     A
        OUT     SD$CARD             ;Shadow out the ROM at 0H in RAM

ZAPPLE: MVI     A,0FFH              ;Clear Printer strobe, comes up 0 on a reset
        OUT     PRINTER$STROBE      ;also it turn all LED's off as a diagnostic
        MVI     A,00000000B         ;FLAG PROGRESS VISUALLY FOR DIAGNOSTIC (ALL LED' ON)
        OUT     DIAG$LEDS           ;LED's will go off one at a time

        MVI     A,0FFH
        OUT     SELECT              ;DESELECT ANY FLOPPYS ON VERSAFLOPPY FDC (If Present)

        MVI     A,10000000B         ;FLAG PROGRESS VISUALLY FOR DIAGNOSTIC (1 LED off)
        OUT     DIAG$LEDS

        MVI     A,0FFH
        OUT     RSET                ;RESET VERSAFLOPPY II FLOPPY DISK CONTROLLER (If Present)
        OUT     RESET$ZFDC$PORT             ;RESET ZFDC FLOPPY DISK CONTROLLER (If Present)

                                    ;We need to clear the 8259A otherewise teh 8086 monitor sometimes hangs
        MVI     A,MasterICW1        ;Initilize the 8259A PIC Controller (;EDGE triggered, 4 bytes, single Master,ICW4 needed)
        OUT     MASTER$PIC$PORT
        MVI     A,MasterICW2        ;Ints starts at 20H in RAM (IBM-PC uses 8X4 = 20H)
        OUT     MASTER$PIC$PORT+1
        MVI     A,MasterICW4        ;No slaves above, so 8259 does not expect ICW3
        out     MASTER$PIC$PORT+1

        MVI     A,11111111b         ;Allow no interrupts to 8259A with Z80.
        out     MASTER$PIC$PORT+1

        MVI     A,11000000B         ;FLAG PROGRESS VISUALLY FOR DIAGNOSTIC (2 LED's off)
        OUT     DIAG$LEDS

        LXI     SP,STACK            ;SETUP THE STACK AT 0EFF0H

        LXI     H,MSG0              ;Have a Stack, so we can use CALL
        CALL    PRINT$STRING

        CALL    INIT$S100$IO        ;Initilize the Zilog 8530 & 8255 on the S100Computers I/O Board

        MVI     A,11100000B         ;FLAG PROGRESS (Have a Stack with 3 LED's off)
        OUT     DIAG$LEDS

        CALL    PRINT$TIME          ;PRINT TIME ON CRT (IF RTC BOARD PRESENT)
        JC      NO$CLOCK
        LXI     H,GAP$MSG
```

```
        CALL    PRINT$STRING
        CALL    PRINT$DATE              ;PRINT DATE ON CRT, then CRLF
NO$CLOCK:
        CALL    CRLF

        MVI     A,11110000B             ;FLAG PROGRESS (I/O board initilized, 4 LED's Off)
        OUT     DIAG$LEDS

        LXI     H,SP$MSG                ;Print Current Stack Location
        CALL    PRINT$STRING


        CALL    HLSP                    ;Print HL/SP
        CALL    CRLF                    ;Then CRLF
        CALL    CSTS                    ;CHECK IF GARBAGE AT KEYBOARD
        CNZ     CI                      ;If so flush it

        MVI     A,11111000B             ;FLAG PROGRESS (Ready to go, 5 LED's off)
        OUT     DIAG$LEDS

        LXI     H,CR$SMSG               ;lets V-Stamp chip get baud rate
        CALL    SPEAK$

        MVI     A,11111100B             ;FLAG PROGRESS (Initilization done, 6 LED's off)
        OUT     DIAG$LEDS


;-------THIS IS THE START ON THE MAIN MONITOR LOOP--------------------------------

START:  LXI     D,START
        PUSH    D                       ;EXTRA UNBALANCED POP & [DE] WOULD END UP IN [PC]
        CALL    CRLF
        MVI     C,BELL                  ;A BELL HERE WILL SIGNAL WHEN JOBS ARE DONE
        CALL    CO
        MVI     C,'-'
        CALL    CO
        MVI     C,'>'
        CALL    CO

STARO:  CALL    TI                      ;Main loop. Monitor will stay here until cmd.
        ANI     7FH
        JZ      STARO
        SBI     '@'                     ;Commands @ to Z only
        RM
        CPI     1BH                     ;A-Z only
        RNC
        ADD     A
        LXI     H,TBL
        ADD     L
        MOV     L,A
        MOV     A,M
        INX     H
        MOV     H,M
        MOV     L,A
        MVI     C,02H
        PCHL                    ;JUMP TO COMMAND TABLE
```

```
;
;SEND MESSAGE TO CONSOL MESSAGE IN [HL],LENGTH IN [B]

TOM:    MOV     C,M
        INX     H
        CALL    CO
        DCR     B
        JNZ     TOM
        RET
;
PRINT$STRING:
        MOV     A,M                     ;A ROUTINE TO PRINT OUT A STRING @ [HL]
        INX     H                       ;UP TO THE FIRST '$'.
        CPI     '$'
        RZ
        MOV     C,A
        CALL    CO
        JMP     PRINT$STRING

;ABORT IF ESC  AT CONSOL, PAUSE IF ^S AT CONSOL

CCHK:   CALL    CSTS                    ;FIRST IS THERE ANYTHING THERE
        RZ
        CALL    CI
        CPI     'S'-40H
        JNZ     CCHK1
CCHK2:  CALL    CSTS                    ;WAIT HERE UNTIL ANOTHER INPUT IS GIVEN
        JZ      CCHK2
CCHK1:  CPI     ESC
        RNZ                             ;RETURN EXECPT IF ESC

;RESTORE SYSTEM AFTER ERROR

ERROR:  CALL    MEMSIZ                  ;GET RAM AVAILABLE - WORKSPACE IN [HL]
        XTHL                            ;[HL]->[SP]SET STACK UP IN WORKSPACE AREA
        MVI     C,'*'
        CALL    CO
        JMP     START

;PRINT HIGHEST MEMORY FROM BOTTOM

SIZE:
        CALL    MEMSIZ                  ;RETURNS WITH [HL]= RAM AVAILABLE-WORKSPACE

LFADR:  CALL    CRLF

;PRINT [HL] AND A SPACE
HLSP:   PUSH    H
        PUSH    B
        CALL    LADR
        MVI     C,SPACE
        CALL    CO
        POP     B
        POP     H
        RET
```

```
;PRINT A SPACE

SF488:  MVI     C,SPACE
        JMP     CO

;CONVERT HEX TO ASCII

CONV:   ANI     0FH
        ADI     90H
        DAA
        ACI     40H
        DAA
        MOV     C,A
        RET

;GET TWO PARAMETERS AND PUT THEM IN [HL] & [DE] THEN CRLF

EXLF:   CALL    HEXSP
        POP     D
        POP     H

;SEND TO CONSOL CR/LF

CRLF:   PUSH    B
        MVI     C,LF
        CALL    CO
        MVI     C,CR
        CALL    CO
        POP     B
        RET

;PUT THREE PARAMETERS IN [BC] [DE] [HL] THEN CR/LF

EXPR3:  INR     C                       ;ALREADY HAD [C]=2 FROM START
        CALL    HEXSP
        CALL    CRLF
        POP     B
        POP     D
        POP     H
        RET

;GET ONE PARAMETER

EXPR1:  MVI     C,01H
HEXSP:  LXI     H,0000
EX0:    CALL    TI
EX1:    MOV     B,A
        CALL    NIBBLE
        JC      EX2X
        DAD     H
        DAD     H
        DAD     H
        DAD     H
        ORA     L
        MOV     L,A
        JMP     EX0
```

```
EX2X:   XTHL
        PUSH    H
        MOV     A,B
        CALL    QCHK
        JNC     SF560
        DCR     C
        RZ
SF560:  JNZ     ERROR
        DCR     C
        JNZ     HEXSP
        RET
EXF:    MVI     C,01H
        LXI     H,0000H
        JMP     EX1

;RANGE TEST ROUTINE CARRY SET = RANGE EXCEEDED

HILOX:  CALL    CCHK
        CALL    HILO
        RNC
        POP     D                       ;DROP ONE LEVEL BACK TO START
        RET
HILO:   INX     H                       ;RANGE CHECK SET CARRY IF [DE]=[HL]
        MOV     A,H
        ORA     L
        STC
        RZ
        MOV     A,E
        SUB     L
        MOV     A,D
        SBB     H
        RET

;PRINT [HL] ON CONSOL

LADR:   MOV     A,H
        CALL    LBYTE
        MOV     A,L
LBYTE:  PUSH    PSW
        RRC
        RRC
        RRC
        RRC
        CALL    SF598
        POP     PSW
SF598:  CALL    CONV
        JMP     CO

;THIS IS A CALLED ROUTINE USED TO CALCULATE TOP OF RAM IS USED BY
;THE ERROR TO RESET THE STACK. Returns top of RAM in [HL]

MEMSIZ: PUSH    B                       ;SAVE [BC]
MEMSZ1: LXI     H,0EFFFH                ;START FROM THE TOP DOWN (Don't check Monitor area)
MEMSZ2: MOV     A,M
        CMA
        MOV     M,A
```

```
        CMP     M
        CMA                             ;PUT BACK WHAT WAS THERE
        MOV     M,A
        JZ      GOTTOP
        DCR     H                       ;TRY 100H BYTES LOWER
        JMP     MEMSZ2                  ;KEEP LOOKING FOR RAM
GOTTOP: POP     B                       ;RESTORE [BC]
        RET

NIBBLE: SUI     30H
        RC
        CPI     17H
        CMC
        RC
        CPI     LF
        CMC
        RNC
        SUI     07H
        CPI     LF
        RET

COPCK:  MVI     C,'-'
        CALL    CO

PCHK:   CALL    TI

;TEST FOR DELIMITERS

QCHK:   CPI     SPACE
        RZ
        CPI     ','
        RZ
        CPI     CR
        STC
        RZ
        CMC
        RET

;KEYBOARD HANDELING ROUTINE (WILL NOT ECHO CR/LF)
;IT CONVERTS LOWER CASE TO UPPER CASE FOR LOOKUP COMMANDS
;ALSO ^C WILL FORCE A JUMP TO BOOT IN CP/M
;ALL OTHERE CHARACTERS ARE ECHOED ON CONSOL

TI:     CALL    CI
        CPI     CR
        RZ
        CPI     'C'-40H                 ;^C TO BOOT IN CP/M
        JZ      FBOOT
        PUSH    B
        MOV     C,A
        CALL    CO
        MOV     A,C
        POP     B
        CPI     40H                     ;LC->UC
        RC
        CPI     7BH
```

```
        RNC
SF754:  ANI     5FH
        RET

BITS1:  PUSH    D                       ;DISPLAY 8 BITS OF [A]
        PUSH    B
        MOV     E,A
        CALL    BITS
        POP     B
        POP     D
        RET

BITS:   MVI     B,08H                   ;DISPLAY 8 BITS OF [E]
        CALL    SF488                   ;CRLF
BITS2:  MOV     A,E                     ;There is no 8080 equivalent to the Z80's SLA E
        ANI     80H
        JZ      BITS4
        MVI     C,'1'
BITS3:  CALL    CO
        MOV     A,E
        RAL
        MOV     E,A
        DCR     B
        JNZ     BITS2
        RET
BITS4:  MVI     C,'0'
        JMP     BITS3

;<<<<<<<<<<<<<<<<<<<<<<< MAIN CONSOL OUTPUT ROUTINE >>>>>>>>>>>>>>>>>>>>>>>>

CO:     IN      IOBYTE                  ;NOTE CHARACTER IS IN [C]
        ANI     01H                     ;CHECK IF OUTPUT TO PRINTER IS ALSO REQ
        JZ      LOX
SDCONO: IN      CONSOL$STATUS           ;SD SYSTEMS VIDIO BOARD PORT
        ANI     4H
        JZ      SDCONO
        MOV     A,C
        CPI     07H                     ;IS IT A BELL
        JZ      BELL1
        CPI     0H                      ;SD BOARD CANNOT TAKE A NULL!
        RZ
        OUT     CONSOL$OUT
        IN      IOBYTE
        ANI     20H                     ;BIT 5,A SEE IF SERIAL PORT OUTPUT IS REQ
        JNZ     SDCON5                  ;MAKE SURE TO RETURN CHARACTER SENT IN [A]
        CALL    SERIAL$OUT              ;Send data in [C] to Serial Port
SDCON5: MOV     A,C
        RET                             ;RETURN CHARACTER SENT IN [A]

LOX:    CALL    SDCONO                  ;OUTPUT TO BOTH PRINTER & CONSOLE
        CALL    LO
        RET

BELL1:  MVI     A,06H                   ;SEND A BELL
        OUT     CONSOL$OUT
        MVI     A,0FH
```

```
        CALL    DELAY
        MVI     A,07H
        OUT     CONSOL$OUT
        JMP     SDCON5


DELAY:  DCR     A                       ;GENERAL COUNT DOWN TIME DELAY
        RZ                              ;LENGTH SET IN [A]
        PUSH    PSW
        MVI     A,05H
MORE:   DCR     A
        PUSH    PSW
        XRA     A
MORE2:  DCR     A
        JNZ     MORE2
        POP     PSW
        JNZ     MORE
        POP     PSW
        JMP     DELAY


;<<<<<<<<<<<<<<<<<<< MAIN CONSOL STATUS ROUTINE >>>>>>>>>>>>>>>>>>>>>>

CSTS:   IN      CONSOL$STATUS
        ANI     02H
        JZ      TRYSER          ;See if input from Serial Port is req
        XRA     A
        DCR     A                       ;RETURN WITH 0FFH IN [A] IF SOMETHING
        RET
TRYSER: IN      IOBYTE
        ANI     20H                     ;SEE IF SERIAL PORT INPUT IS REQ
        JZ      SERIAL$STAT             ;Check if anything at Modem IN status routine
        XRA     A                       ;IF IOBYTE bit 1 then skip modem input
        RET                             ;RETURN WITH 0 IN A IF NOTHING THERE


;<<<<<<<<<<<<<<<<<<< MAIN CONSOL INPUT ROUTINE >>>>>>>>>>>>>>>>>>>>>>

CI:     IN      CONSOL$STATUS           ;NEED CONSTAT TO CLEAN UP SHIFT KEYS ETC
        ANI     02H
        JZ      CHKSERIAL
        IN      CONSOL$IN
        ANI     7FH
        RET
CHKSERIAL:
        IN      IOBYTE
        ANI     20H                     ;SEE IF SERIAL PORT INPUT IS REQ
        JNZ     CI                      ;NO, then do normal CI
        CALL    SERIAL$STAT             ;See if anything at Modem input
        JZ      CI
        JMP     SERIAL$IN


;>>>>>>>>>>>>>>>>>>>> MAIN PRINTER OUTPUT ROUTINE <<<<<<<<<<<<<<<<<<<<<

LO:     MVI     B,0FFH
```

```
LO2:    CALL    LSTAT
        JNZ     LO1
        DCR     B
        JNZ     LO2
        XRA     A
        MOV     A,C
        RET                             ;RET Z if Printer problem (Not switched on)
LO1:    MVI     A,0FFH                  ;Setup strobe high to low then high
        OUT     PRINTER$STROBE
        MOV     A,C
        OUT     PRINTER$OUT             ;Now Data
        XRA     A                       ;STROBE FOR CENTRONICS
        OUT     PRINTER$STROBE
        MVI     A,0FFH                  ;Raise strobe again
        OUT     PRINTER$STROBE
        ORA     A
        RET                             ;Ret NZ if OK

FLUSH:  MVI     C,FF                    ;Send a Form Feed to laserJet Printer
        CALL    LO                      ;This forces a partial page to be printed
        RET

;>>>>>>>>>>>>>>>>>>>>>> PRINTER STATUS ROUTINE <<<<<<<<<<<<<<<<<<<<<<<<

LSTAT:  IN      PRINTER$STATUS
        ANI     00001111B               ;XXXX0110 IS READY (BIT 3=PAPER BIT 2=FAULT
        CPI     00000110B               ;BIT 1=SELECT  BIT 0=BUSY
        JZ      LSTAT1
        XRA     A
        RET
LSTAT1: XRA     A                       ;PUT 0FFH IN [A] IF READY & NO ZERO FLAG
        DCR     A
        RET


;-------------- BOOT UP CPM FROM HARD DISK ON S100COMPUTERS IDR BOARD ----------------

;BOOT UP THE 8255/IDE Board HARD DISK/Flash Memory Card
;NOTE CODE IS ALL HERE IN CASE A 2716 IS USED

HBOOTCPM:
        POP     H                       ;CLEAN UP STACK
        JMP     ERR$NR                  ;For now just say drive not ready
;------------------------------------------------------------------------------

;MEMORY MAP PROGRAM CF.DR.DOBBS VOL 31 P40.
;IT WILL SHOW ON CONSOL TOTAL MEMORY SUMMARY OF RAM,PROM, AND NO MEMORY
;
MEMMAP:
        CALL    ZCRLF
        LXI     H,0
        MVI     B,1
MAP1:   MVI     E,'R'                   ;PRINT R FOR RAM
        MOV     A,M
        CMA
        MOV     M,A
```

```
        CMP     M
        CMA
        MOV     M,A
        JNZ     MAP2
        CMP     M
        JZ      PRINT
MAP2:   MVI     E,'p'
MAP3:   MVI     A,0FFH
        CMP     M
        JNZ     PRINT
        INR     L
        XRA     A
        CMP     L
        JNZ     MAP3
        MVI     E,'.'
PRINT:  MVI     L,0
        DCR     B
        JNZ     NLINE
        MVI     B,16
        CALL    ZCRLF
        CALL    HXOT4
NLINE:  MVI     A,SPACE
        CALL    OTA
        MOV     A,E
        CALL    OTA
        INR     H
        JNZ     MAP1
        CALL    ZCRLF
        CALL    ZCRLF
        JMP     ZSTART

;16 HEX OUTPUT ROUTINE

HXOT4:  MOV     C,H
        CALL    HXO2
        MOV     C,L
HXO2:   MOV     A,C
        RAR
        RAR
        RAR
        RAR
        CALL    HXO3
        MOV     A,C
HXO3:   ANI     0FH
        CPI     10
        JC      HADJ
        ADI     7
HADJ:   ADI     30H
OTA:    PUSH    B
        MOV     C,A
        CALL    ZCO                     ;SEND TO CONSOL
        POP     B
        RET

;DISPLAY MEMORY IN HEX
```

```
DISP:   CALL    EXLF                ;GET PARAMETERS IN [HL],[DE]
        MOV     A,L                 ;ROUND OFF ADDRESSES TO XX00H
        ANI     0F0H
        MOV     L,A
        MOV     A,E                 ;FINAL ADDRESS LOWER HALF
        ANI     0F0H
        ADI     10H                 ;FINISH TO END 0F LINE
SF172:  CALL    LFADR
SF175:  CALL    BLANK
        MOV     A,M
        CALL    ZLBYTE
        CALL    HILOX
        MOV     A,L
        ANI     0FH
        JNZ     SF175
        MVI     C,TAB               ;INSERT A TAB BETWEEN DATA
        CALL    ZCO
        MVI     B,4H                ;ALSO 4 SPACES
TA11:   MVI     C,SPACE
        CALL    ZCO
        DCR     B
        JNZ     TA11
        MVI     B,10H               ;RAM pointer back to start of line location
SUBX:   DCX     H                   ;8080 does not have a SBC HL,DE opcode
        DCR     B
        JNZ     SUBX
        MVI     B,16                ;NOW PRINT ASCII (16 CHARACTERS)

T11:    MOV     A,M
        ANI     7FH
        CPI     ' '                 ;FILTER OUT CONTROL CHARACTERS
        JNC     T33
T22:    MVI     A,'.'
T33:    CPI     07CH
        JNC     T22
        MOV     C,A                 ;SET UP TO SEND
        CALL    ZCO
        INX     H
        DCR     B
        JNZ     T11                 ;REPEAT FOR WHOLE LINE
        JMP     SF172

BLANK:  MVI     C,' '
        JMP     ZCO

;INSPECT AND / OR MODIFY MEMORY

SUBS:   MVI     C,1
        CALL    ZHEXSP
        POP     H
SF2E3:  MOV     A,M
        CALL    ZLBYTE
        MVI     C,'-'
        CALL    ZCO
        CALL    ZPCHK
        RC
```

```
        JZ      SF2FC
        CPI     5FH
        JZ      SF305
        PUSH    H
        CALL    EXF
        POP     D
        POP     H
        MOV     M,E
        MOV     A,B
        CPI     CR
        RZ
SF2FC:  INX     H
SF2FD:  MOV     A,L
        ANI     07H
        CZ      LFADR
        JMP     SF2E3
SF305:  DCX     H
        JMP     SF2FD

;FILL A BLOCK OF MEMORY WITH A VALUE

FILL:   CALL    EXPR3
SF1A5:  MOV     M,C
        CALL    HILOX
        JNC     SF1A5
        POP     D
        JMP     ZSTART

;GO TO A RAM LOCATION

GOTO:   MVI     C,1                     ;SIMPLE GOTO FIRST GET PARMS.
        CALL    HEXSP
        CALL    CRLF
        POP     H                       ;GET PARAMETER PUSHED BY EXF
        PCHL

; GET OR OUTPUT TO A PORT
; Unlike the Z80 the 8080 does not have the ability to input from a port
; defined in the [C] register. Port numbers are fixed in code.
; The only way around this is to write a small program in RAM and call
; it from this (fixed) ROM.

QUERY:  CALL    ZPCHK
        CPI     'O'                     ;OUTPUT TO PORT
        JZ      SF77A
        CPI     'I'                     ;INPUT FROM PORT
        JZ      QQQ1
        MVI     C,'*'
        JMP     ZCO                     ;WILL ABORT IF NOT 'I' OR 'O'


                                        ;--- INPUT FROM PORT ---
QQQ1:   XTHL                            ;EX SP<->HL
        MOV     D,H
        MOV     E,L                     ;HL->DE
        XTHL                            ;Return Stack back to its original value
```

```
        MVI     B,10H                   ;Get RAM location 10H below stack (to be safe)
SUBDX1: DCX     D
        DCR     B
        JNZ     SUBDX1
        XCHG                            ;EX DE<->HL ([HL] now points well below the stack)
        PUSH    H                       ;Save RAM location
        PUSH    H                       ;2 times

        MVI     C,1                     ;Only one value required, the port #
        CALL    ZHEXSP
        POP     B                       ;Port # is in [C]

        POP     H
        MVI     M,0DBH                  ;IN opcode
        INX     H
        MOV     M,C                     ;Port #
        INX     H
        MVI     M,0C3H                  ;JMP opcode
        INX     H
        LXI     D,PORT$RET$IN           ;Put location where the RAM code will return to in [DE]
        MOV     M,E                     ;Place in RAM return location to here in ROM
        INX     H
        MOV     M,D
        POP     H                       ;Back to start or RAM code
        PCHL
PORT$RET$IN:
        JMP     ZBITS



                                        ;--- OUTPUT TO PORT ---
SF77A:  XTHL                            ;EX SP<->HL
        MOV     D,H
        MOV     E,L                     ;HL->DE
        XTHL                            ;Return Stack back to its original value

        MVI     B,10H                   ;Get RAM location 10H below stack (to be safe)
SUBDX:  DCX     D
        DCR     B
        JNZ     SUBDX
        XCHG                            ;EX DE<->HL ([HL] now points well below the stack)
        PUSH    H                       ;Save RAM location
        PUSH    H                       ;2 times

        CALL    ZHEXSP                  ;Get values
        POP     D
        POP     B

        POP     H                       ;Get saved RAM pointer
        MVI     M,0D3H                  ;OUT opcode
        INX     H
        MOV     M,C                     ;Port #
        MOV     A,E                     ;Save value to output to the port #
        INX     H
        MVI     M,0C9H                  ;RET Opcode
```

```
        POP     H                       ;Back to start or RAM code
        PCHL                            ;HL->PC
        HLT                             ;Just in case



; MEMORY TEST

RAMTEST:CALL    EXLF
SF200:  MOV     A,M
        MOV     B,A
        CMA
        MOV     M,A
        XRA     M
        JZ      SF215
        PUSH    D
        MOV     D,B
        MOV     E,A                     ;TEMP STORE BITS
        CALL    ZHLSP
        CALL    BLANK
        MOV     A,E
        CALL    ZBITS
        CALL    ZCRLF
        MOV     B,D
        POP     D
SF215:  MOV     M,B
        CALL    HILOX
        JMP     SF200

;MOVE A BLOCK OF MEMORY TO ANOTHER LOCATION

MOVE:   CALL    EXPR3
SF21E:  MOV     A,M
        PUSH    H                       ;For MOV (BC),A
        PUSH    B                       ;BC -> HL
        POP     H
        MOV     M,A
        INX     H                       ;For INC BC
        PUSH    H                       ;HL -> BC
        POP     B
        POP     H
        CALL    HILOX
        JMP     SF21E

;VERIFY ONE BLOCK OF MEMORY WITH ANOTHER

VERIFY: CALL    EXPR3
VERIO:  PUSH    H                       ;For MOV A,(BC)
        PUSH    B                       ;BC -> HL
        POP     H
        MOV     A,M
        POP     H                       ;Get back origional HL
        CMP     M                       ;CMP A,(BC)
        JZ      SF78E
        PUSH    B
        CALL    CERR
```

```
        POP     B
SF78E:  INX     B
        CALL    HILOX
        JMP     VERIO
        RET

CERR:   MOV     B,A
        CALL    ZHLSP
        MOV     A,M
        CALL    ZLBYTE
        CALL    BLANK
        MOV     A,B
        CALL    ZLBYTE
        JMP     ZCRLF

ECHO:   CALL    CI                      ;Routeen to check keyboard etc.
        CPI     'C'-40H                 ;Loop until  ^C or ESC character is entered
        RZ
        CPI     'Z'-40H
        RZ
        CPI     ESC
        RZ
        MOV     C,A
        CALL    CO
        JMP     ECHO



;READ ASCII FROM MEMORY

TYPE:   CALL    EXLF
SF30B:  CALL    LFADR
        MVI     B,56
SF310:  MOV     A,M
        ANI     7FH
        CPI     SPACE
        JNC     SF319
SF317:  MVI     A,2EH
SF319:  CPI     7CH
        JNC     SF317
        MOV     C,A
        CALL    ZCO
        CALL    HILOX
        DCR     B
        JNZ     SF310
        JMP     SF30B


;       Display all active IO input ports in the system
;
INPORTS:CALL    ZCRLF
        MVI     D,6                     ;Display 6 ports across
        MVI     E,0FFH                  ;Will contain port number
LOOPIO: MOV     A,E                     ;[A] will contain port # for below
        CPI     SW$TMA0                 ;Inputting here will switch out the 8080 to 8086/80286
```

```
        JZ      SKIP
        CPI     SW68K                   ;Also this one (68K, for V1 SMB)
        JZ      SKIP
        PUSH    D                       ;Save display count and port #

                                        ;The 8080 has no IN  A,(C) so must put code in RAM (below the stack)
        XTHL                            ;EX SP<->HL
        MOV     D,H
        MOV     E,L                     ;HL->DE
        XTHL                            ;Return Stack back to its original value

        MVI     B,10H                   ;Get RAM location 10H below stack (to be safe)
SUBDXX: DCX     D
        DCR     B
        JNZ     SUBDXX
        XCHG                            ;EX DE<->HL ([HL] now points well below the stack)
        PUSH    H                       ;Save RAM location

        MVI     M,0DBH                  ;IN opcode
        INX     H
        MOV     M,A                     ;Port #
        INX     H
        MVI     M,0C3H                  ;JMP opcode
        INX     H
        LXI     D,RET$IN                ;Put location where the RAM code will return to in [DE]
        MOV     M,E                     ;Place in RAM return location to here in ROM
        INX     H
        MOV     M,D
        POP     H                       ;Back to start or RAM code
        PCHL

RET$IN: POP     D                       ;Get back Port/Count info
        CPI     0FFH                    ;No need for 0FF's
        JZ      SKIP
        MOV     H,A                     ;store hardware returned port data in H for below
        MOV     A,E                     ;Need to print port # first
        CALL    LBYTE                   ;Print port number
        MVI     C,'-'
        CALL    ZCO
        MVI     C,'>'
        CALL    ZCO
        MOV     A,H                     ;get back port data
        CALL    LBYTE                   ;print it
        MVI     C,TAB
        CALL    ZCO
        DCR     D                       ;6 ports per line
        JNZ     SKIP
        MVI     D,6
        CALL    ZCRLF
SKIP:   DCR     E
        MOV     A,E
        CPI     0FFH
        JNZ     LOOPIO
        CALL    ZCRLF
        RET
```

```
;S100Computers Serial I/O Board Initilization
;Note both Zilog SCC serial ports (A & B) will be set to 19,200 Baud initially.

INIT$S100$IO:                           ;First the 8255
        MVI     A,AinBout8255cfg        ;A input, B output, C(bits 0-3) output, (bits 4-7)input
        OUT     PortCtrl$8255           ;Config 8255 chip, Mode 0

                                        ;Then the SCC
        MVI     B,0EH                   ;Byte count for OTIR below
        LXI     H,SCCINIT
MORE$IO1:
        MOV     A,M                     ;8080 does not have an OTIR opcode
        OUT     ACTL                    ;Program Channel A
        INX     H
        DCR     B
        JNZ     MORE$IO1

        MVI     B,0EH                   ;Byte count for OTIR below
        LXI     H,SCCINIT
MORE$IO2:
        MOV     A,M                     ;8080 does not have an OTIR opcode
        OUT     BCTL                    ;Program Channel B
        INX     H
        DCR     B
        JNZ     MORE$IO2
        RET

;       ALL SSC's are set for 19,200 BAUD
SCCINIT:
        DB      04H                     ;Point to WR4
        DB      44H                     ;X16 clock,1 Stop,NP
;
        DB      03H                     ;Point to WR3
        DB      0C1H                    ;Enable reciever, Auto Enable, Recieve 8 bits
;       DB      0E1H                    ;Enable reciever, No Auto Enable, Recieve 8 bits (for CTS bit)
;
        DB      05H                     ;Point to WR5
        DB      0EAH                    ;Enable, Transmit 8 bits
;                                       ;Set RTS,DTR, Enable
;
        DB      0BH                     ;Point to WR11
        DB      56H                     ;Recieve/transmit clock = BRG
;
        DB      0CH                     ;Point to WR12
;       DB      40H                     ;Low Byte 2400 Baud
;       DB      1EH                     ;Low Byte 4800 Baud
;       DB      0EH                     ;Low Byte 9600 Baud
        DB      06H                     ;Low byte 19,200 Baud <<<<<<<<<<< Note Speech synthizer defaults to this value
;       DB      02H                     ;Low byte 38,400 Baud
;       DB      00H                     ;Low byte 76,800 Baud
;
        DB      0DH                     ;Point to WR13
        DB      00H                     ;High byte for Baud
;
        DB      0EH                     ;Point to WR14
```

```
        DB      01H                     ;Use 4.9152 MHz Clock. Note SD Systems uses a 2.4576 MHz clock, enable BRG
;
        DB      0FH                     ;Point to WR15
        DB      00H                     ;Generate Int with CTS going high


;PRINT MAIN MONITOR MENU ON CRT

KCMD:   LXI     H,MSG0                  ;Signon Msg again (K Command)
        CALL    PRINT$STRING
        LXI     H,SMSG                  ;SPEECH MESSAGE
        CALL    SPEAK$
        LXI     H,MENUMSG               ;Then Menu Message
        JMP     PRINT$STRING

;
;THIS ROUTINE JUMPS OVER TO THE 8088, 8086 or 80286. Port SW86 raises S-100 PIN #55
;THIS WILL CAUSE THE 8086/80286 BOARD TO BECOME ACTIVE AND TAKE OVER THE BUS. THE
;8080 WILL BE IN A PERMANANT HOLD STATE UNTIL PIN #55 IS AGAIN LOWERED.

SWITCH$8086:
        LXI     H,MSG14
        CALL    PRINT$STRING
        IN      SW$TMA0                 ;THIS SWITCHES CPU'S
        NOP                             ;8080 WILL BE HELD HERE
        NOP
        MVI     A,01                    ;Utilize the more specific circuit on the V2-SMB
        OUT     SW$TMAX                 ;Make sure its bit 0
        NOP
        NOP
        JMP     BEGIN                   ;WILL DROP BACK TO REBOOT MONITOR



;THIS ROUTINE JUMPS OVER TO THE 68000 CPU Board. Port SW68K raises S-100 PIN #56
;THIS WILL CAUSE THE 68000 CPU BOARD TO BECOME ACTIVE AND TAKE OVER THE BUS. THE
;8080 WILL BE IN A PERMANANT HOLD STATE UNTIL PIN #56 IS AGAIN LOWERED.

SWITCH$68K:
        LXI     H,MSG68K
        CALL    PRINT$STRING
        IN      SW68K                   ;THIS SWITCHES CPU'S with the old (V1)SMB
        NOP                             ;68K WILL BE HELD HERE
        NOP
        MVI     A,02H                   ;For V2 SMB
        OUT     SW$TMAX                 ;Make sure its bit 1 for 68K
        NOP                             ;8080 WILL BE HELD HERE
        NOP
        JMP     BEGIN                   ;WILL DROP BACK TO REBOOT MONITOR

;
MSG0:   DB SCROLL,QUIT,NO$ENHANCEMENT,FAST,BELL,CR,LF,LF
        DB '8080 ROM MONITOR V1.2 (John Monahan, 8/1/2014)  $'
SMSG:   DB 'THE 8080 ROM MONITOR VERSION 1.2 IS NOW RESIDENT $'

NOP
```

```
NOP
NOP
;-----------------------------------------------------------------------------------------------
;
;    <<< BE SURE ADDITIONS TO CODE ABOVE WILL NOT OVERWRITE CODE BELOW HERE >>>
;    <<< OTHEREWISE THIS WILL LEAD TO STRANGE PROPORTIES.
;    <<< RELOCATE SECTIONS IF YOU NEED MORE ROOM REVIEW THE 8080.PRN FILE >>>
;
;-----------------------------------------------------------------------------------------------

       ORG     VERSA                ;<--------- THIS LOCATION MUST NOT BE CHANGED (F800H)
                                    ;My Load CPM V1.4 systems are counting on it being here
;
;      VERSAFLOPPY II DOS SYSTEM LINKAGES    (USED BY SDOS & 2.2 CP/M)
;      These are residule JMP's for Load CPM BIOS'es. Only LOADER is now functional.

FBOOT:  JMP     BOOT                 ;COLD START ENTRY
WBOOT:  JMP     BIOS$JMP$ERR         ;WARM START ENTRY
CSE:    JMP     ZCSTS                ;CONSOLE STATUS
CIE:    JMP     ZCI                  ;CONSOLE IN
COE:    JMP     ZCO                  ;CONSOLE OUT
LIST:   JMP     ZLO                  ;TO MONITOR FOR PRINTER
PUNCH:  JMP     ZPOO                 ;TO MONITOR FOR PUNCH
READR:  JMP     ZRI                  ;TO MONITOR FOR READER
HME:    JMP     BIOS$JMP$ERR         ;HOME          ;MOVE TO TRACK 0
SDSKE:  JMP     BIOS$JMP$ERR         ;SEEK
S@TRKE: JMP     BIOS$JMP$ERR         ;SET@TRK
SSECE:  JMP     BIOS$JMP$ERR         ;SETSEC
SDMAE:  JMP     BIOS$JMP$ERR         ;SETDMA
RDE:    JMP     BIOS$JMP$ERR         ;READF
WRE:    JMP     BIOS$JMP$ERR         ;WRITEF
LISTS:  JMP     LSTAT                ;LIST STATUS
SECTR:  JMP     BIOS$JMP$ERR         ;SECTRAN FOR 2.2 SECTOR TRANSLATION TABLE

DTYPE:  JMP     BIOS$JMP$ERR         ;UNITSL SET UP @UNIT BYTE (DISK DENSITY)
SVE:    JMP     BIOS$JMP$ERR         ;SAVER  SAVE N RECORDS
MMOVE:  JMP     LOADER               ;LOADER LOAD N SECTORS FROM TRACK 0 (& TRACK 1)

BIOS$JMP$ERR:
       LXI     H,BIOS$ERR           ;"BIOS JMP longer implemented in ROM @ F800H."
       JMP     ABORT$ERR$MSG




;BOOT LOADS A SECTOR TO 80H AND THEN JUMPS TO 80H
;NOTE. Two FDC Boards are supported here:-
;
;      VFDC$BOOT Boots CPM from the Versafloppy-II disk controller board
;      ZFDC$BOOT Boots CPM from the ZFDC controller board


VBOOT:  XRA     A                    ;0 = Flag as Boot from Versafloppy II FDC
       JMP     BOOT$COMOV
ZBOOT:  XRA     A
       DCR     A                    ;0FFH = Flag as Boot from ZFDC FDC
```

```
BOOT$COMOV:
        STA     @FDCTYPE                ;0 for VF, 0FFH for ZFDC

BOOT:   MVI     A,11111111B             ;FLAG PROGRESS VISUALLY FOR DIAGNOSTIC
        OUT     DIAG$LEDS

        LXI     H,SPEAKCPM$MSG          ;Announce on speaker
        CALL    SPEAK$
        XRA     A
        STA     CDISK                   ;MAKE CURRENT DISK A:
        STA     CIOBYTE                 ;CLEANUP IOBYTE
        STA     @UNIT                   ;8LOAD.Z80 (The first sector loader module) will count on this being 0H
                                        ;for the Versafloppy-II boots
        STA     ZERO$L                  ;These need to be zero's here for the CPM Loader/Versafloppy-II of my oMOV
        STA     ZERO$H                  ;NON-BANKED CPM3 or CPM2.2 disks. Need to later find out why!
        LXI     H,128                   ;Assume 128 byte sectors for 8" disk
        SHLD    @SEC$SIZE

BOOTW1: LXI     SP,@SSTACK
        LDA     @FDCTYPE                ;Are we using a Versafloppy II or ZFDC FDC board
        ORA     A
        JNZ     ZFDC$BOOT               ;<<<<<<<<< Go to ZFDC Board BOOT >>>>>>>>>>>>>

VFDC$BOOT:
        LXI     H,BOOT$MSG0             ;<<<<<<<<< BOOT FROM VERSAFLOPPY-II >>>>>>>>>>>>
        CALL    PRINT$STRING            ;"Loading CPM from VF FDC"
        LXI     H,VF$MSG
        CALL    PRINT$STRING

        MVI     A,0D0H                  ;FORCE CHIP INTERUPT
        OUT     CMD

        MVI     A,STDSDT                ;SETUP FOR SD
        STA     @COUNT                  ;STORE AS 26 SECTORS/TRACK
        MVI     A,0FEH
        OUT     SELECT                  ;Select Drive A: (Always)

        XRA     A
        STA     @TRK
        INR     A
        STA     @SCTR

        CALL    READY$CHK               ;Critical to make sure chip is ready first!
        MVI     A,RSCMD                 ;RESTORE COMMAND (Note 3 Ms seek)
        OUT     CMD
        CALL    READY$CHK               ;Critical to make sure chip is ready first!

        LXI     H,COLD
        SHLD    @TADDR

        CALL    VF$READ$SECTOR          ;Read the Boot Sector
BOOT$SEC$READ:
        JNZ     ERR$MOV
BOOT$SEC$CHECK:
        LXI     H,COLD
        MOV     A,M
```

```
        CPI     31H                     ;EXPECT TO HAVE 31H @80H IE. MOV SP,80H
        JZ      COLD                    ;AS THE FIRST INSTRUCTION. IF OK JMP 80H
        JMP     ERR$LD1                 ;Boot Sector Data incorrect


VF$READ$SECTOR:                                 ;READ SECTOR COMMAND
        MVI     B,3                     ;Will Try 3 times
READ1:  PUSH    B
        CALL    DRINIT                  ;Setup sector paramaters
        MOV     A,E
        CPI     80H                     ;128 or 512 byte sectors ?
        MVI     B,128
        DI
        MVI     A,RDCMD
        OUT     CMD                     ;Note wait states are now switched on

        JMP     M2
M2:     JMP     MM2
MM2:    JZ      RD$128                  ;From CPI 80H above
        MVI     B,0                     ;256X2

MM3:    IN      DATA                    ;For Z80 INIR,  [C]-> [HL++],[B--]
        MOV     M,A
        INX     H
        DCR     B
        JNZ     MM3

RD$128: IN      DATA                    ;128 or 256 bytes
        MOV     M,A
        INX     H
        DCR     B
        JNZ     RD$128

        EI
        CALL    WAITF                   ;Wait states are now off
        IN      VERSA$STATUS
        ANI     SRMASK                  ;Check sector was read OK
        POP     B
        RZ
        DCR     B
        JNZ     READ1
        XRA     A
        DCR     A
        RET                             ;Return NZ if failure after 3 reads


DRINIT: CALL    SEEK                    ;DRIVE INITIALIZATION
        LHLD    @TADDR                  ;SETUP DMA ADDRESS AND BYTE COUNT
        LDA     @SCTR
        OUT     SECTOR

        PUSH    H
        LHLD    @SEC$SIZE               ;This will be 128 or 512 sectors
        PUSH    H
        POP     D
        POP     H
```

```
        MVI     C,DATA                  ;8067H in BC

SWEB:   IN      SELECT                  ;ENABLE WAIT STATES
        ANI     7FH
        OUT     SELECT
        RET


;       SEEK TRACK
SEEK:   LDA     @TRK
        MOV     C,A
        IN      TRACK
        CMP     C
        RZ                              ;IF SAME TRACK NO NEED TO SEEK

        LDA     @TRK
        OUT     DATA
        CALL    READY$CHK               ;Critical to make sure chip is ready first!
        MVI     A,FSKCMD                ;Send Seeek Command to WD1791
        OUT     CMD
        CALL    DELAY$15                ;Delay ~15ms
        CALL    READY$CHK
        IN      TRACK
        MOV     C,A
        LDA     @TRK
        CMP     C
        RZ
        LXI     H,SEEK$ERROR$MSG
        JMP     ABORT$ERR$MSG


READY$CHK:
        LXI     B,0
READY$CHK1:
        IN      VERSA$STATUS
        ANI     1
        RZ
        DCX     B
        MOV     A,C
        ORA     B
        JNZ     READY$CHK1              ;Wait until 1791/5 is ready
        JMP     WAIT3

WAITF:  MVI     E,0
        PUSH    B
        MVI     C,2
WAITX2: IN      VERSA$STATUS
        ANI     1
        JZ      DWAIT
        DCR     B
        JNZ     WAITX2
        DCR     E
        JNZ     WAITX2
        DCR     C
        JNZ     WAITX2
        POP     B
```

```
WAIT3:  IN      SELECT                  ;IF BY THIS TIME NOT READY FORCE
        ORI     80H                     ;A HARDWARE RESET
        OUT     RSET
        LXI     H,VF$HUNG
        JMP     ABORT$ERR$MSG


;       DISABLE WAIT STATES
DWAIT:  POP     B                       ;TO BALANCE THE ABOVE PUSH IN WAIT
DDWAIT: IN      SELECT
        ORI     80H
        OUT     SELECT
        RET

DELAY$15:                               ;DELAY ~15 MS
        MVI     A,40
DELAY1: MVI     B,0
M0:     DCR     B
        JNZ     M0
        DCR     A
        JNZ     DELAY1
        RET

DELAY$150:                              ;DELAY ~150 MS
        MVI     C,10
DELAY320A:
        CALL    DELAY$15
        DCR     C
        JNZ     DELAY320A
        RET

LOADER: MVI     A,(@FDCTYPE)            ;Are we using a Versafloppy II or ZFDC FDC board
        ORA     A
        JNZ     ZFDC$LOADER             ;Go to ZFDC Board Loader


;       LOAD A NUMBER OF SECTORS
VF$LOADER:
        CALL    VF$READ$SECTOR
        JNZ     ERR$MOV
        MVI     C,'.'                   ;Show progress
        CALL    CO
        CALL    INCP
        JNZ     VF$LOADER
        RET

;       INC SECTOR AND TRACK
INCP:   LHLD    @TADDR
        PUSH    H
        LHLD    @SEC$SIZE               ;128 or 512 byte sectors
        PUSH    H
        POP     D                       ;@SEC$SIZE -> DE
        POP     H
INCP2:  DAD     D
        SHLD    @TADDR
        LHLD    @NREC
```

```
        DCR     M
        RZ                          ;Return when we have done all sectors (~51)
        LXI     H,@SCTR
        INR     M
        LDA     @COUNT              ;IS ONE TRACK DONE YET (Sec/track+1)
        INR     A
        CPI     M
        RNZ                         ;IF FULL Z, THEN GO TO NEXT TRACK
        MOV     M,1                 ;SET SECTOR COUNT BACK TO 1
        INX     H                   ;ASSUMES @TRK=SECTOR+1 IE 44H
        INR     M
        ORA     A                   ;MAKE SURE TO RETURN NZ
        RET


ERR$NR: LXI     H,DRIVE$NR$ERR      ;"DRIVE NOT READY
        JMP     ABORT$ERR$MSG
ERR$MOV:
        LXI     H,BOOT$MOV$ERR      ;"ERROR READING BOOT/LOADER SECTORS"
        JMP     ABORT$ERR$MSG
ERR$LD1:
        LXI     H,BOOT$LD1$ERR      ;"DATA ERROR IN BOOT SECTOR"

ABORT$ERR$MSG:
        CALL    PRINT$STRING
        JMP     ZAPPLE              ;BACK TO START OF MONITOR.

;
;
;---------------------- ZFDC FDC BOOT & LOADER -----------------------------------

ZFDC$BOOT:                          ;CoMOV Boot with ZFDC FDC Board
        LXI     H,BOOT$MSG0         ;<<<<<<<<< BOOT FROM ZFDC BOARD >>>>>>>>>>>>>>
        CALL    PRINT$STRING        ;"Loading CPM from ZFDC FDC"
        LXI     H,ZFDC$MSG
        CALL    PRINT$STRING

        OUT     RESET$ZFDC$PORT             ;Do a hardware reset. Does not matter what is in [A]

        MVI     A,STATUS$DELAY      ;~0.5 second at 10 MHz
        LXI     B,0                 ;Delay to allow board to setup hardware
WAIT$D: DCR     B
        JNZ     WAIT$D              ;Delay for ~0.5 seconds
        DCR     B                   ;Reset B to 0FFH
        DCR     C
        JNZ     WAIT$D
        DCR     A
        JNZ     WAIT$D

        IN      S100$DATA$B         ;Check the board is there
        CPI     CMD$HANDSHAKE       ;Make sure we get HANDSHAKE byte back
        JNZ     ERR$NR              ;If error, just abort

        MVI     A,CMD$HANDSHAKE             ;Send another byte just to be sure.
        OUT     S100$DATA$B         ;This clears up ints on ZFDC board
        CALL    WAIT$FOR$ACK        ;Wait to make sure all is well.
```

```
        ORA     A
        JNZ     ERR$NR                  ;If error, just abort

        MVI     C,CMD$SET$FORMAT        ;Send Set Disk Format to 8 Inch SSSD DISK
        CALL    S100OUT
        MVI     C,0                     ;Floppy Drive 0, (ZFDC Board expects a 0H, 1H, 2H or 3H)
        CALL    S100OUT
        MVI     C,STD8IBM               ;ZFDC Board expects a Disk Format Table Number (0,1,2...13H)
        CALL    S100OUT
        CALL    WAIT$FOR$ACK            ;Return Z (and NO$ERRORS$FLAG in [A]), or NZ with error # in [A]
        JNZ     ERR$NR                  ;If error, just abort

        MVI     C,CMD$SET$DRIVE                 ;Send a "Set Drive CMD" to ZFDC board
        CALL    S100OUT
        MVI     C,0                     ;Floppy Drive #, (ZFDC Board expects a 0H, 1H, 2H or 3H)
        CALL    S100OUT
        CALL    WAIT$FOR$ACK            ;Return Z (and NO$ERRORS$FLAG in [A]), or NZ with error # in [A]
        JNZ     ERR$NR                  ;If error, just abort

                                        ;Drive selected and ready to read sectors. Note this code
                                        ;is written to eb compatible with the boot loader for the
                                        ;Versafloppy-II disk controller as well.

        MVI     A,STDSDT                ;SETUP FOR SD
        STA     @COUNT                  ;STORE AS 26 SECTORS/TRACK

        XRA     A                       ;Setup Boot Sector read track
        STA     @TRK
        INR     A
        STA     @SCTR
        STA     @NREC                   ;read only 1 sector initially

        LXI     H,COLD
        SHLD    @TADDR

        CALL    ZFDC$MULTI$READ$SECTOR ;Actully we will only read one sector here
        JMP     BOOT$SEC$READ           ;JMP to same section as for Versafloppy boot


ZFDC$MULTI$READ$SECTOR:
        MVI     C,CMD$SET$TRACK                 ;Set Track
        CALL    S100OUT
        LDA     @TRK
        MOV     C,A
        CALL    S100OUT                 ;Send Selected track HEX number
        CALL    WAIT$FOR$ACK            ;Return Z (and NO$ERRORS$FLAG in [A]), or NZ with error # in [A]
        JNZ     ERR$NR                  ;If error, just abort

        MVI     C,CMD$SET$SECTOR        ;Set Sector # to side A (or for DS disks also side B)
        CALL    S100OUT
        LDA     @SCTR
        MOV     C,A
        CALL    S100OUT                 ;Send Selected sector HEX number
        CALL    WAIT$FOR$ACK            ;Return Z (and NO$ERRORS$FLAG in [A]), or NZ with error # in [A]
        JNZ     ERR$NR                  ;If error, just abort
```

```
        MVI     C,CMD$SEEK$TRACK        ;Later can let board do this
        CALL    S100OUT
        CALL    WAIT$FOR$ACK            ;Return Z (and NO$ERRORS$FLAG in [A]), or NZ with error # in [A]
        JNZ     ERR$NR                  ;If error, just abort

        MVI     C,CMD$RD$MULTI$SECTOR   ;Routine assumes required Drive Table,Drive,Side,Track, and sector are already sent to board
        CALL    S100OUT                 ;(Note [HL]-> Sector DMA address)
        LDA     @NREC                   ;How many sectors
        MOV     C,A
        CALL    S100OUT
        CALL    WAIT$FOR$ACK            ;Wait for NO$ERRORS$FLAG to come back
        JNZ     ERR$NR                  ;If error, just abort

        LHLD    @TADDR                  ;Set DMA address

MULTI$RD$SEC:
        PUSH    H
        LHLD    @SEC$SIZE               ;For CPM this will be 128 Byte sector(s)
        PUSH    H
        POP     D                       ;@DEC$SIZE -> DE
        POP     H
RD$SEC:CALL     S100IN                  ;Note potential to lockup here & below (but unlightly)
        MOV     M,A
        INX     H
        DCX     D
        MOV     A,E
        ORA     D
        JNZ     RD$SEC

        LDA     @NREC                   ;How many sectors of data worth
        DCR     A
        STA     @NREC
        JNZ     MULTI$RD$SEC            ;Are there more

        CALL    WAIT$FOR$ACK            ;Return Z (and NO$ERRORS$FLAG in [A]), or NZ with error # in [A]
        RET


S100OUT:
        IN      S100$STATUS$B           ;Send data to ZFDC output (arrive with character to be sent in C)
        ANI     80H                     ;Is ZFDC in output mode, if not wait  (For Z80 use BIT DIRECTION$BIT,A)
        JNZ     S100OUT
        IN      S100$STATUS$B           ;Send data to ZFDC output (arrive with character to be sent in C)
        ANI     02H                     ;Has previous (if any) character been read.  (For Z80 use BIT DATA$OUT$RDY,A)
        JZ      S100OUT                 ;Z if not yet ready
        MOV     A,C
        OUT     S100$DATA$B
        RET

S100STAT:
        IN      S100$STATUS$B           ;Check if ZFDC has any data for S-100 system
        ANI     01H                     ;Anything there ?   (For Z80 use DATA$IN$RDY,A)
        RZ                              ;Return 0 if nothing
        XRA     A
        DCR     A                       ;Return NZ, & 0FFH in A if something there
        RET
```

```
S100IN:
        IN      S100$STATUS$B           ;Check if ZFDC has any data for S-100 system
        ANI     80H                     ;Is ZFDC in output mode, if not wait  (For Z80 use BIT DIRECTION$BIT,A)
        JZ      S100IN                  ;If low then ZFDC board is still in input mode, wait
        IN      S100$STATUS$B           ;Check if ZFDC has any data for S-100 system
        ANI     01H                     ;Anything there ?   (For Z80 use DATA$IN$RDY,A)
        JZ      S100IN
        IN      S100$DATA$A             ;return with character in A
        RET

WAIT$FOR$ACK:                           ;Delay to wait for ZFDC to return data. There is a timeout of about 2 sec.
        PUSH    B                       ;This can be increased if you are displaying debugging info on the ZFDC
        PUSH    D                       ;HEX LED display.
        LXI     B,0
        MVI     E,STATUS$DELAY          ;Timeout, (about 2 seconds)
WAIT$1: IN      S100$STATUS$B           ;Check if ZFDC has any data for S-100 system
        ANI     80H                     ;Is ZFDC in output mode, if not wait  (For Z80 use BIT DIRECTION$BIT,A)
        JZ      WAIT$2                  ;if low then ZFDC is still in input mode
        CALL    S100STAT                ;Wait until ZFDC Board sends something
        JZ      WAIT$2
        CALL    S100IN                  ;Get returned Error # (Note this releases the SEND$DATA routine on the ZFDC board)
        CPI     NO$ERRORS$FLAG          ;Was SEND$OK/NO$ERRORS$FLAG sent back from ZFDC Board
        POP     D                       ;Balance up stack
        POP     B
        RET                             ;Return NZ if problem, Z if no problem
WAIT$2: DCR     B
        JNZ     WAIT$1                  ;Try for ~2 seconds
        DCR     B                       ;Reset B to 0FFH
        DCR     C
        JNZ     WAIT$1
        DCR     B                       ;Reset B to 0FFH
        DCR     C
        DCR     E
        JNZ     WAIT$1
        XRA     A
        DCR     A
        POP     D                       ;Balance up stack
        POP     B
        RET                             ;Return NZ flag set if timeout AND 0FFH in [A]


;       LOAD A NUMBER OF SECTORS        ;Note this loader utilizes the fast multi-sec read in V2.8 of later

ZFDC$LOADER:                            ;CPM Loader with ZFDC FDC Board
        CALL    ZFDC$MULTI$READ$SECTOR ;Note the Boot sector has by now setup the sector count etc. in low RAM
        RET



;------THIS IS THE MAIN ROUTINE TO GET THE TIME DATA FROM THE CMOS-RTC Chip on the MSDOS Support Board

SHOW$TIME:
        LXI     H,TIME$MSG
        CALL    PRINT$STRING            ;Print message up to '$'
        CALL    PRINT$TIME
```

```
        RET

SHOW$DATE:
        LXI     H,DATE$MSG
        CALL    PRINT$STRING            ;Print message up to '$'
        CALL    PRINT$DATE
        RET


PRINT$TIME:
        CALL    UPD$IN$PR               ;CHECK FOR UPDATE IN PROCESS
        JNC     RTC$2A          ;GO AROUND IF OK
        JMP     RTC$ERROR               ;IF ERROR

RTC$2A: MVI     E,-2                    ;-2 goes to 0 for PORT$INC$2
        CALL    PORT$INC$2              ;SET ADDRESS OF SECONDS
        IN      CMOS$PORT+1             ;Get BCD value returned
        MOV     D,A                     ;SAVE IN D
        CALL    PORT$INC$2              ;SET ADDRESS OF MINUTES
        IN      CMOS$PORT+1             ;Get BCD value returned
        MOV     C,A                     ;SAVE IN C
        CALL    PORT$INC$2              ;SET ADDRESS OF HOURS
        IN      CMOS$PORT+1             ;Get BCD value returned
        MOV     B,A                     ;SAVE
        MVI     E,0                     ;SET E TO ZERO
        CALL    DisplayTime
        XRA     A                       ;Clear Carry
        RET                             ;BACK TO MONITOR

RTC$ERROR:                              ;Indicate RTC Board is not present or Error
        STC                             ;SET CARRY FOR ERROR
        RET

;Display time
;       Arrive with B = HOURS IN BCD
;                    C = Minutes in BCD
;                    D = Seconds in BCD
DisplayTime:
        PUSH    D
        PUSH    B
        MOV     A,B
        CALL    PRINT$BCD               ;Hours.  Convert BCD to ASCII
        MVI     C,':'
        CALL    ZCO
        POP     B
        MOV     A,C
        CALL    PRINT$BCD               ;Minutes.  Convert BCD to ASCII
        MVI     C,':'
        CALL    ZCO
        POP     D
        MOV     A,D
        CALL    PRINT$BCD               ;Seconds.  Convert BCD to ASCII
        RET

PRINT$DATE:
        CALL    UPD$IN$PR
```

```
        JNC     RTC$4A
        JMP     RTC$ERROR               ;IF ERROR

RTC$4A: MVI     E,6
        CALL    PORT$INC                ;POINT TO DAY
        IN      CMOS$PORT+1
        MOV     B,A                     ;SAVE IN A
        CALL    PORT$INC                ;POINT TO MONTH
        IN      CMOS$PORT+1
        MOV     D,A                     ;SAVE IN D
        CALL    PORT$INC                ;POINT TO YEAR
        IN      CMOS$PORT+1
        MOV     C,A                     ;SAVE IN C
        MVI     E,31H                   ;POINT TO CENTURY BYTE SAVE AREA
        CALL    PORT$INC                ;
        IN      CMOS$PORT+1             ;GET VALUE
        MOV     E,B                     ;GET DAY BACK
        MOV     B,A
        CALL    DisplayDate
        XRA     A                       ;Clear Carry
        RET                             ;FINISHED


;Display date
;       Return  B = CENTURY IN BCD
;               C = Year in BCD
;               D = Month in BCD
;               E = Day in BCD
DisplayDate:
        PUSH    D
        PUSH    D
        PUSH    B
        PUSH    B

        POP     B
        MOV     A,B
        CALL    PRINT$BCD               ;Century (19/20).  Convert BCD to ASCII
        POP     B
        MOV     A,C
        CALL    PRINT$BCD               ;Year.  Convert BCD to ASCII
        MVI     C,'/'
        CALL    ZCO
        POP     D
        MOV     A,D
        CALL    PRINT$BCD               ;Month.  Convert BCD to ASCII
        MVI     C,'/'
        CALL    ZCO
        POP     D
        MOV     A,E
        CALL    PRINT$BCD               ;Day.  Convert BCD to ASCII
        RET


UPD$IN$PR:                              ;Check we are ready to read clock
        PUSH    B
        LXI     B,600                   ;SET LOOP COUNT
```

```
UPDATE: MVI     A,0AH                   ;ADDRESS OF [A] REGISTER
        OUT     CMOS$PORT
        NOP
        NOP
        NOP
        IN      CMOS$PORT+1             ;READ  IN REGISTER [A]
        ANI     80H                     ;IF 8XH--> UIP BIT  IS ON (CANNOT READ TIME)
        JZ      UPD$IN$PREND            ;Are we ready/done
        DCX     B
        MOV     A,C
        ORA     B
        JNZ     UPDATE                  ;Try again
        XRA     A                       ;
        STC                             ;SET CARRY FOR ERROR
        POP     B
        RET
UPD$IN$PREND:
        XRA     A                       ;Clear Carry
        POP     B
        RET                             ;RETURN

PORT$INC:
        MOV     A,E
        INR     A                       ;INCREMENT ADDRESS
        MOV     E,A
        OUT     CMOS$PORT
        RET


PORT$INC$2:
        MOV     A,E
        ADI     2                       ;INCREMENT ADDRESS
        MOV     E,A
        OUT     CMOS$PORT
        RET


PRINT$BCD:                              ;Print BCD in [A]
        PUSH    PSW
        PUSH    PSW
        RAR
        RAR
        RAR
        RAR
        ANI     0FH
        ADI     30H
        MOV     C,A                     ;Write high byte mins to CRT
        CALL    ZCO
        POP     PSW
        ANI     0FH
        ADI     30H
        MOV     C,A
        CALL    ZCO
        POP     PSW
        RET
```

```
;>>>>>>>>>>>>>>>>>>>>>>>> SPEECH OUTPUT ROUTINES <<<<<<<<<<<<<<<<<<
;
;SPEAK OUTPUT (WILL BE USED TO COMMUNICATE WITH TALKER)
; Note the S100Computers I/O board V-Stamp speech chip will use the initial baud rate
; of of the SCC to communicate with it. This is determines after each reset/slave clear.

SPEAKER$CTS:                        ;Cannot get this to work. SCC does not change bit 5 of RR1
                                    ;when E1 sent to WR3 (No Auto Enable). See SCCINIT:
        IN      BCTL                ;A0H
        ANI     20H                 ;For Z80 use BIT 5,A
        MVI     A,0FFH
        RNZ                         ;Ret NZ if CTS is High
        XRA     A
        RET                         ;Ret Z if CTS is Low

SPEAKOUT:
        XRA     A                   ;Will try 256 times, then timeout
SPXXX:  PUSH    PSW
        IN      BCTL                ;(A0), Is SCC TX Buffer empty
        ANI     04H
        JNZ     SENDS               ;NZ if ready to recieve character
        POP     PSW
        DCR     A
        JNZ     SPXXX
        RET
SENDS:  POP     PSW
        MOV     A,C
        OUT     BDTA                ;(A2), Send it
        RET

;SPEAKTOMM THIS IS A ROUTINE TO SEND A STRING TO TALKER [HL] AT STRING
SPEAK$: MOV     A,M
        CPI     '$'
        JZ      SPEAK1
        MOV     C,A
        CALL    SPEAKOUT
        INX     H
        JMP     SPEAK$
SPEAK1: MVI     C,0DH               ;MUST END WITH A CR
        JMP     SPEAKOUT


;>>>>>>>>>>>>>>>>>> MODEM/SERIAL I/O <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; Note the S100Computers I/O board will have the SSC set initially to 19,200 Baud
;
SERIAL$OUT:
        XRA     A                   ;Will try 256 times, then timeout
MODXXX: PUSH    PSW
        IN      ACTL                ;MODEM/SERIAL OUT
        ANI     04H                 ;Are we ready for a character
        JNZ     SENDM               ;NZ if ready to recieve character
        POP     PSW
        DCR     A
        JNZ     MODXXX
        RET
SENDM:  POP     PSW
```

```
        MOV     A,C
        OUT     ADTA                    ;Send it
        RET

SERIAL$IN:
        XRA     A                       ;Will try 256 times, then timeout
SERIAL$INX:
        PUSH    PSW
        CALL    SERIAL$STAT             ;MODEN/SERIAL IN
        JNZ     GETMOD
        POP     PSW
        DCR     A
        JNZ     SERIAL$INX
        RET
GETMOD: POP     PSW
        IN      ADTA
        RET

SERIAL$STAT:
        IN      ACTL
        ANI     01H
        RZ                              ;Ret Z if nothing
        XRA     A
        DCR     A
        RET                             ;Ret FF/NZ if something

;
;
;THESE ARE ROUTINES NOT YET IMPLEMENTED
;
RI:                                     ;READER
POO:                                    ;PUNCH
PRDY:                                   ;PUNCH STATUS (Sent to Serial port right now)
RSTAT:                                  ;READER STATUS (Input from Serial port right now)
ONLIST:                                 ;ON LIST
OFLIST: RET                             ;OFF LIST
DOS:                                    ;Not done yet
TRAP:   HLT
        NOP

;
;
DRIVE$NR$ERR:   DB      BELL,CR,LF
                DB      'Drive not Ready.',CR,LF,LF,'$'
RESTORE$ERR:    DB      BELL,CR,LF
                DB      'Restore Failed.',CR,LF,LF,'$'
BOOT$MOV$ERR:   DB      BELL,CR,LF
                DB      'Read Error.',CR,LF,LF,'$'
SEEK$ERROR$MSG:         DB      BELL,CR,LF
                DB      'Seek Error.',CR,LF,LF,'$'

BOOT$LD1$ERR:   DB      BELL,CR,LF
                DB      'BOOT error.',CR,LF,LF,'$'
VF$HUNG:        DB      'VF Controller Hung',CR,LF,LF,'$'
BIOS$ERR:       DB      'BIOS JMP not in ROM',CR,LF,LF,'$'
BOOT$MSG0:      DB      CR,LF,'Loading CPM from $'
```

```
VF$MSG:         DB      'VF FDC.',CR,LF,'$'
ZFDC$MSG:       DB      'ZFDC FDC.',CR,LF,'$'

MENUMSG:        DB      CR,LF
                DB      'A=Memmap B=68000 C=ZFDC D=Disp E=Echo   F=Fill G=Goto'
                DB      CR,LF
                DB      'H=Date   I=Time  J=Test  K=Menu L=VERSA  M=Move P=SD-CARD'
                DB      CR,LF
                DB      'Q=Port   R=Ports S=Subs  T=Type V=Verify W=Port EDH'
                DB      CR,LF
                DB      'X=DOS(H) Z=Top   @=Flush Printer'
                DB      CR,LF,LF,'$'
;
MSG14:          DB      BELL,CR,LF
                DB      '8086/80286 Active'
                DB      CR,LF,LF,'$'
MSG68K:         DB      BELL,CR,LF
                DB      '68K Active'
                DB      CR,LF,LF,'$'
MSG17:          DB      CR,LF
                DB      'Segment (0-F):$'
TIME$MSG:       DB      CR,LF,'Time:- $'
DATE$MSG:       DB      CR,LF,'Date:- $'
GAP$MSG:        DB      '  $'

IDE$RW$ERROR:   DB      CR,LF
                DB      'IDE Drive R/W Error'
                DB      CR,LF,'$'
SP$MSG          DB      CR,LF,'SP=$'
SPEAKCPM$MSG:   DB      'LOADING CPM $'
SPEAKDOS$MSG:   DB      'LOADING DOS $'
CR$SMSG:        DB      CR,CR,CR,CR,'$'
;
;
;-------------- Code to relocate the ROM up to High RAM -----------------------------
;
;+++++++++++++++ MAKE SURE THIS CODE IS NOT OVERWRITTEN +++++++++++++++++++++++++++

                ORG     0FEB0H          ;So we always know where to jump to

MOVEROM:
        MVI     A,'#'                   ;For quick hardware diagnostic test
        OUT     CONSOL$OUT

        LXI     H,0                     ;Move total ROM from 0H up to F000H
        LXI     D,0F000H
MOVEX:  MOV     A,M
        STAX    D
        INX     H
        INX     D
        MOV     A,D
        CPI     10H                     ;1000H Bytes total
;       JNZ     MOVEX
        DB      0C2H,0BAH,0EH           ;We are operating in low RAM
        JMP     BEGIN                   ;JMP to start of ROM (now at F000H)
        HALT
```