```
;       MASTER.Z80  This is the main monitor program for my system.
;       It resided in 1 2732 PROM at F000H (or top half of 28C64)
;       Assemble and SLR's Z80ASM Assembler (Can also use Cromemco's Assembler)
;       Use:- Z80ASM MASTER FH
;
;       Note the monitor is is two sections. The F000H-F7FFH is for typical display
;       move memory type functions.  The second portion starts at F800H and contains
;       a series of CPM BIOS compatable 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 master
;
;       Master.HEX is written back to the same windows folder as altair.exe is in.

;       Programming an EEPROM for the Z80 Board with the VP-280 Programmer
;       Using a MK28C28A EEPROM or uP28C64:-
;       For monitor at F000H-FFFFH
;       Load Buffer Address - 1000
;       From File address F000H
;       This will put the code (4K) in the top “half” of the 8K EEPROM. It can be seen/edited at 1000H

;       Recent History...

;              26/2/09             Added ability to switch CO/CI to ACIA serial from PC.
;              5/3/09              Adjust RTS/CTS levels for Modem
;       V3.5   6/3/09              Set talker messages for new V-Stamp chip.
;              12/09/09            Add SD Systems IO-8 board Serial ports.
;       V3.52  16/9/09             Add SD Systems IO-8 Board clock display on signon
;       v3.6   21/9/09             Add display command for greater than 64K RAM, removed
;       V4.0   10/26/09            Switched in 8255 driven IDE HD Controller (Removed XComp)
;                                  some old commands to make more room.
;       V4.1   11/7/09             Added input ports scan/diagnostic
;       V4.2   11/14/09            Remove Date (keep time) from Clock (Chip is messed up by CPM3 routine)
;                                  also modified to switch from the SD System assembler to the SLR one.
;       V4.21  11/17/09            Removed 8086 jump far setting code
;       V4.3   11/18/09            Implement movement of 8086 Monitor code (EPROM) to correct location in RAM space
;       V4.31  11/19/09            Check 8086 Monitor ROM->ROM went OK. Added W command.
;       V4.32  12/7/09             Turn off any SD Systems 8024 video screen enhancements (flashing, underline etc).
;       V4.33  12/25/09            Correct High/Low byte Sector read for IDE board
;       V4.34  2/23/10             "O" command, 8086 Far jump to 500H (IF RAM @ FFFF0H), W command boots 8086 from reset at FFFF0H.
;       V4.35  3/25/10             "O" command just puts 8086 Far JMP to 500H (IF RAM @ FFFF0H). Done also at each reset.
;       V4.4   7/29/10             Removed all SD Systems IO-8. Added S-100Computers I/O board drivers.
;       V4.41  7/29/10             Initilization of V-Stamp chip done. Cleaned up Serial port names etc
```

```
;      V4.42  7/31/10            Switched RTC over to S-100Computers board (Ports A4,A5)
;      V4.50  2/7/11             Added Floppy Boot loader for ZFDC board. Still have the Versafloppy loader but no BIOS functions
;      V4.51  2/13/11            Check IDE if Boot sector is valid
;      V4.52  2/15/11            Pulse CF/IDE cards twice to reset (some) cards properly
;      V4.53  2/16/11            Initilize IDE board with IDE RD/WR lines inactive on power-up.
;      V4.54  2/28/11            Incoporated new fast multi-sector boot for CPM floppy loading with ZFDC board
;      V4.55  2/28/11            "O" command now jumps to SWITCH_8086 (activates 8086) when done
;      V4.55a 3/1/11             "O" cmd will just put 33 on Consol (temporary 8086 board test)
;      V4.56   3/15/11           Re-did IDE drive hardware reset pulse to one (delayed) pulse, then wait for drive ready status.
;      V4.57  6/3/11             Set up an equate for IDE drive reset pulse, Fixed Z command (Last version using MM58167 RTC chip)
;      V4.6   11/27/11           Switched to Dallas Semiconductor/IBM-PC CMOS-RTC chip & MSDOS Support board for time & dates
;      V4.7   3/26/12            Cleaned up IOBYTE options. Added 68000 CPU Slave activate option (B menu command)

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 COLD BOOT
STARTDOS EQU  100H                ;LOCATION WHERE MSDOS WILL BE PLACED FOR COLD BOOT
FFILE_SIZE EQU      9000h/512           ;SIZE OF 5MSDOS20.COM IN 512 BYTE SECTORS
;
;
;IOBYTE = SENSE SWITCHES AT PORT 0EFH
;
; BIT MAP OF PORT 0EFH:------- X X X X  X X X X      (11111111=NORMAL CONFIG)
;                           | | | |  | | | |..For Z80 Monitor,      0=CONSOLE DATA TO PRINTER ALSO
;                           | | | |  | | |....For 8086 Monitor,     0=Force Consol output to CGA/VGA Board
;                           | | | |  | |......For 8086 Monitor,     0=Do not initilize extra ROMS
;                           | | | |  |........Unused
;                           | | | |...........For CPM3,            0=Prevents LF's in CPM3
;                           | | |.............For Z80 Monitor,      0=Consol I/O via ACIA Serial port
;                           | |...............For CPM3,            0=Force format of Mdisk ith CPM3
;                           |.................For CPM3,            0=R/W protect Mdisk
;                                              For 8086 Monitor, 0=Prevent doing a JMPF to 0000:0500H after 8086 reset
;
```

```
;
;-------------- SD SYSTEMS VIDIO BOARD FOR CONSOLE INPUT & OUTPUT
CONSOL_STATUS EQU    0H
CONSOL_IN     EQU    01H
CONSOL_OUT    EQU    01H



;-------------- THIS IS MY PORT TO OUTPUT DATA TO HP 4050T LASAR PRINTER  (IMSAI 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         ;<--- Adjust as necessary,
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)
Z80PORT       EQU    0D0H                ;4 PORTS ON Z80 BOARD FOR MEMORY MANAGEMENT (& INT Controller on IA Z80 CPU Board)
SW86   EQU    0EDH                ;INPUT FROM THIS PORT SWITCHES IN THE 8088,8086, or 80286 board
SW68K  EQU    0ECH                ;INPUT FROM THIS PORT SWITCHES IN THE 68000 CPU Board


;-------------- 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
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 old 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                ;OLD @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                ;COLD 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 CPMLDR.COM Loader here with
                                  ;CPMLDR.COM will ALWAYS be on TRK 0,SEC2, (LBA Mode)
SEC_COUNT     EQU    12           ;CPMLDR.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 should 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 half of a 278C64)
```

```
       ORG    BASE                ;<--------<<<<<< LOCATION OF START OF MONITOR (First part)
VERSA  EQU    BASE+800H           ;<--------<<<<<< LOCATION OF FLOPPY BIOS (For old 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.

ZAPPLE:       JP     BEGIN               ;INITILIZATION
ZCI:   JP     CI                 ;CONSOL INPUT
ZRI:   JP     SERIAL_IN          ;READER      INPUT = Modem Input for Now
ZCO:   JP     CO                 ;CONSOL OUTPUT
ZPOO:  JP     SERIAL_OUT         ;PUNCH OUTPUT = Modem Output for Now
ZLO:   JP     LO                 ;LIST OUTPUT
ZCSTS: JP     CSTS               ;CONSOL STATUS
ZMEMCK:       JP     MEMSIZ              ;GET HIGHEST RAM RETURNS IT IN [HL]
ZTRAP: JP     TRAP               ;ERROR TRAP ADDRESS
ZSTART:       JP     START               ;JUMP TO MONITOR DO NOT RESET HARDWARE
ZTALK: JP     SPEAKOUT           ;SEND AN ASCII CHARACTER TO TALKER (One at a time)
ZTALKS:       JP     SPEAKER_CTS         ;STATUS FOR SPEECH CTS Line (V-Stamp CTS low when ready)
ZDELAY:       JP     DELAY               ;SOFTWARE DELAY LENGTH IN [A]
ZLSTAT:       JP     LSTAT               ;LIST STATUS
ZONLIST:JP    ONLIST             ;INITILIZE LIST DEVICE
ZOFFLIST:JP   OFLIST             ;TURN OFF LIST DEVICE
ZTIME: JP     PRINT_TIME         ;PUT TIME ON CRT @ CURSOR POSITION
ZDATE:JP      PRINT_DATE         ;PRINT DATE ON CRT @ CURSOR POSITION
ZSPEAK$:JP    SPEAK$             ;SEND ASCII STRING TO TALKER [HL] UP TO '$'
ZSERIAL_OUT:
       JP     SERIAL_OUT         ;OUT TO ZILOG SCC SERIAL PORT
ZSERIAL_IN:
       JP     SERIAL_IN          ;INPUT FROM ZILOG SCC SERIAL PORT
ZSERIAL_STAT:
       JP     SERIAL_STAT        ;STATUS FROM ZILOG SCC SERIAL PORT
ZLOADER:JP    LOADER             ;LOAD IN CPM IMAGE ON TRACKS 0 & 1 (VIA FLOPPY BOOT LOADER ON DISK SECTOR 1)
ZPMSG0:       JP     TOM                 ;DISPLAY STRING ON CONSOL [HL]=START ADD. [B]=LENGTH
ZPMSG$:       JP     PRINT_STRING        ;DISPLAY STRING ON CONSOL [HL]=START ADD. '$'=END
ZHLSP: JP     HLSP               ;DISPLAY [HL] ON CONSOL THEN ONE SPACE
ZBITS: JP     BITS1              ;DISPLAY 8 BITS OF [A] ON CONSOL
ZLBYTE:       JP     LBYTE               ;DISPLAY [A] ON CONSOL
ZHEXSP:       JP     HEXSP               ;PUT 16 BIT PARAMETERS ON STACK FROM CONSOL, [C]=PARAMETER #
ZCRLF: JP     CRLF               ;SEND CRLF TO CONSOL
ZHILO: JP     HILO               ;RANGE CHECK (INC [HL], IF HL=DE THEN SET CARRY)
ZCONV: JP     CONV               ;CONVERT HEX IN [A] TO ASCII IN [A]
ZDOS    JP    DOS                ;LOAD MSDOS FROM 5" DRIVE D:
ZPCHK: JP     PCHK               ;INPUT FROM CONSOL & TEST FOR DELIMITERS RET {Z} IF
                                 ;SPACE OR , RET {C} IF A CR ELSE NON ZERO NON CARRY
VFLOPPY       JP     VBOOT               ;BOOT UP CPM-80 FROM VERSAFLOPPY II FDC
ZHARD: JP     HBOOTCPM           ;BOOT UP CPM-80 FROM HARD DISK
```

```
ZPRDY: JP     PRDY                ;PUNCH READY CHECK
ZRSTAT:       JP     RSTAT               ;READER STATUS
ZCCHK: JP     CCHK                ;CHECK FOR ^S & ESC AT KEYBOARD
ZFLOPPY       JP     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  XMEMMAP   ; "N"  Display extended memory Segment:Address
       DW  UP8086    ; "O"  SWITCH CONTROL TO 8088, 8086 or 80286.
       DW  HBOOTCPM  ; "P"  BOOT IN CPM FROM IDE HARD DISK
       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: LD     A,'#'               ;For quick hardware diagnostic test
       OUT    (CONSOL_OUT),A
       LD     A,0FFH              ;Clear Printer strobe, comes up 0 on a reset
       OUT    (PRINTER_STROBE),A  ;also it turn all LED's off as a diagnostic
       LD     A,00000000B         ;FLAG PROGRESS VISUALLY FOR DIAGNOSTIC (ALL LED' ON)
       OUT    (DIAG_LEDS),A       ;LED's will go off one at a time

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

```
        LD      A,10000000B         ;FLAG PROGRESS VISUALLY FOR DIAGNOSTIC (1 LED off)
        OUT     (DIAG_LEDS),A

        LD      A,0FFH
        OUT     (RSET),A            ;RESET VERSAFLOPPY II FLOPPY DISK CONTROLLER (If Present)
        OUT     RESET_ZFDC_PORT,A   ;RESET ZFDC FLOPPY DISK CONTROLLER (If Present)

        XOR     A                   ;SET INTERUPT TO PAGE 0H
        OUT     (Z80PORT+1),A       ;KILL THE INTERSYSTEMS Z80 CPU BOARD INT CONTROLLER (If present)
        LD      I,A
                                    ;We need to clear the 8259A otherewise teh 8086 monitor sometimes hangs
        LD      A,MasterICW1        ;Initilize the 8259A PIC Controller (;EDGE triggered, 4 bytes, single Master,ICW4 needed)
        OUT     (MASTER_PIC_PORT),A
        LD      A,MasterICW2        ;Ints starts at 20H in RAM (IBM-PC uses 8X4 = 20H)
        OUT     (MASTER_PIC_PORT+1),A
        LD      A,MasterICW4        ;No slaves above, so 8259 does not expect ICW3
        out     (MASTER_PIC_PORT+1),A

        LD      A,11111111b         ;Allow no interrupts to 8259A with Z80.
        out     (MASTER_PIC_PORT+1),A

        LD      A,0H                ;SETUP MEMORY MANAGEMENT TO OVERLAP WITH
        OUT     (Z80PORT+2),A       ;CURRENT RAM in 64K Space
        LD      A,04H
        OUT     (Z80PORT+3),A

        LD      A,11000000B         ;FLAG PROGRESS VISUALLY FOR DIAGNOSTIC (2 LED's off)
        OUT     (DIAG_LEDS),A

ZAXXLE: LD      SP,AHEAD-4          ;SETUP A FAKE STACK
        JP      MEMSZ1              ;RETURNS WITH TOP OF RAM IN [HL]
        DW      AHEAD               ;Ret will pick up this address
AHEAD:  LD      SP,HL               ;[HL] CONTAINS TOP OF RAM - WORKAREA

        PUSH    HL
        POP     IX                  ;Store stack pointer for below in [IX]

        LD      HL,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

        LD      A,11100000B         ;FLAG PROGRESS (Have a Stack with 3 LED's off)
        OUT     (DIAG_LEDS),A

        CALL    PRINT_TIME          ;PRINT TIME ON CRT (IF RTC BOARD PRESENT)
        JP      C,NO_CLOCK
        LD      HL,GAP_MSG
        CALL    PRINT_STRING
        CALL    PRINT_DATE          ;PRINT DATE ON CRT, then CRLF
```

```
NO_CLOCK:
       CALL   CRLF

       LD     A,11110000B         ;FLAG PROGRESS (I/O board initilized, 4 LED's Off)
       OUT    (DIAG_LEDS),A

       LD     HL,SP_MSG           ;Print Current Stack Location
       CALL   PRINT_STRING


       PUSH   IX                  ;SP is stored from above in [IX]
       POP    HL
       CALL   HLSP                ;Print HL/SP
       CALL   CRLF                ;Then CRLF
       CALL   CSTS                ;CHECK IF GARBAGE AT KEYBOARD
       CALL   NZ,CI               ;If so flush it

       LD     A,11111000B         ;FLAG PROGRESS (Ready to go, 5 LED's off)
       OUT    (DIAG_LEDS),A

       LD     HL,CR_SMSG          ;lets V-Stamp chip get baud rate
       CALL   SPEAK$

       CALL   INITILIZE_IDE_BOARD ;initilize first IDE drive (if present)

       LD     A,11111100B         ;FLAG PROGRESS (Initilization done, 6 LED's off)
       OUT    (DIAG_LEDS),A


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

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

STARO: CALL   TI                  ;Main loop. Monitor will stay here until cmd.
       AND    7FH
       JR     Z,STARO
       SUB    '@'                 ;Commands @ to Z only
       RET    M
       CP     1BH                 ;A-Z only
       RET    NC
       ADD    A,A
       LD     HL,TBL
```

```
       ADD    A,L
       LD     L,A
       LD     A,(HL)
       INC    HL
       LD     H,(HL)
       LD     L,A
       LD     C,02H
       JP     (HL)                ;JUMP TO COMMAND TABLE
;
;----- GO CARRY OUT COMMAND AND POP BACK TO START----------
; NOTE STRING IS HERE IN CASE A 2716 IS USED BY MISTAKE (Monitor will at least signon)

MSG0:  DB SCROLL,QUIT,NO_ENHANCEMENT,FAST,BELL,CR,LF,LF
       DB 'Z80 ROM MONITOR V4.7 (John Monahan, 3/27/2012)  $'
SMSG:  DB 'HELLOW JOHN THE Z80 ROM MONITOR VERSION 4.7 IS NOW RESIDENT $'

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

TOM:   LD     C,(HL)
       INC    HL
       CALL   CO
       DJNZ   TOM
       RET
;
PRINT_STRING:
       LD     A,(HL)              ;A ROUTINE TO PRINT OUT A STRING @ [HL]
       INC    HL                  ;UP TO THE FIRST '$'.
       CP     '$'
       RET    Z
       LD     C,A
       CALL   CO
       JR     PRINT_STRING

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

CCHK:  CALL   CSTS                ;FIRST IS THERE ANYTHING THERE
       RET    Z
       CALL   CI
       CP     'S'-40H
       JR     NZ,CCHK1
CCHK2: CALL   CSTS                ;WAIT HERE UNTIL ANOTHER INPUT IS GIVEN
       JR     Z,CCHK2
CCHK1: CP     ESC
       RET    NZ                  ;RETURN EXECPT IF ESC

;RESTORE SYSTEM AFTER ERROR

ERROR: CALL   MEMSIZ              ;GET RAM AVAILABLE - WORKSPACE IN [HL]
       LD     SP,HL               ;SET STACK UP IN WORKSPACE AREA
       LD     C,'*'
```

```
       CALL   CO
       JP     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   HL
       PUSH   BC
       CALL   LADR
       LD     C,SPACE
       CALL   CO
       POP    BC
       POP    HL
       RET

;PRINT A SPACE

SF488: LD     C,SPACE
       JP     CO

;CONVERT HEX TO ASCII

CONV:  AND    0FH
       ADD    A,90H
       DAA
       ADC    A,40H
       DAA
       LD     C,A
       RET

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

EXLF:  CALL   HEXSP
       POP    DE
       POP    HL

;SEND TO CONSOL CR/LF

CRLF:  PUSH   BC
       LD     C,LF
       CALL   CO
       LD     C,CR
       CALL   CO
       POP    BC
       RET
```

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

EXPR3: INC    C                  ;ALREADY HAD [C]=2 FROM START
       CALL   HEXSP
       CALL   CRLF
       POP    BC
       POP    DE
       POP    HL
       RET

;GET ONE PARAMETER

EXPR1: LD     C,01H
HEXSP: LD     HL,0000
EX0:   CALL   TI
EX1:   LD     B,A
       CALL   NIBBLE
       JR     C,EX2X
       ADD    HL,HL
       ADD    HL,HL
       ADD    HL,HL
       ADD    HL,HL
       OR     L
       LD     L,A
       JR     EX0
EX2X:  EX     (SP),HL
       PUSH   HL
       LD     A,B
       CALL   QCHK
       JR     NC,SF560
       DEC    C
       RET    Z
SF560: JP     NZ,ERROR
       DEC    C
       JR     NZ,HEXSP
       RET
EXF:   LD     C,01H
       LD     HL,0000H
       JR     EX1

;RANGE TEST ROUTINE CARRY SET = RANGE EXCEEDED

HILOX: CALL   CCHK
       CALL   HILO
       RET    NC
       POP    DE                 ;DROP ONE LEVEL BACK TO START
       RET
HILO:  INC    HL                 ;RANGE CHECK SET CARRY IF [DE]=[HL]
       LD     A,H
```

```
       OR     L
       SCF
       RET    Z
       LD     A,E
       SUB    L
       LD     A,D
       SBC    A,H
       RET

;PRINT [HL] ON CONSOL

LADR:  LD     A,H
       CALL   LBYTE
       LD     A,L
LBYTE: PUSH   AF
       RRCA
       RRCA
       RRCA
       RRCA
       CALL   SF598
       POP    AF
SF598: CALL   CONV
       JP     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   BC                  ;SAVE [BC]
MEMSZ1:       LD     HL,0FFFFH           ;START FROM THE TOP DOWN
MEMSZ2:       LD     A,(HL)
       CPL
       LD     (HL),A
       CP     (HL)
       CPL                        ;PUT BACK WHAT WAS THERE
       LD     (HL),A
       JP     Z,GOTTOP
       DEC    H                   ;TRY 100H BYTES LOWER
       JR     MEMSZ2              ;KEEP LOOKING FOR RAM
GOTTOP:       POP    BC                  ;RESTORE [BC]
       RET

NIBBLE:       SUB    30H
       RET    C
       CP     17H
       CCF
       RET    C
       CP     LF
       CCF
       RET    NC
       SUB    07H
```

```
       CP     LF
       RET

COPCK: LD     C,'-'
       CALL   CO

PCHK:  CALL   TI

;TEST FOR DELIMITERS

QCHK:  CP     SPACE
       RET    Z
       CP     ','
       RET    Z
       CP     CR
       SCF
       RET    Z
       CCF
       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
       CP     CR
       RET    Z
       CP     'C'-40H                  ;^C TO BOOT IN CP/M
       JP     Z,FBOOT
       PUSH   BC
       LD     C,A
       CALL   CO
       LD     A,C
       POP    BC
       CP     40H                ;LC->UC
       RET    C
       CP     7BH
       RET    NC
SF754: AND    5FH
       RET

BITS1: PUSH   DE                  ;DISPLAY 8 BITS OF [A]
       PUSH   BC
       LD     E,A
       CALL   BITS
       POP    BC
       POP    DE
       RET
```

```
BITS:  LD     B,08H               ;DISPLAY 8 BITS OF [E]
       CALL   SF488
SF76E: SLA    E
       LD     A,18H
       ADC    A,A
       LD     C,A
       CALL   CO
       DJNZ   SF76E
       RET
;
;<<<<<<<<<<<<<<<<<<<<<< MAIN CONSOL OUTPUT ROUTINE >>>>>>>>>>>>>>>>>>>>>>>>

CO:    IN     A,(IOBYTE)          ;NOTE CHARACTER IS IN [C]
       BIT    0,A                 ;CHECK IF OUTPUT TO PRINTER IS ALSO REQ
       JP     Z,LOX
SDCONO:       IN     A,(CONSOL_STATUS)   ;SD SYSTEMS VIDIO BOARD PORT
       AND    4H
       JR     Z,SDCONO
       LD     A,C
       CP     07H                 ;IS IT A BELL
       JR     Z,BELL1
       CP     0H                  ;SD BOARD CANNOT TAKE A NULL!
       RET    Z
       OUT    (CONSOL_OUT),A
       IN     A,(IOBYTE)
       BIT    5,A                 ;SEE IF SERIAL PORT OUTPUT IS REQ
       JR     NZ,SDCON5           ;MAKE SURE TO RETURN CHARACTER SENT IN [A]
       CALL   SERIAL_OUT          ;Send data in [C] to Serial Port
SDCON5:       LD     A,C
       RET                        ;RETURN CHARACTER SENT IN [A]

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

BELL1: LD     A,06H               ;SEND A BELL
       OUT    (CONSOL_OUT),A
       LD     A,0FH
       CALL   DELAY
       LD     A,07H
       OUT    (CONSOL_OUT),A
       JR     SDCON5


DELAY: DEC    A                   ;GENERAL COUNT DOWN TIME DELAY
       RET    Z                   ;LENGTH SET IN [A]
       PUSH   AF
       LD     A,05H
MORE:  DEC    A
       PUSH   AF
```

```
       XOR    A
MORE2: DEC    A
       JR     NZ,MORE2
       POP    AF
       JR     NZ,MORE
       POP    AF
       JR     DELAY


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

CSTS:  IN     A,(CONSOL_STATUS)
       AND    02H
       JP     Z,TRYSER            ;See if input from Serial Port is req
       XOR    A
       DEC    A                   ;RETURN WITH 0FFH IN [A] IF SOMETHING
       RET
TRYSER:       IN     A,(IOBYTE)
       BIT    5,A                 ;SEE IF SERIAL PORT INPUT IS REQ
       JP     Z,SERIAL_STAT       ;Check if anything at Modem IN status routine
       XOR    A                   ;IF IOBYTE bit 1 then skip modem input
       RET                        ;RETURN WITH 0 IN A IF NOTHING THERE


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

CI:    IN     A,(CONSOL_STATUS)   ;NEED CONSTAT TO CLEAN UP SHIFT KEYS ETC
       AND    02H
       JR     Z,CHKSERIAL
       IN     A,(CONSOL_IN)
       AND    7FH
       RET
CHKSERIAL:
       IN     A,(IOBYTE)
       BIT    5,A                 ;SEE IF SERIAL PORT INPUT IS REQ
       JR     NZ,CI               ;NO, then do normal CI
       CALL   SERIAL_STAT         ;See if anything at Modem input
       JP     Z,CI
       JP     SERIAL_IN


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

LO:    LD     B,0FFH
LO2:   CALL   LSTAT
       JR     NZ,LO1
       DJNZ   LO2
       XOR    A
       LD     A,C
       RET                        ;RET Z if Printer problem (Not switched on)
```

```
LO1:   LD     A,0FFH              ;Setup strobe high to low then high
       OUT    (PRINTER_STROBE),A
       LD     A,C
       OUT    (PRINTER_OUT),A           ;Now Data
       XOR    A                   ;STROBE FOR CENTRONICS
       OUT    (PRINTER_STROBE),A
       LD     A,0FFH              ;Raise strobe again
       OUT    (PRINTER_STROBE),A
       OR     A,A
       RET                        ;Ret NZ if OK

FLUSH: LD     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     A,(PRINTER_STATUS)
       AND    00001111B           ;XXXX0110 IS READY (BIT 3=PAPER BIT 2=FAULT
       CP     00000110B           ;BIT 1=SELECT  BIT 0=BUSY
       JR     Z,LSTAT1
       XOR    A
       RET
LSTAT1:       XOR    A                  ;PUT 0FFH IN [A] IF READY & NO ZERO FLAG
       DEC    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    HL                  ;CLEAN UP STACK
       LD     HL,SPEAKCPM_MSG           ;Announce on speaker
       CALL   SPEAK$

       CALL   INITILIZE_IDE_BOARD ;Initilze the 8255 and drive (again just in case)

       LD     D,11100000B         ;Data for IDE SDH reg (512bytes, LBA mode,single drive)
       LD     E,REGshd            ;00001110,(0EH) CS0,A2,A1,
       CALL   IDEwr8D                   ;Write byte to select the MASTER device

       LD     B,0FFH              ;Delay time to allow a Hard Disk to get up to speed
WaitInit:
       LD     E,REGstatus         ;Get status after initilization
       CALL   IDErd8D                   ;Check Status (info in [D])
       BIT    7,D
       JR     Z,SECREAD           ;Zero, so all is OK to write to drive
```

```
                                ;Delay to allow drive to get up to speed
       PUSH   BC
       LD     BC,0FFFFH
DXLAY2:       LD     D,2               ;May need to adjust delay time to allow cold drive to
DXLAY1:       DEC    D                 ;to speed
       JR     NZ,DXLAY1
       DEC    BC
       LD     A,C
       OR     B
       JR     NZ,DXLAY2
       POP    BC
       DJNZ   WaitInit            ;If after 0FFH, 0FEH, 0FDH... 0, then drive initilization problem
IDError:
       LD     HL,DRIVE_NR_ERR           ;Drive not ready
       JP     ABORT_ERR_MSG

SECREAD:                        ;Note CPMLDR will ALWAYS be on TRK 0,SEC 1,Head 0
       LD     A,11111111B       ;FLAG PROGRESS VISUALLY FOR DIAGNOSTIC
       OUT    (DIAG_LEDS),A

       CALL   IDEwaitnotbusy            ;Make sure drive is ready
       JR     C,IDError          ;NC if ready

       LD     D,1                ;Load track 0,sec 1, head 0
       LD     E,REGsector        ;Send info to drive
       CALL   IDEwr8D

       LD     D,0                ;Send Low TRK#
       LD     E,REGcyLSB
       CALL   IDEwr8D

       LD     D,0                ;Send High TRK#
       LD     E,REGcyMSB
       CALL   IDEwr8D

       LD     D,SEC_COUNT        ;Count of CPM sectors we wish to read
       LD     E,REGcnt
       CALL   IDEwr8D

       LD     D,CMDread          ;Send read CMD
       LD     E,REGCMD
       CALL   IDEwr8D                   ;Send sec read CMD to drive.
       CALL   IDEwdrq                   ;Wait until it's got the data

       LD     HL,CPM_ADDRESS            ;DMA address where the CPMLDR resides in RAM
       LD     B,0                ;256X2 bytes
       LD     C,SEC_COUNT        ;Count of sectors X 512
MoreRD16:
       LD     A,REGdata          ;REG regsiter address
       OUT    (IDECport),A
```

```
        OR      IDErdline           ;08H+40H, Pulse RD line
        OUT     (IDECport),A

        IN      A,(IDEAport)        ;read the LOWER byte
        LD      (HL),A
        INC     HL
        IN      A,(IDEBport)        ;read the UPPER byte
        LD      (HL),A
        INC     HL

        LD      A,REGdata           ;Deassert RD line
        OUT     (IDECport),A
        DJNZ    MoreRD16
        DEC     C
        JR      NZ,MoreRD16

        LD      E,REGstatus         ;Check the R/W status when done
        CALL    IDErd8D
        BIT     0,D
        JR      NZ,IDEerr1          ;Z if no errors
        LD      HL,STARTCPM
        LD      A,(HL)
        CP      31H                 ;EXPECT TO HAVE 31H @80H IE. LD SP,80H
        JP      Z,STARTCPM          ;AS THE FIRST INSTRUCTION. IF OK JP to 100H in RAM
        JP      ERR_LD1                 ;Boot Sector Data incorrect

IDEerr1:
        LD      HL,IDE_RW_ERROR             ;Drive R/W Error
        JP      ABORT_ERR_MSG


;       ----- SUPPORT ROUTINES --------------

INITILIZE_IDE_BOARD:                    ;Drive Select in [A]. Note leaves selected drive as [A]
        LD      A,RDcfg8255         ;Config 8255 chip (10010010B), read mode on return
        OUT     (IDECtrl),A         ;Config 8255 chip, READ mode

                                    ;Hard reset the disk drive
                                    ;For some reason some CF cards need to the RESET line
                                    ;pulsed very carefully. You may need to play around
        LD      A,IDEreset          ;with the pulse length. Symptoms are: incorrect data comming
        OUT     (IDECport),A        ;back from a sector read (often due to the wrong sector being read)
                                    ;I have a (negative)pulse of 60 uSec. (10Mz Z80, two IO wait states).

        LD      C,IDE_Reset_Delay   ;~60 uS seems to work for the 5 different CF cards I have
ResetDelay:
        DEC     C
        JP      NZ,ResetDelay       ;Delay (reset pulse width)
        XOR     A
```

```
       OUT    (IDECport),A        ;No IDE control lines asserted (just bit 7 of port C)

       CALL   DELAY_15            ;Need to delay a little before checking busy status

IDEwaitnotbusy:                         ;Drive READY if 01000000
       LD     B,0FFH
       LD     C,080H              ;Delay, must be above 80H for 4MHz Z80. Leave longer for slower drives
MoreWait:
       LD     E,REGstatus         ;Wait for RDY bit to be set
       CALL   IDErd8D
       LD     A,D
       AND    11000000B
       XOR    01000000B
       JR     Z,DoneNotbusy
       DJNZ   MoreWait
       DEC    C
       JR     NZ,MoreWait
       SCF                        ;Set carry to indicate an error
       RET
DoneNotBusy:
       OR     A                   ;Clear carry it indicate no error
       RET



                                  ;Wait for the drive to be ready to transfer data.
IDEwdrq:                          ;Returns the drive's status in Acc
       LD     B,0FFH
       LD     C,0FFH              ;Delay, must be above 80H for 4MHz Z80. Leave longer for slower drives
MoreDRQ:
       LD     E,REGstatus         ;wait for DRQ bit to be set
       CALL   IDErd8D
       LD     A,D
       AND    10001000B
       CP     00001000B
       JR     Z,DoneDRQ
       DJNZ   MoreDRQ
       DEC    C
       JR     NZ,MoreDRQ
       SCF                        ;Set carry to indicate error
       RET
DoneDRQ:
       OR     A                   ;Clear carry
       RET
;
;---------------------------------------------------------------------
; Low Level 8 bit R/W to the drive controller.  These are the routines that talk
; directly to the drive controller registers, via the 8255 chip.
; Note the 16 bit I/O to the drive (which is only for SEC Read here) is done directly
; in the routine MoreRD16 for speed reasons.
```

```
IDErd8D:                        ;READ 8 bits from IDE register in [E], return info in [D]
       LD     A,E
       OUT    (IDECport),A      ;drive address onto control lines

       OR     IDErdline         ;RD pulse pin (40H)
       OUT    (IDECport),A      ;assert read pin

       IN     A,(IDEAport)
       LD     D,A               ;return with data in [D]

       LD     A,E               ;<---Ken Robbins suggestion
       OUT    (IDECport),A      ;Deassert RD pin

       XOR    A
       OUT    (IDECport),A      ;Zero all port C lines
       RET


IDEwr8D:                        ;WRITE Data in [D] to IDE register in [E]
       LD     A,WRcfg8255       ;Set 8255 to write mode
       OUT    (IDECtrl),A

       LD     A,D               ;Get data put it in 8255 A port
       OUT    (IDEAport),A

       LD     A,E               ;select IDE register
       OUT    (IDECport),A

       OR     IDEwrline         ;lower WR line
       OUT    (IDECport),A

       LD     A,E               ;<-- Kens Robbins suggestion, raise WR line
       OUT    (IDECport),A

       XOR    A                 ;Deselect all lines including WR line
       OUT    (IDECport),A

       LD     A,RDcfg8255       ;Config 8255 chip, read mode on return
       OUT    (IDECtrl),A
       RET

;-------------------------------------------------------------------------------

;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
       LD     HL,0
       LD     B,1
```

```
MAP1:  LD     E,'R'               ;PRINT R FOR RAM
       LD     A,(HL)
       CPL
       LD     (HL),A
       CP     (HL)
       CPL
       LD     (HL),A
       JR     NZ,MAP2
       CP     (HL)
       JR     Z,PRINT
MAP2:  LD     E,'p'
MAP3:  LD     A,0FFH
       CP     (HL)
       JR     NZ,PRINT
       INC    L
       XOR    A
       CP     L
       JR     NZ,MAP3
       LD     E,'.'
PRINT: LD     L,0
       DEC    B
       JR     NZ,NLINE
       LD     B,16
       CALL   ZCRLF
       CALL   HXOT4
NLINE: LD     A,SPACE
       CALL   OTA
       LD     A,E
       CALL   OTA
       INC    H
       JR     NZ,MAP1
       CALL   ZCRLF
       CALL   ZCRLF
       JP     ZSTART

;16 HEX OUTPUT ROUTINE

HXOT4: LD     C,H
       CALL   HXO2
       LD     C,L
HXO2:  LD     A,C
       RRA
       RRA
       RRA
       RRA
       CALL   HXO3
       LD     A,C
HXO3:  AND    0FH
       CP     10
       JR     C,HADJ
```

```
       ADD    A,7
HADJ:  ADD    A,30H
OTA:   PUSH   BC
       LD     C,A
       CALL   ZCO                ;SEND TO CONSOL
       POP    BC
       RET

;DISPLAY MEMORY IN HEX

DISP:  CALL   EXLF               ;GET PARAMETERS IN [HL],[DE]
       LD     A,L                ;ROUND OFF ADDRESSES TO XX00H
       AND    0F0H
       LD     L,A
       LD     A,E                ;FINAL ADDRESS LOWER HALF
       AND    0F0H
       ADD    A,10H              ;FINISH TO END 0F LINE
SF172: CALL   LFADR
SF175: CALL   BLANK
       LD     A,(HL)
       CALL   ZLBYTE
       CALL   HILOX
       LD     A,L
       AND    0FH
       JR     NZ,SF175
       LD     C,TAB              ;INSERT A TAB BETWEEN DATA
       CALL   ZCO
       LD     B,4H               ;ALSO 4 SPACES
TA11:  LD     C,SPACE
       CALL   ZCO
       DJNZ   TA11
       LD     B,16               ;NOW PRINT ASCII (16 CHARACTERS)
       PUSH   DE                 ;TEMPORLY SAVE [DE]
       LD     DE,0010H
       SBC    HL,DE
       POP    DE
T11:   LD     A,(HL)
       AND    7FH
       CP     ' '                ;FILTER OUT CONTROL CHARACTERS'
       JR     NC,T33
T22:   LD     A,'.'
T33:   CP     07CH
       JR     NC,T22
       LD     C,A                ;SET UP TO SEND
       CALL   ZCO
       INC    HL
       DJNZ   T11                ;REPEAT FOR WHOLE LINE
       JR     SF172

BLANK: LD     C,' '
```

```
       JP     ZCO

;INSPECT AND / OR MODIFY MEMORY

SUBS:  LD     C,1
       CALL   ZHEXSP
       POP    HL
SF2E3: LD     A,(HL)
       CALL   ZLBYTE
       LD     C,'-'
       CALL   ZCO
       CALL   ZPCHK
       RET    C
       JR     Z,SF2FC
       CP     5FH
       JR     Z,SF305
       PUSH   HL
       CALL   EXF
       POP    DE
       POP    HL
       LD     (HL),E
       LD     A,B
       CP     CR
       RET    Z
SF2FC: INC    HL
SF2FD: LD     A,L
       AND    07H
       CALL   Z,LFADR
       JR     SF2E3
SF305: DEC    HL
       JR     SF2FD

;FILL A BLOCK OF MEMORY WITH A VALUE

FILL:  CALL   EXPR3
SF1A5: LD     (HL),C
       CALL   HILOX
       JR     NC,SF1A5
       POP    DE
       JP     ZSTART

;GO TO A RAM LOCATION

GOTO:  LD     C,1                 ;SIMPLE GOTO FIRST GET PARMS.
       CALL   HEXSP
       CALL   CRLF
       POP    HL                  ;GET PARAMETER PUSHED BY EXF
       JP     (HL)

; GET OR OUTPUT TO A PORT
```

```
QUERY: CALL   ZPCHK
       CP     'O'                ;OUTPUT TO PORT
       JR     Z,SF77A
       CP     'I'                ;INPUT FROM PORT
       JP     Z,QQQ1
       LD     C,'*'
       JP     ZCO                ;WILL ABORT IF NOT 'I' OR 'O'
QQQ1:  LD     C,1
       CALL   ZHEXSP
       POP    BC
       IN     A,(C)
       JP     ZBITS
;
SF77A: CALL   ZHEXSP
       POP    DE
       POP    BC
       OUT    (C),E
       RET


; MEMORY TEST

RAMTEST:CALL  EXLF
SF200: LD     A,(HL)
       LD     B,A
       CPL
       LD     (HL),A
       XOR    (HL)
       JR     Z,SF215
       PUSH   DE
       LD     D,B
       LD     E,A                ;TEMP STORE BITS
       CALL   ZHLSP
       CALL   BLANK
       LD     A,E
       CALL   ZBITS
       CALL   ZCRLF
       LD     B,D
       POP    DE
SF215: LD     (HL),B
       CALL   HILOX
       JR     SF200

;MOVE A BLOCK OF MEMORY TO ANOTHER LOCATION

MOVE:  CALL   EXPR3
SF21E: LD     A,(HL)
       LD     (BC),A
       INC    BC
```

```
       CALL   HILOX
       JR     SF21E

;VERIFY ONE BLOCK OF MEMORY WITH ANOTHER

VERIFY:       CALL   EXPR3
VERIO: LD     A,(BC)
       CP     (HL)
       JR     Z,SF78E
       PUSH   BC
       CALL   CERR
       POP    BC
SF78E: INC    BC
       CALL   HILOX
       JR     VERIO
       RET
;
CERR:  LD     B,A
       CALL   ZHLSP
       LD     A,(HL)
       CALL   ZLBYTE
       CALL   BLANK
       LD     A,B
       CALL   ZLBYTE
       JP     ZCRLF

ECHO:  CALL   CI                  ;Routeen to check keyboard etc.
       CP     'C'-40H                   ;Loop until  ^C
       RET    Z
       CP     'Z'-40H
       RET    Z
       LD     C,A
       CALL   CO
       JR     ECHO


;Display Extended memory map for 1MG RAM using IA-2 Z80 Board window registers

XMEMMAP:
       LD     HL,MSG17           ;Get segment (0-F)
       CALL   PRINT_STRING
       LD     C,1
       CALL   ZHEXSP             ;Get 2 or 4 hex digits (count in C).
       POP    HL
       LD     A,L                ;Get single byte value
       AND    0FH
       EXX
       LD     D,A                ;Store in D' for 000X:YYYY display below
       SLA    A
       SLA    A
```

```
       SLA    A
       SLA    A
       OUT    (Z80PORT+2),A       ;Re-map to first 16K in segment:64K Space
       LD     E,A                 ;store shifted nibble in E'
       LD     HL,0                ;Will store 0-FFFF for total RAM display (not actual access)
       EXX
       LD     D,0                 ;Total display line count (256 characters, 16lines X 16 characters)

       CALL   ZCRLF
       LD     HL,0
       LD     B,1
XMAP1: LD     A,H
       AND    00111111B           ;Wrap 16K window
       LD     H,A
       LD     E,'R'               ;PRINT R FOR RAM
       LD     A,(HL)
       CPL
       LD     (HL),A
       CP     (HL)
       CPL
       LD     (HL),A              ;Save it back
       JR     NZ,XMAP2
       CP     (HL)
       JR     Z,XPRINT
XMAP2: LD     E,'p'
XMAP3: LD     A,0FFH
       CP     (HL)
       JR     NZ,XPRINT
       INC    L
       XOR    A
       CP     L
       JR     NZ,XMAP3
       LD     E,'.'
XPRINT:       LD     L,0
       DEC    B
       JR     NZ,XNLINE
       LD     B,16
       CALL   ZCRLF
       CALL   SET_WINDOW
       LD     A,SPACE
       JR     XN11
XNLINE:       LD     A,SPACE
       CALL   OTA
       LD     A,E
XN11:  CALL   OTA
       INC    H
       INC    D                   ;Are we done yet
       JR     NZ,XMAP1
       CALL   ZCRLF
       XOR    A
```

```
       OUT    (Z80PORT+2),A       ;Set RAM window back to the way it was
       JP     ZSTART

SET_WINDOW:                       ;Setup the unique IA-II Z80 board window to address > 64k
       EXX
       LD     C,D                 ;Print seg value
       CALL   HXO2
       LD     C,':'
       CALL   CO
       CALL   HXOT4               ;Print HL' (not origional HL)

       LD     A,H                 ;get current H being displayed (Already pointed to first 16K window)
NOTW0: CP     40H
       JR     NZ,NOTW1
       LD     A,E
       ADD    A,04H               ;Window for 4,5,6,7, set to H from above
       JR     DOWIN
NOTW1: CP     80H
       JR     NZ,NOTW2
       LD     A,E
       ADD    A,08H               ;Window for 8,9,A,B set to H from above
       JR     DOWIN
NOTW2: CP     0C0H
       JR     NZ,NOTW3            ;Must be values in between
       LD     A,E
       ADD    A,0CH               ;Window for 4,5,6,7, set to H from above
DOWIN: OUT    (Z80PORT+2),A       ;Re-map to first 16K in segment:64K Space
NOTW3: LD     A,H
       ADD    A,10H
       LD     H,A
       EXX                        ;Get back normal register set
       RET


;Place an 8086 a Far Jump at F000:FFF0H (FFFF0H) to 500H in RAM for the 8086/80286
;If there is a ROM there nothing will change and the 8086 reset/boot will jump
;from F000:FFF0 to the start or the ROM monitor at F000:FC00H.  If however
;no ROM is present the 8086 will find the RAM code below and jump to 500H in RAM
;Whatever is at that location will then run - usually CPM86.

UP8086:       LD     A,0FCH              ;Point to 8086 Reset location
       OUT    (Z80PORT+2),A       ;Re-map to 0000H to FC000H
       LD     HL,3FF0H
       LD     (HL),0EAH
       INC    HL
       LD     (HL),0H
       INC    HL
       LD     (HL),05H
       INC    HL
       LD     (HL),0H
```

```
       INC    HL
       LD     (HL),0H
       INC    HL
       LD     (HL),0F4H           ;Put an 8086 HLT here just in case

;      LD     (HL),0B0H           ;Continously put "3" on Consol via port 01
;      INC    HL                  ;Basic test for 8086 on reset
;      LD     (HL),33H
;      INC    HL
;      LD     (HL),0E6H
;      INC    HL
;      LD     (HL),01H
;      INC    HL
;      LD     (HL),0EBH
;      INC    HL
;      LD     (HL),0FAH

       XOR    A
       OUT    (Z80PORT+2),A       ;Re-map back to 0H
       JP     SWITCH_8086               ;Switch over control to the 8086


;READ ASCII FROM MEMORY

TYPE:  CALL   EXLF
SF30B: CALL   LFADR
       LD     B,56
SF310: LD     A,(HL)
       AND    7FH
       CP     SPACE
       JR     NC,SF319
SF317: LD     A,2EH
SF319: CP     7CH
       JR     NC,SF317
       LD     C,A
       CALL   ZCO
       CALL   HILOX
       DJNZ   SF310
       JR     SF30B


;      Display all active IO inputports in the system
;
INPORTS:CALL  ZCRLF
       LD     B,0                 ;Now loop through all ports (0-FF)
       LD     D,6                 ;Display 6 ports across
       LD     E,0FFH              ;Will contain port number
LOOPIO:       LD     C,E
       LD     A,E
       CP     A,SW86              ;Inputting here will switch out the Z80 to 8086/80286
```

```
       JR     Z,SKIP
       CP     A,SW68K                  ;Also this one (68K)
       JR     Z,SKIP
;
       IN     A,(C)               ;Remember  [ZASMB does not work with this opcode,SLR is OK]
       CP     A,0FFH              ;No need for 0FF's
       JR     Z,SKIP
       LD     H,A                 ;store port data in H for below
       LD     A,E                 ;Need to print port # first
       CALL   LBYTE               ;Print port number
       LD     C,'-'
       CALL   ZCO
       LD     C,'>'
       CALL   ZCO
       LD     A,H                 ;get back port data
       CALL   LBYTE               ;print it
       LD     C,TAB
       CALL   ZCO
       DEC    D                   ;6 ports per line
       JR     NZ,SKIP
       LD     D,6
       CALL   ZCRLF
SKIP:  DEC    E                   ;Next Port
       DJNZ   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
       LD     A,AinBout8255cfg    ;A input, B output, C(bits 0-3) output, (bits 4-7)input
       OUT    (PortCtrl_8255),A   ;Config 8255 chip, Mode 0

                                  ;Then the SCC
       LD     A,ACTL              ;Program Channel A
       LD     C,A
       LD     B,0EH               ;Byte count for OTIR below
       LD     HL,SCCINIT
       OTIR
;
       LD     A,BCTL              ;Program Channel B
       LD     C,A
       LD     B,0EH               ;Byte count for OTIR below
       LD     HL,SCCINIT
       OTIR
       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 <<<<<<<<<<<
;      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


NOP
NOP
NOP
;--------------------------------------------------------------------------------
       ORG    VERSA               ;<--------- THIS LOCATION MUST NOT BE CHANGED (F800H)
                                  ;My old 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 JP's for old CPM BIOS'es. Only LOADER is now functional.

FBOOT: JP     BOOT                ;COLD START ENTRY
WBOOT: JP     BIOS_JP_ERR         ;WARM START ENTRY
CSE:   JP     ZCSTS               ;CONSOLE STATUS
CIE:   JP     ZCI                 ;CONSOLE IN
COE:   JP     ZCO                 ;CONSOLE OUT
LIST:  JP     ZLO                 ;TO MONITOR FOR PRINTER
PUNCH: JP     ZPOO                ;TO MONITOR FOR PUNCH
```

```
READR: JP     ZRI                 ;TO MONITOR FOR READER
HME:   JP     BIOS_JP_ERR         ;HOME         ;MOVE TO TRACK 0
SDSKE: JP     BIOS_JP_ERR         ;SELDSK
S@TRKE:       JP     BIOS_JP_ERR         ;SET@TRK
SSECE: JP     BIOS_JP_ERR         ;SETSEC
SDMAE: JP     BIOS_JP_ERR         ;SETDMA
RDE:   JP     BIOS_JP_ERR         ;READF
WRE:   JP     BIOS_JP_ERR         ;WRITEF
LISTS: JP     LSTAT               ;LIST STATUS
SECTR: JP     BIOS_JP_ERR         ;SECTRAN FOR 2.2 SECTOR TRANSLATION TABLE

DTYPE: JP     BIOS_JP_ERR         ;UNITSL       SET UP @UNIT BYTE (DISK DENSITY)
SVE:   JP     BIOS_JP_ERR         ;SAVER SAVE N RECORDS
LDE:   JP     LOADER              ;LOADER       LOAD N SECTORS FROM TRACK 0 (& TRACK 1)

BIOS_JP_ERR:
       LD     HL,BIOS_ERR         ;"BIOS JMP longer implemented in ROM @ F800H."
       JP     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: XOR    A,A                 ;0 = Flag as Boot from Versafloppy II FDC
       JR     BOOT_COLD
ZBOOT: XOR    A,A
       DEC    A                   ;0FFH = Flag as Boot from ZFDC FDC
BOOT_COLD:
       LD     (@FDCTYPE),A        ;0 for VF, 0FFH for ZFDC

BOOT:  LD     A,11111111B         ;FLAG PROGRESS VISUALLY FOR DIAGNOSTIC
       OUT    (DIAG_LEDS),A

       LD     HL,SPEAKCPM_MSG            ;Announce on speaker
       CALL   SPEAK$
       XOR    A
       LD     (CDISK),A           ;MAKE CURRENT DISK A:
       LD     (CIOBYTE),A         ;CLEANUP IOBYTE
       LD     (@UNIT),A           ;8LOAD.Z80 (The first sector loader module) will count on this being 0H
                                  ;for the Versafloppy-II boots
       LD     (ZERO_L),A          ;These need to be zero's here for the CPM Loader/Versafloppy-II of my old
       LD     (ZERO_H),A          ;NON-BANKED CPM3 or CPM2.2 disks. Need to later find out why!
       LD     HL,128              ;Assume 128 byte sectors for 8" disk
       LD     (@SEC_SIZE),HL
```

```
BOOTW1:       LD     SP,@SSTACK
       LD     A,(@FDCTYPE)        ;Are we using a Versafloppy II or ZFDC FDC board
       OR     A,A
       JP     NZ,ZFDC_BOOT        ;<<<<<<<<< Go to ZFDC Board BOOT >>>>>>>>>>>>>

VFDC_BOOT:
       LD     HL,BOOT_MSG0        ;<<<<<<<<< BOOT FROM VERSAFLOPPY-II >>>>>>>>>>>>
       CALL   PRINT_STRING              ;"Loading CPM from VF FDC"
       LD     HL,VF_MSG
       CALL   PRINT_STRING

       LD     A,0D0H              ;FORCE CHIP INTERUPT
       OUT    (CMD),A

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

       XOR    A
       LD     (@TRK),A
       INC    A
       LD     (@SCTR),A

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

       LD     HL,COLD
       LD     (@TADDR),HL

       CALL   VF_READ_SECTOR             ;Read the Boot Sector
BOOT_SEC_READ:
       JP     NZ,ERR_LD
BOOT_SEC_CHECK:
       LD     HL,COLD
       LD     A,(HL)
       CP     31H                 ;EXPECT TO HAVE 31H @80H IE. LD SP,80H
       JP     Z,COLD              ;AS THE FIRST INSTRUCTION. IF OK JP 80H
       JP     ERR_LD1                   ;Boot Sector Data incorrect


VF_READ_SECTOR:                          ;READ SECTOR COMMAND
       LD     B,3                 ;Will Try 3 times
READ1: PUSH   BC
       CALL   DRINIT              ;Setup sector paramaters
       LD     A,E
       CP     A,80H               ;128 or 512 byte sectors ?
```

```
        LD      B,128
        DI
        LD      A,RDCMD
        OUT     (CMD),A                 ;Note wait states are now switched on

        JR      M2
M2:     JR      MM2
MM2:    JR      Z,RD_128
        LD      B,0                     ;256X2
        INIR                            ;[C]-> [HL++],[B--]
RD_128:         INIR

        EI
        CALL    WAITF                   ;Wait states are now off
        IN      A,(STATUS)
        AND     A,SRMASK                ;Check sector was read OK
        POP     BC
        RET     Z
        DEC     B
        JR      NZ,READ1
        XOR     A,A
        DEC     A
        RET                             ;Return NZ if failure after 3 reads


DRINIT:         CALL   SEEK                     ;DRIVE INITIALIZATION
        LD      HL,(@TADDR)             ;SETUP DMA ADDRESS AND BYTE COUNT
        LD      A,(@SCTR)
        OUT     (SECTOR),A

        LD      DE,(@SEC_SIZE)                  ;This will be 128 or 512 sectors
        LD      C,DATA                  ;8067H in BC

SWEB:   IN      A,(SELECT)              ;ENABLE WAIT STATES
        AND     7FH
        OUT     (SELECT),A
        RET


;       SEEK TRACK
SEEK:   LD      A,(@TRK)
        LD      C,A
        IN      A,(TRACK)
        CP      C
        RET     Z                       ;IF SAME TRACK NO NEED TO SEEK

        LD      A,(@TRK)
        OUT     (DATA),A
        CALL    READY_CHK               ;Critical to make sure chip is ready first!
        LD      A,FSKCMD                ;Send Seeek Command to WD1791
```

```
       OUT    (CMD),A
       CALL   DELAY_15            ;Delay ~15ms
       CALL   READY_CHK
       IN     A,(TRACK)
       LD     C,A
       LD     A,(@TRK)
       CP     A,C
       RET    Z
       LD     HL,SEEK_ERROR_MSG
       JP     ABORT_ERR_MSG


READY_CHK:
       LD     BC,0
READY_CHK1:
       IN     A,(STATUS)
       AND    A,1
       RET    Z
       DEC    BC
       LD     A,C
       OR     A,B
       JP     NZ,READY_CHK1       ;Wait until 1791/5 is ready
       JP     WAIT3

WAITF: LD     E,0
       PUSH   BC
       LD     C,2
WAIT2: IN     A,(STATUS)
       AND    1
       JR     Z,DWAIT
       DJNZ   WAIT2
       DEC    E
       JR     NZ,WAIT2
       DEC    C
       JR     NZ,WAIT2
       POP    BC
WAIT3: IN     A,(SELECT)          ;IF BY THIS TIME NOT READY FORCE
       OR     80H                 ;A HARDWARE RESET
       OUT    (RSET),A
       LD     HL,VF_HUNG
       JP     ABORT_ERR_MSG


;      DISABLE WAIT STATES
DWAIT: POP    BC                  ;TO BALANCE THE ABOVE PUSH IN WAIT
DDWAIT:       IN     A,(SELECT)
       OR     80H
       OUT    (SELECT),A
       RET
```

```
DELAY_15:                       ;DELAY ~15 MS
       LD     A,40
DELAY1:        LD      B,0
M0:    DJNZ    M0
       DEC     A
       JR      NZ,DELAY1
       RET

DELAY_150:                      ;DELAY ~150 MS
       LD     C,10
DELAY320A:
       CALL    DELAY_15
       DEC     C
       JP      NZ,DELAY320A
       RET

LOADER:        LD      A,(@FDCTYPE)        ;Are we using a Versafloppy II or ZFDC FDC board
       OR      A,A
       JP      NZ,ZFDC_LOADER              ;Go to ZFDC Board Loader


;      LOAD A NUMBER OF SECTORS
VF_LOADER:
       CALL    VF_READ_SECTOR
       JP      NZ,ERR_LD
       LD      C,'.'                ;Show progress
       CALL    CO
       CALL    INCP
       JR      NZ,VF_LOADER
       RET

;      INC SECTOR AND TRACK
INCP:  LD      HL,(@TADDR)
       LD      DE,(@SEC_SIZE)              ;128 or 512 byte sectors
INCP2: ADD     HL,DE
       LD      (@TADDR),HL
       LD      HL,@NREC
       DEC     (HL)
       RET     Z                    ;Return when we have done all sectors (~51)
       LD      HL,@SCTR
       INC     (HL)
       LD      A,(@COUNT)           ;IS ONE TRACK DONE YET (Sec/track+1)
       INC     A
       CP      (HL)
       RET     NZ                   ;IF FULL Z, THEN GO TO NEXT TRACK
       LD      (HL),1               ;SET SECTOR COUNT BACK TO 1
       INC     HL                   ;ASSUMES @TRK=SECTOR+1 IE 44H
       INC     (HL)
       OR      A                    ;MAKE SURE TO RETURN NZ
       RET
```

```
ERR_NR:       LD     HL,DRIVE_NR_ERR            ;"DRIVE NOT READY
       JP     ABORT_ERR_MSG
ERR_LD:       LD     HL,BOOT_LD_ERR             ;"ERROR READING BOOT/LOADER SECTORS"
       JP     ABORT_ERR_MSG
ERR_LD1:LD    HL,BOOT_LD1_ERR            ;"DATA ERROR IN BOOT SECTOR"

ABORT_ERR_MSG:
       CALL   PRINT_STRING
       JP     ZAPPLE              ;BACK TO START OF MONITOR.


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

ZFDC_BOOT:                         ;Cold Boot with ZFDC FDC Board
       LD     HL,BOOT_MSG0        ;<<<<<<<<< BOOT FROM ZFDC BOARD >>>>>>>>>>>>>>
       CALL   PRINT_STRING        ;"Loading CPM from ZFDC FDC"
       LD     HL,ZFDC_MSG
       CALL   PRINT_STRING

       OUT    RESET_ZFDC_PORT,A   ;Do a hardware reset. Does not matter what is in [A]

       LD     A,STATUS_DELAY             ;~0.5 second at 10 MHz
       LD     BC,0                ;Delay to allow board to setup hardware
WAIT_D:       DEC    B
       JR     NZ,WAIT_D           ;Delay for ~0.5 seconds
       DEC    B                   ;Reset B to 0FFH
       DEC    C
       JR     NZ,WAIT_D
       DEC    A
       JR     NZ,WAIT_D

       IN     A,S100_DATA_B       ;Check the board is there
       CP     A,CMD_HANDSHAKE            ;Make sure we get HANDSHAKE byte back
       JP     NZ,ERR_NR           ;If error, just abort

       LD     A,CMD_HANDSHAKE            ;Send another byte just to be sure.
       OUT    S100_DATA_B,A       ;This clears up ints on ZFDC board
       CALL   WAIT_FOR_ACK        ;Wait to make sure all is well.
       OR     A,A
       JP     NZ,ERR_NR           ;If error, just abort

       LD     C,CMD_SET_FORMAT    ;Send Set Disk Format to 8" SSSD DISK
       CALL   S100OUT
       LD     C,0                 ;Floppy Drive 0, (ZFDC Board expects a 0H, 1H, 2H or 3H)
       CALL   S100OUT
       LD     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]
```

```
        JP      NZ,ERR_NR           ;If error, just abort

        LD      C,CMD_SET_DRIVE             ;Send a "Set Drive CMD" to ZFDC board
        CALL    S100OUT
        LD      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]
        JP      NZ,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.

        LD      A,STDSDT            ;SETUP FOR SD
        LD      (@COUNT),A          ;STORE AS 26 SECTORS/TRACK

        XOR     A                   ;Setup Boot Sector read track
        LD      (@TRK),A
        INC     A
        LD      (@SCTR),A
        LD      (@NREC),A           ;read only 1 sector initially

        LD      HL,COLD
        LD      (@TADDR),HL

        CALL    ZFDC_MULTI_READ_SECTOR      ;Actully we will only read one sector here
        JP      BOOT_SEC_READ       ;JMP to same section as for Versafloppy boot


ZFDC_MULTI_READ_SECTOR:
        LD      C,CMD_SET_TRACK             ;Set Track
        CALL    S100OUT
        LD      A,(@TRK)
        LD      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]
        JP      NZ,ERR_NR           ;If error, just abort

        LD      C,CMD_SET_SECTOR    ;Set Sector # to side A (or for DS disks also side B)
        CALL    S100OUT
        LD      A,(@SCTR)
        LD      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]
        JP      NZ,ERR_NR           ;If error, just abort

        LD      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]
        JP      NZ,ERR_NR           ;If error, just abort
```

```
       LD     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)
       LD     A,(@NREC)           ;How many sectors
       LD     C,A
       CALL   S100OUT
       CALL   WAIT_FOR_ACK        ;Wait for NO_ERRORS_FLAG to come back
       JP     NZ,ERR_NR           ;If error, just abort

       LD     HL,(@TADDR)         ;Set DMA address

MULTI_RD_SEC:
       LD     DE,(@SEC_SIZE)             ;For CPM this will be 128 Byte sector(s)
RD_SEC:CALL   S100IN              ;Note potential to lockup here & below (but unlightly)
       LD     (HL),A
       INC    HL
       DEC    DE
       LD     A,E
       OR     A,D
       JR     NZ,RD_SEC

       LD     A,(@NREC)           ;How many sectors of data worth
       DEC    A
       LD     (@NREC),A
       JR     NZ,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     A,S100_STATUS_B            ;Send data to ZFDC output (arrive with character to be sent in C)
       BIT    DIRECTION_BIT,A            ;Is ZFDC in output mode, if not wait
       JR     NZ,S100OUT
       BIT    DATA_OUT_RDY,A             ;Has previous (if any) character been read.
       JR     Z,S100OUT           ;Z if not yet ready
       LD     A,C
       OUT    S100_DATA_B,A
       RET

S100STAT:
       IN     A,S100_STATUS_B            ;Check if ZFDC has any data for S-100 system
       BIT    DATA_IN_RDY,A       ;Anything there ?
       RET    Z                   ;Return 0 if nothing
       XOR    A,A
       DEC    A                   ;Return NZ, & 0FFH in A if something there
       RET

S100IN:
       IN     A,S100_STATUS_B            ;Check if ZFDC has any data for S-100 system
```

```
       BIT    DIRECTION_BIT,A           ;Is ZFDC in input mode, if not wait
       JR     Z,S100IN            ;If low then ZFDC board is still in input mode, wait
       BIT    DATA_IN_RDY,A
       JR     Z,S100IN
       IN     A,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   BC                  ;This can be increased if you are displaying debugging info on the ZFDC
       PUSH   DE                  ;HEX LED display.
       LD     BC,0
       LD     E,STATUS_DELAY             ;Timeout, (about 2 seconds)
WAIT_1:       IN     A,S100_STATUS_B            ;Check if ZFDC has any data for S-100 system
       BIT    DIRECTION_BIT,A            ;Is ZFDC in input mode
       JR     Z,WAIT_2            ;if low then ZFDC is still in input mode
       CALL   S100STAT            ;Wait until ZFDC Board sends something
       JR     Z,WAIT_2
       CALL   S100IN              ;Get returned Error # (Note this releases the SEND_DATA routine on the ZFDC board)
       CP     A,NO_ERRORS_FLAG    ;Was SEND_OK/NO_ERRORS_FLAG sent back from ZFDC Board
       POP    DE                  ;Balance up stack
       POP    BC
       RET                        ;Return NZ if problem, Z if no problem
WAIT_2:       DEC    B
       JR     NZ,WAIT_1           ;Try for ~2 seconds
       DEC    B                   ;Reset B to 0FFH
       DEC    C
       JR     NZ,WAIT_1
       DEC    B                   ;Reset B to 0FFH
       DEC    C
       DEC    E
       JR     NZ,WAIT_1
       XOR    A,A
       DEC    A
       POP    DE                  ;Balance up stack
       POP    BC
       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


;-------------------------------------------------------------------------
;      Module to boot MSDOS from 5" DDDS disk  (Note this module has not been updated yet)
;
DOS:   LD     HL,SPEAKDOS_MSG            ;Announce on speaker
       CALL   SPEAK$
```

```
       CALL   CRLF

       JP     ERR_NR              ;Not done Yet

;      LD     A,4                 ;MSDOS.SYS STARTS AT SECTOR 4 SIDE B
;      LD     (@SCTR),A
;      LD     A,0
;      LD     (@TRK),A                  ;START ON TRACK 0
;      LD     A,1
;      LD     (@SIDE),A           ;START ON SIDE B
;      LD     A,FFILE_SIZE        ;SIZE OF DOS IN 512 BYTE SECTORS
;      LD     (@NREC),A
;
;      LD     A,01110100B         ;0,DD,5",SIDE 1, 0100=D:
;      CALL   MDSEL
;      JP     NZ,ERR_NR           ;ROUTINE TO SAY DRIVE NOT READY
;      LD     A,RSVCMD            ;SEND RESTORE COMMAND
;      CALL   DCMDI
;      JR     Z,DGETID
;DOS1: LD     HL,RESTORE_ERR            ;RESTORE FAILED
;      JP     ABORT_ERR_MSG
;
;DGETID:      CALL   DIDRD
;      JR     NZ,DOS1
;
;GETSEC: LD   HL,STARTDOS
;DGET1:       LD     C,'.'              ;to indicate on CRT sectors read
;      CALL   CO
;      LD     A,(@SCTR)
;      OUT    (SECTOR),A
;      LD     B,0                 ;256 BYTES
;      LD     C,DATA              ;DATA PORT
;      DI                         ;just in case
;      CALL   SWEB                ;SET WAIT ENABLE BIT
;      LD     A,RDCMD
;      OUT    (CMD),A
;      INIR
;      INIR                       ;512 BYTES TOTAL
;      LD     B,0
;DWAITF:      IN     A,(STATUS)
;      AND    1
;      DJNZ   DWAITF
;
;      CALL   DDWAIT
;
;      IN     A,(STATUS)          ;CHECK STATUS
;      AND    0FEH
;      JP     NZ,ERR_LD           ;ROUTINE TO SAY SECTOR READ ERROR
;
;      LD     A,(@NREC)
```

```
;       DEC     A
;       LD      (@NREC),A
;       JP      Z,STARTDOS
;
;       LD      A,(@SCTR)
;       INC     A
;       LD      (@SCTR),A
;       CP      0AH                 ;end of track yet?
;       JR      NZ,DGET1
;
;       LD      A,(@SIDE)
;       CP      1                   ;if on track 1 go to side 1 else side 0
;       JR      Z,TRK1A
;       LD      A,1                 ;FLAG CURRENT SIDE IS NOW B
;       LD      (@SIDE),A
;       LD      A,01110100B         ;SWITCH TO SIDE B
;       JR      TRK1B
;TRK1A:  LD   A,(@TRK)
;       INC     A
;       LD      (@TRK),A
;       LD      A,0
;       LD      (@SIDE),A           ;FLAG CURRENT SIDE IS NOW A
;       LD      A,01100100B         ;SWITCH TO SIDE A
;TRK1B:  CALL MDSEL
;       JP      NZ,ERR_NR           ;ROUTINE TO SAY DRIVE NOT READY
;
;DSEC: LD      A,1
;       LD      (@SCTR),A
;
;       LD      A,(@TRK)
;       OUT     (DATA),A
;       LD      A,MSKCMD            ;SEEK TO TRACK WITH VERIFY
;       CALL    DCMDI
;       JP      Z,DDRS3
;DSEC1:       LD      HL,MSGH4            ;SEEK ERROR MESSAGE
;       JP      ABORT_ERR_MSG
;
;xxxz: HALT

;DDRS3:  PUSH HL
;       CALL    DIDRD
;       POP     HL
;       JR      NZ,DSEC1
;       JP      DGET1
;
;DIDRD:       LD      HL,@IDSV
;       LD      BC,600H+DATA
;       CALL    SWEB
;       LD      A,RDACMD            ;SEND READ ID COMMAND
;       OUT     (CMD),A
```

```
;       INIR
;DWAITS:      IN     A,(STATUS)
;       AND    1
;       JR     NZ,DWAITS
;       CALL   DDWAIT              ;DISABEL WAIT STATE GENERATOR
;       LD     A,(@IDSV)           ;+++++++++++++
;       LD     B,A
;       LD     A,(@TRK)
;       CP     B                   ;RETURN WITH Z IF AT RIGHT TRACK
;       RET

;MDSEL:       CPL
;       OUT    (SELECT),A
;DRDYCK:      IN     A,(STATUS)
;       AND    80H
;       JP     NZ,DRDYCK
;       RET
;
;SEND  TYPE 1 COMMANDS (RESTORE,SEEK,STEP)
;
;DCMDI:       LD     (@CMDSV),A          ;TEMPORLY STORE COMMAND
;       LD     A,80H
;       LD     (@ERMASK),A
;DCMDI1:      IN     A,(STATUS)          ;IS 1793 READY
;       AND    01H
;       JP     NZ,DCMDI1
;       LD     A,(@CMDSV)
;       OUT    (CMD),A
;       CALL   DELAY_15            ;DELAY REQUIRED FOR A VALID STATUS
;DEEND:       IN     A,(STATUS)          ;END OF DISK COMMANDS ROUTINE
;       AND    01H
;       JP     NZ,DEEND            ;IS 1793 STILL BUSY
;       IN     A,(STATUS)
;       LD     D,A
;       LD     A,(@ERMASK)
;       AND    D                   ;CHECK FOR ERRORS
;       RET

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

SHOW_TIME:
       LD     HL,TIME_MSG
       CALL   PRINT_STRING        ;Print message up to '$'
       CALL   PRINT_TIME
       RET

SHOW_DATE:
       LD     HL,DATE_MSG
       CALL   PRINT_STRING        ;Print message up to '$'
       CALL   PRINT_DATE
```

```
       RET


PRINT_TIME:
       CALL   UPD_IN_PR           ;CHECK FOR UPDATE IN PROCESS
       JP     NC,RTC_2A           ;GO AROUND IF OK
       JP     RTC_ERROR           ;IF ERROR

RTC_2A:       LD     E,-2               ;-2 goes to 0 for PORT_INC_2
       CALL   PORT_INC_2          ;SET ADDRESS OF SECONDS
       IN     A,(CMOS_PORT+1)            ;Get BCD value returned
       LD     D,A                 ;SAVE IN D
       CALL   PORT_INC_2          ;SET ADDRESS OF MINUTES
       IN     A,(CMOS_PORT+1)            ;Get BCD value returned
       LD     C,A                 ;SAVE IN C
       CALL   PORT_INC_2          ;SET ADDRESS OF HOURS
       IN     A,(CMOS_PORT+1)            ;Get BCD value returned
       LD     B,A                 ;SAVE
       LD     E,0                 ;SET E TO ZERO
       CALL   DisplayTime
       XOR    A,A                 ;Clear Carry
       RET                        ;BACK TO MONITOR

RTC_ERROR:                        ;Indicate RTC Board is not present or Error
       SCF                        ;SET CARRY FOR ERROR
       RET

;Display time
;      Arrive with B = HOURS IN BCD
;                  C = Minutes in BCD
;                  D = Seconds in BCD
DisplayTime:
       PUSH   DE
       PUSH   BC
       LD     A,B
       CALL   PRINT_BCD           ;Hours.  Convert BCD to ASCII
       LD     C,':'
       CALL   ZCO
       POP    BC
       LD     A,C
       CALL   PRINT_BCD           ;Minutes.  Convert BCD to ASCII
       LD     C,':'
       CALL   ZCO
       POP    DE
       LD     A,D
       CALL   PRINT_BCD           ;Seconds.  Convert BCD to ASCII
       RET

PRINT_DATE:
       CALL   UPD_IN_PR
```

```
       JP     NC,RTC_4A
       JP     RTC_ERROR           ;IF ERROR

RTC_4A:       LD     E,6
       CALL   PORT_INC            ;POINT TO DAY
       IN     A,(CMOS_PORT+1)
       LD     B,A                 ;SAVE IN A
       CALL   PORT_INC            ;POINT TO MONTH
       IN     A,(CMOS_PORT+1)
       LD     D,A                 ;SAVE IN D
       CALL   PORT_INC            ;POINT TO YEAR
       IN     A,(CMOS_PORT+1)
       LD     C,A                 ;SAVE IN C
       LD     E,31H               ;POINT TO CENTURY BYTE SAVE AREA
       CALL   PORT_INC            ;
       IN     A,(CMOS_PORT+1)            ;GET VALUE
       LD     E,B                 ;GET DAY BACK
       LD     B,A
       CALL   DisplayDate
       XOR    A,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   DE
       PUSH   DE
       PUSH   BC
       PUSH   BC

       POP    BC
       LD     A,B
       CALL   PRINT_BCD           ;Century (19/20).  Convert BCD to ASCII
       POP    BC
       LD     A,C
       CALL   PRINT_BCD           ;Year.  Convert BCD to ASCII
       LD     C,'/'
       CALL   ZCO
       POP    DE
       LD     A,D
       CALL   PRINT_BCD           ;Month.  Convert BCD to ASCII
       LD     C,'/'
       CALL   ZCO
       POP    DE
       LD     A,E
       CALL   PRINT_BCD           ;Day.  Convert BCD to ASCII
```

```
       RET


UPD_IN_PR:                      ;Check we are ready to read clock
       PUSH   BC
       LD     BC,600           ;SET LOOP COUNT
UPDATE:       LD     A,0AH            ;ADDRESS OF [A] REGISTER
       OUT    (CMOS_PORT),A
       NOP
       NOP
       NOP
       IN     A,(CMOS_PORT+1)          ;READ  IN REGISTER [A]
       AND    A,80H            ;IF 8XH--> UIP BIT  IS ON (CANNOT READ TIME)
       JP     Z,UPD_IN_PREND           ;Are we ready/done
       DEC    BC
       LD     A,C
       OR     A,B
       JP     NZ,UPDATE        ;Try again
       XOR    A,A              ;
       SCF                     ;SET CARRY FOR ERROR
       POP    BC
       RET
UPD_IN_PREND:
       XOR    A,A              ;Clear Carry
       POP    BC
       RET                     ;RETURN

PORT_INC:
       LD     A,E
       INC    A                ;INCREMENT ADDRESS
       LD     E,A
       OUT    (CMOS_PORT),A
       RET


PORT_INC_2:
       LD     A,E
       ADD    2                ;INCREMENT ADDRESS
       LD     E,A
       OUT    (CMOS_PORT),A
       RET


PRINT_BCD:                      ;Print BCD in [A]
       PUSH   AF
       PUSH   AF
       RRA
       RRA
       RRA
       RRA
```

```
       AND    A,0FH
       ADD    A,30H
       LD     C,A                 ;Write high byte mins to CRT
       CALL   ZCO
       POP    AF
       AND    A,0FH
       ADD    A,30H
       LD     C,A
       CALL   ZCO
       POP    AF
       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     A,(BCTL)            ;A0H
       BIT    5,A
       LD     A,0FFH
       RET    NZ                  ;Ret NZ if CTS is High
       XOR    A
       RET                        ;Ret Z if CTS is Low

SPEAKOUT:
       XOR    A,A                 ;Will try 256 times, then timeout
SPXXX: PUSH   AF
       IN     A,(BCTL)            ;(A0), Is SCC TX Buffer empty
       AND    04H
       JR     NZ,SENDS            ;NZ if ready to recieve character
       POP    AF
       DEC    A
       JR     NZ,SPXXX
       RET
SENDS: POP    AF
       LD     A,C
       OUT    (BDTA),A            ;(A2), Send it
       RET
;
;SPEAKTOMM THIS IS A ROUTINE TO SEND A STRING TO TALKER [HL] AT STRING
SPEAK$:       LD     A,(HL)
       CP     '$'
       JR     Z,SPEAK1
       LD     C,A
       CALL   SPEAKOUT
       INC    HL
       JR     SPEAK$
```

```
SPEAK1:       LD     C,0DH               ;MUST END WITH A CR
       JP     SPEAKOUT


;>>>>>>>>>>>>>>>>>> MODEM/SERIAL I/O <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; Note the S100Computers I/O board will have the SSC set initially to 19,200 Baud
;
SERIAL_OUT:   XOR    A,A           ;Will try 256 times, then timeout
MODXXX:       PUSH   AF
       IN     A,(ACTL)            ;MODEM/SERIAL OUT
       AND    04H                 ;Are we ready for a character
       JR     NZ,SENDM            ;NZ if ready to recieve character
       POP    AF
       DEC    A
       JR     NZ,MODXXX
       RET
SENDM: POP    AF
       LD     A,C
       OUT    (ADTA),A            ;Send it
       RET

SERIAL_IN:    XOR    A,A           ;Will try 256 times, then timeout
SERIAL_INX:   PUSH   AF
       CALL   SERIAL_STAT         ;MODEN/SERIAL IN
       JR     NZ,GETMOD
       POP    AF
       DEC    A
       JR     NZ,SERIAL_INX
       RET
GETMOD:       POP    AF
       IN     A,(ADTA)
       RET

SERIAL_STAT:IN       A,(ACTL)
       AND    01H
       RET    Z                   ;Ret Z if nothing
       LD     A,0FFH
       XOR    A
       RET                        ;Ret FF/NZ if something



;PRINT MAIN MONITOR MENU ON CRT

KCMD:  LD     HL,MSG0                    ;Signon Msg again (K Command)
       CALL   PRINT_STRING
       LD     HL,SMSG                    ;SPEECH MESSAGE
       CALL   SPEAK$
       LD     HL,MENUMSG          ;Then Menu Message
       JP     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
;Z80 WILL BE IN A PERMANANT HOLD STATE UNTIL PIN #55 IS AGAIN LOWERED.

SWITCH_8086:
       LD     HL,MSG14
       CALL   PRINT_STRING
       IN     A,(SW86)            ;THIS SWITCHES CPU'S with no block Move
       NOP                        ;Z80 WILL BE HELD HERE
       NOP
       NOP
       NOP
       JP     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
;Z80 WILL BE IN A PERMANANT HOLD STATE UNTIL PIN #56 IS AGAIN LOWERED.
SWITCH_68K:
       LD     HL,MSG68K
       CALL   PRINT_STRING
       IN     A,(SW68K)           ;THIS SWITCHES CPU'S
       NOP                        ;Z80 WILL BE HELD HERE
       NOP
       NOP
       NOP
       JP     BEGIN               ;WILL DROP BACK TO REBOOT MONITOR

;
;
;
;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
TRAP:  HALT
;
;
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_LD_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=CP/M(Z) D=Disp  E=Echo   F=Fill G=Goto'
              DB     CR,LF
              DB     'H=Date   I=Time     J=Test    K=Menu  L=CPM(V) M=Move N=SeqMap'
              DB     CR,LF
              DB     'O=8086   P=CPM(IDE) Q=Port    R=Ports S=Subs   T=Type'
              DB     CR,LF
              DB     'V=Verify W=Port EDH 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 $'
CR_SMSG:      DB     CR,CR,CR,CR,'$'
;
              NOP
              HALT

;END
```