

```

; MASTER.Z80 This is the main monitor program for my system.
; Assemble and SLR's Z80ASM Assembler (Can also use the Cromemco Assembler)
; Use:- Z80ASM MASTER FH
;
; Since V4.9, this monitor is now duplicated into two similar but not identical
; copies. IF you have the V2 Z80 CPU S100 Board, the 4K monitor can reside in duplicate in an
; 8K 28C64 EEPROM (or 27C64 UV-ROM). There is the "normal" lower 4K monitor where the A12
; address line is low. This is the "normal" state upon reset (pin 16 of U17 is low which
; via P39, 7-8 will pull A12 of the EEPROM low, thereby selecting the lower 4K half
; of the 8K ROM. If you are using the original S100Computers Z80 CPU board this
; this is the setup.
;
; If you have the newer V2 board, outputting to port D3H bit 1 high will raise the A12 line
; thereby selecting the top 4K of the EEPROM. Outputting D3H bit 1 low will bring back
; the lower 4K section again.
;
; There will be two "versions" of the monitor; LOW & HIGH. The LOW version will reside
; from 0-FFFH in a 28C64 EEPROM. The HIGH version will reside from 1000H-1FFFFH in
; the same 28C64 EEPROM. The EQUATE ROM_HIGH_PAGE = FALSE sets things for the LOW version.
; If ROM_HIGH_PAGE = TRUE then the HIGH version will be assembled. Clearly there must be code
; common to both sections. The page switching code is near the start of the monitor. It is
; at the location "PAGE_CHANGE:" and uses the "XH" o "XL" menu commands.
;
; Both versions of the monitor have most of the same menu options. The main difference is
; the "HIGH" image has the XModem routine (Menu "C") to download files directly into RAM
; from another compute over a serial port and the "O" command to switch in/test the 8086.
;
; The "LOW" image has (Menu "C") code to boot CPM (V3.0) from an 8" floppy using either the
; ZFDC or Versafloppy_II boards.
;
; BTW, within the code (which normally starts at F000H), there is a small jump table fixed at
; F800H for old CPM V1.4 floppy disk BIOS boot routines. My old CPM V1.4 software counts on
; these locations and they should not be changed. You can ignore them if you like.
;
; Also remember things are very tight right now. The code below has only a few spare bytes
; left to fit everything in ROMs. Always check for code overflow.
;
; To assemble under windows...
; Load Altair.EXE in Windows CMD box
; do cpm3
; I:
; I:>Submit master
;
; Master.HEX is written back to the same windows folder that the PC file "altair.exe" is in.
;
; Programming an EEPROM for the V2-Z80 Board with a PROM burner is a little tricky.
; To get started just set ROM_HIGH_PAGE = FALSE and do the following
; Using a 28C64 EEPROM and a Wellon VP-280 or VP290 Programmer
; For a monitor at F000H-FFFFH:-
;
; Make sure the ORG is F000H.

```

```

; Load the master.HEX file
; Clear Buffer Options:- 00
; Load Buffer Address :- 0000
; From File address use:- F000H
; File Size use:- 1000H

; This will put the code (4K) in the lower "half" of the 8K EEPROM. It can be seen/edited
; at 0000H in the ROM. On the S100Computers/N8VEM Z80 board make sure P39 is jumpered
; 3-4 (PROM address line A12 always low). You can use all your Z80 monitor commands
; except those that utilize page switching.
;
; To burn an EEPROM that adds code to the upper half of the PROM. Repeat the above. Then set
; the equate ROM_HIGH_PAGE = TRUE and reassemble the code. This time use the following
; VP-20 Programmer settings:-
;
; Clear Buffer Options:- Disable <----- Note
; Load Buffer Address :- 1000
; From File address use:- F000H
; File Size use:- 1000H
;
; Examine the code to be burnt to the EEPROM, It should start at 0H and Run to 1FFFH with the
; early code at 0h the same as at 1000H.
; Burn the EEPROM, insert in the CPU board and use the "XH" and "XL" commands to switch pages
;
;
; 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    Initalization 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 Incorporated 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, I/O to serial port.
; V4.8 10/27/12 Corrected SERIAL_STAT: error. Further cleaned up info on IOBYTE
; V4.9 11/3/12 Added ability (XH, XL) Command to switch to low/high pages in 28C64 EEPROMS (for V2 Z80 Board)
; V5.0 12/18/13 Add "C" menu option to download a ZModem file from a PC/Serial/USB port directly to RAM
; V5.01 12/30/13 Add PC-Printer routines for new parallel Port Board with ST8C34 chip (ST8C4 = TRUE)
; V5.02 1/31/14 Allow output to port EEH 01h activate TMA0* (8086,80286,80386 Boards) and 02h TMA1* (68000 Board)
; V5.03 7/25/2014 Removed error in Z memu command
; V5.04 7/29/2014 Patched date to show correct century
; V5.1 3/18/2015 Added to test module to output "3" on console with 8986 reset (UP8086:) in High RAM section
; V5.2 12/17/2016 Change master/slave signon message ("O" commamd) for more general use(multiple slave CPU boards)

FALSE EQU 0
TRUE EQU NOT FALSE

BASE EQU 0F000H ;Start or EPROM Location (Assume a 2732 or half of a 28C64)
;Note can test running in CPM (set ORG to 100H), BUT disable setting the stack
;by commenting out the 4 lines below SETUP_STACK:

VERSA EQU BASE+800H ;<-----<<<<<< LOCATION OF FLOPPY BIOS JMP's (For old CPM V1.2 Software)
; BE SURE THERE IS NO OVERFLOW OF THE FIRST
; PART OR THIS MONITOR INTO THIS BIOS AREA
; CHECK THE MASTER.LST FILE

ORG BASE
;<-----<<<<<< LOCATION OF START OF MONITOR (First part)
;Normally this monitor will reside at F000H
;The CPM (V1.2) jump table will be at F800H forn some old software I have.
;Note. This is not a requirement for all my recent software/CPM3.

ROM_HIGH_PAGE EQU TRUE ;If false, code is for lower 4K of 8K EEPROM. If TRUE, then upper 4K section
ST8C4 EQU TRUE ;TRUE if S100_Parallel_IO Board. False if IMSAI PIO Board

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)
NN      EQU      0H          ;[I] INITIAL VALUE
SOH     EQU      1           ; For XModem etc.
EOT     EQU      4
ACK     EQU      6
NAK     EQU      15H

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 MSDOS Consol output to CGA/VGA Board instead of Propeller board
;          | | | | | | | | |.....For 8086 Monitor, 0=Do not initilize extra ROMS
;          | | | | | | | | |.....Unused
;          | | | | | | | | |.....Unused
;          | | | | | | | | |.....For Z80 Monitor,  0=ALL Consol I/O via ACIA Serial port on S100Computers Serial-IO Board
;          | | | | | | | | |.....Remember the serial port is currently set for 38.4K, 1 Stop,NP
;          | | | | | | | | |.....For CPM3,         0=Force reformat of Memory disk upon any CPM3 cold 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 hardware diagnostic test. (See code at FFFF0H in the 8086 monitor)
;
;
;----- PROPELLER CONSOLE_IO (OR 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)
IF ST8C4          ;If S100_Parallel_IO Board for Printer output
PRINTER_CTRL EQU 0C2H ;ST8C4 Control Port
PRINTER_STATUS EQU 0C1H ;ST8C4 Status port
PRINTER_OUT EQU 0C0H ;ST8C4 Data port
PRINTER_ST_LOW EQU 0DH ;OUT STROBE LOW
PRINTER_ST_HIGH EQU 0CH ;OUT STROBE HIGH

```

```

ELSE
PRINTER_STATUS EQU 5 ;IN, IMSAI PIO Board PARRELL PORT
PRINTER_OUT EQU 5 ;OUT
PRINTER_STROBE EQU 4 ;OUT
ENDIF

DIAG_LEDS EQU 5 ;OUT, Will use this port (either board) 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,
BDTA EQU 0A2H ;CHANNEL B DATA
ACTL EQU 0A1H ;CHANNEL A CONTROL (MODEM_CTL_PORT)
ADTA EQU 0A3H ;CHANNEL A DATA (MODEM_DATA_PORT)

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

BASE_PORT EQU 0A1H ;For XModem communication routines
MODEM_CTL_PORT EQU BASE_PORT ;A1H (Note modem I/O will be on CHANNEL A. Speaker on CHANNEL B
MODEM_DATA_PORT EQU BASE_PORT+2 ;A3H

MODEM_SEND_MASK EQU 4
SEND_READY EQU 4 ;VALUE WHEN READY
MODEM_RECV_MASK EQU 1
RECV_READY EQU 1 ;BIT ON WHEN READY
MODEM_ERR_LIMIT EQU 8 ;Max number of Modem serial port re-reads aborting
MODEM_RTS_DELAY EQU 1 ;Time to check Modem RTS line (See XMODEM_LOAD & P_XMODEM_LOAD). Not critical.

RECVD_SECT_NO EQU 0H ;IX Offset for XModem Recieved Sector Number
SECTNO EQU 1H ;IX Offset for XModem CURRENT SECTOR NUMBER
ERRCT EQU 2H ;IX Offset for XModem ERROR COUNT

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

CMOS_PORT EQU 70H ;Base Port for CMOS Clock/Date 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_BITEQU 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 initialization/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)

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)

;----- 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)
@CMD SV EQU 4EH ;COMMAND SAVE
@SP SV 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 RESEKES
@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
IDEalline 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 + IDEalline
REGsector EQU 0BH ;IDECs0line + IDEalline + IDEA0line
REGcyLSB EQU 0CH ;IDECs0line + IDEa2line
REGcyMSB EQU 0DH ;IDECs0line + IDEa2line + IDEA0line
REGshd EQU 0EH ;IDECs0line + IDEa2line + IDEalline ; (0EH)
REGCMD EQU 0FH ;IDECs0line + IDEa2line + IDEalline + IDEA0line ; (0FH)
REGstatus EQU 0FH ;IDECs0line + IDEa2line + IDEalline + IDEA0line
REGcontrol EQU 16H ;IDECs1line + IDEa2line + IDEalline
REGastatus EQU 17H ;IDECs1line + IDEa2line + IDEalline + 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

```



```

DW BEGIN      ; "Y" SPARE
DW SIZE       ; "Z" FIND HIGHEST R/W RAM

;
-----
;
BEGIN:
    LD      A,'#'          ;For quick hardware diagnostic test
    OUT    (CONSOL_OUT),A ;Must see a "#" on the CRT in ROM access is active

    DI                      ;No interrupts
    XOR    A                ;SET INTERRUPT TO PAGE 0H
    LD     I,A              ;Z80 Interrupt page 0
    OUT    (Z80PORT+1),A   ;KILL THE INTERSYSTEMS Z80 CPU BOARD INT CONTROLLER (If present)
    OUT    (SW_TMAX),A     ;Make sure TMA0*,TMA1*,TMA2* & TMA3* S100 lines are high

SETUP_STACK:
    LD     SP,AHEAD-4      ;SETUP A FAKE STACK
    JP     MEMSZ1         ;RETURNS WITH TOP OF RAM IN [HL]
    DW     AHEAD          ;A Return opcode 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]

IF      ST8C4             ;If S100_Parallel_IO Board for Printer output
    LD     A,08H          ;Initilize the ST8C4 PC-Printer Port IO
    OUT    (PRINTER_CTRL),A

ELSE
    LD     A,0FFH         ;IMSAI PIO Board. Clear Printer strobe, comes up 0 on a reset
    OUT    (PRINTER_STROBE),A

ENDIF

    LD     A,0000000B     ;Turn all LED's off as a diagnostic on parallel port board
    OUT    (DIAG_LEDS),A ;FLAG PROGRESS VISUALLY FOR DIAGNOSTIC (ALL LED' ON)
                                ;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,1000000B     ;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)

                                ;We need to clear the 8259A otherwise the 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 the 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 ON)
OUT     (DIAG_LEDS),A

LD      HL,SIGNON_MSG    ;Have a Stack, so now we can use CALL
CALL    PRINT_STRING

CALL    INIT_SCC_A       ;Initilize the S100Computers I/O Board
CALL    INIT_SCC_B       ;Initilize the Zilog 8530-A (XModem I/O)
                          ;Initilize the Zilog 8530-B (Speech synthizer)
                          ;Next initilize the 8255 on this same board

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
                          ;Note 8255 port C used for DLP-USP Port Chip Status

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

LD      HL,IOBYTE_MSG   ;Print Current IOBYTE value
CALL    PRINT_STRING

```

```

IN      A,(IOBYTE)          ;Show IOBYTE. If bit 0=0 (force printer output), CMP/3 boot will hang
CALL    ZBITS

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 ON)
OUT     (DIAG_LEDS),A

LD      HL,CR_SMSG_SP       ;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 ON)
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

```

```

;

```

```

;

```

```

;-----

```

```
; Module to SET 28C64 EEPROM Pages. << Make sure this location is the same for both pages >>
; COMMON PAGE CODE:-
```

```
PAGE_CHANGE:
```

```
CALL ZPCHK ;Get another character from console
CP 'H' ;If we entered "XH" then OUTPUT TO PORT
JR Z,SET_HIGH_PAGE
CP 'L' ;If we entered "XL" then OUTPUT TO PORT
JP Z,SET_LOW_PAGE
LD C,'*'
JP ZCO ;WILL ABORT IF NOT 'H' OR 'L'
```

```
SET_HIGH_PAGE:
```

```
LD HL,H_ROM_MSG
CALL PRINT_STRING
LD A,06H ;04H for MEMORY MANAGEMENT TO OVERLAP + Bit 1 = HIGH
```

```
SET_PAGE:
```

```
OUT (Z80PORT+3),A
NOP
NOP
JP START ;Back to monitors next command
```

```
SET_LOW_PAGE:
```

```
LD HL,L_ROM_MSG
CALL PRINT_STRING
LD A,04H ;04H for MEMORY MANAGEMENT TO OVERLAP + Bit 1 = LOW
JR SET_PAGE
```

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

```
TOM: LD C,(HL)
INC HL
CALL CO
DJNZ TOM
RET
```

```
PRINT_STRING:
```

```
;A ROUTINE TO PRINT OUT A STRING @ [HL]
;UP TO THE FIRST '$'.
LD A,(HL)
INC HL
CP '$'
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                ;Print [HL] with no space afterwards
        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 ROUTINE TO RESET THE STACK.
;Returns top of RAM in [HL]
```

```
MEMSZ:  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,ZBOOT
        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

```



```

LSTAT1:    XOR    A                ;PUT 0FFH IN [A] IF READY & NO ZERO FLAG
          DEC    A
          RET
ELSE
LSTAT: IN   A, (PRINTER_STATUS)    ;IMSAI PIO Board PARRELL PORT
          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
ENDIF

;----- BOOT UP CPM FROM HARD DISK ON S100COMPUTERS IDE 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_SP    ;Announce Booting CPM 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 initialization
          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 initialization problem
IDError:
          LD    HL,DRIVE_NR_ERR    ;Drive not ready

```

```

JP      ABORT_ERR_MSG

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

CALL    IDEwaitnotbusy   ;Make sure drive is ready
JR      C,IDEError      ;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 regisiter address
OUT     (IDECport),A

OR      IDErcline       ;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

```

```

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

```

```

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"
JP      ABORT_ERR_MSG

```

;----- IDE 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
                                ;with the pulse length. Symptoms are: incorrect data comming
LD      A,IDEreset          ;back from a sector read (often due to the wrong sector being read)
OUT     (IDECport),A        ;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

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

;Wait for the drive to be ready to transfer data.
;Returns the drive's status in Acc
IDEwdrq:
    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
    OUT   (IDECport),A                   ;Deselect all lines including WR line

    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

```

```

MAP1: LD B,1
      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 OF 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,OUT_PORT
CP      'I'                ;INPUT FROM PORT
JP      Z,IN_PORT
LD      C,'*'
JP      ZCO                 ;WILL ABORT IF NOT 'I' OR 'O'

```

```

IN_PORT:
LD      C,1                ;IN Port
CALL   ZHEXSP
POP     BC
IN      A,(C)
JP     ZBITS

```

```

;
OUT_PORT:
CALL   ZHEXSP             ;OUT Port
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 IMG 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 original 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
    RET                    ;Get back normal register set

```

```

;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,SW_TMA0     ;Inputting here will switch out the Z80 to 8086/80286
        JR    Z,SKIP       ;Skip because we don't want that right now
;
        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 Initalization
;Note Zilog SCC serial port A will be set to 19,200 Baud initially (for speech synthesizer).
;Note Zilog SCC serial port B will be set to 38,400 Baud initially (for XModem etc).

```

```

INIT_SCC_A:
        LD    A,ACTL       ;Program Channel A
        LD    C,A
        LD    B,0EH       ;Byte count for OTIR below
        LD    HL,SCCINIT_A
        OTIR
        RET

```

```

INIT_SCC_B:
        LD    A,BCTL       ;Program Channel B
        LD    C,A
        LD    B,0EH       ;Byte count for OTIR below

```

```

LD    HL,SCCINIT_B
OTIR
RET
;
;   BOTH SSC's are set for 19,200 BAUD
SCCINIT_A:
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
DB    0BH           ;Set RTS,DTR, Enable. 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 <<<<<<<<<< XModem I/O
;   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

SCCINIT_B:
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    05H           ;Point to WR5
DB    0EAH          ;Enable, Transmit 8 bits
DB    0BH           ;Set RTS,DTR, Enable. Point to WR11
DB    56H           ;Recieve/transmit clock = BRG
DB    0CH           ;Point to WR12
DB    06H           ;Low byte 19,200 Baud <<<<<<<<<< Note Speech synthizer defaults to this value
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
                ;Check the .LST file there is no code overwrite

```

```

;-----

```

```

;    VERSAFLOPPY II DOS SYSTEM LINKAGES      (USED BY SDOS & 2.2 CP/M)
;    These are residue JP's for my old CPM BIOS'es. Only LOADER is now functional.

```

```

FBOOT: JP      VBOOT                ;COLD START ENTRY FOR VERSAFLOPPY-II
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          ;SELDISK
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)

```

```

;----- END of FIXED LOCATION VERSA JMP TABLE-----

```

```

BIOS_JP_ERR:
    LD    HL,BIOS_ERR                ;"BIOS JMP no longer implemented in ROM @ F800H."
    JP    ABORT_ERR_MSG

```

```

;-----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                 ;I am having a problem obtaining teh century.
LD     A,20H              ;just force it to year 2000
CALL   PRINT_BCD         ;<--20
POP    BC                 ;Century (19/20). Convert BCD to ASCII
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:                  ;ADDRESS OF [A] REGISTER
    LD A,0AH
    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

```



```
JP PRINT_STRING
```

```
;
;THIS ROUTINE ACTIVATES THE S100 TMA0* LINE USING THE SMB (SWITCHES CONTROL OVER TO THE 8088, 8086 or 80286....)
;Inputting (any) data from Port SW_DMA0 lowers the S-100 TMA0* PIN #55 line.
;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 THE S100 PIN #55 IS AGAIN RAISED.

;NOTE FOR THE NEW V2 SMB RAISING BIT 0 OF PORT 0EEH (AND ALSO LOWERS TMA0* FOR THE SMB V1 Board).
;WE WILL INCLUDE BOTH OPTIONS
```

```
SWITCH_8086:
    LD    HL,MSG14
    CALL  PRINT_STRING
    IN    A,(SW_TMA0)      ;THIS SWITCHES CPU'S with no block Move
    NOP                                ;Z80 WILL BE HELD HERE
    NOP
    LD    A,01             ;Utilize the more specific circuit on the V2-SMB
    OUT   (SW_TMAX),A     ;Make sure its bit 0
    NOP
    NOP
    NOP
    JP    BEGIN           ;WILL DROP BACK TO REBOOT MONITOR
```

```
;THIS ROUTINE ACTIVATES THE S-100 TMA1* LINE USING THE V2-SMB (SWITCHES CONTROL OVER TO THE 68000 CPU Board)
;ON THE NEW V2 SMB, RAISING BIT 1 OF PORT ECH LOWERS TMA1*.
;THE Z80 WILL BE IN A PERMANANT HOLD STATE UNTIL THE S100 PIN #56 IS AGAIN RAISED.
```

```
SWITCH_68K:
    LD    HL,MSG68K
    CALL  PRINT_STRING
    LD    A,02
    OUT   (SW_TMAX),A     ;Make sure its bit 1
    NOP                                ;Z80 WILL BE HELD HERE
    NOP
    NOP
    NOP
    JP    BEGIN           ;WILL DROP BACK TO REBOOT MONITOR
```

```
;
;
;
;THESE ARE ROUTINES NOT YET IMPLEMENTED
```

```
;
OFLIST:    RET                ;OFF LIST
TRAP:    HALT
```

```
RI:                ;READER
POO:               ;PUNCH
PRDY:              ;PUNCH STATUS (Sent to Serial port right now)
```

```

RSTAT:                ;READER STATUS (Input from Serial port right now)
ONLIST:               ;SWITCH ON LIST
DOS:                  ;BOOT UP MSDOS LOADER FROM 5" DISK THEN SWITCH TO 8086
NOT_DONE:            LD    HL, BIOS_ERR
                     CALL  PRINT_STRING
                     JP    START

```

```

;----- MESSAGES COMMON TO HIGH & LOW PAGE SECTIONS -----

```

```

;
SIGNON_MSG:  DB  SCROLL, QUIT, NO_ENHANCEMENT, FAST, BELL, CR, LF, LF
             DB  'Z80 ROM MONITOR @ F000H - '

```

```

IF  ROM_HIGH_PAGE
    DB  'HIGH '
ELSE
    DB  'LOW '
ENDIF
    DB  'PAGE (J.Monahan,12/17/2016) $'

```

```

DRIVE_NR_ERR: DB    BELL, CR, LF
              DB    'Drive not Ready.', CR, LF, LF, '$'
BIOS_ERR:     DB    'Code not done yet', CR, LF, LF, '$'
BOOT_LD_ERR:  DB    BELL, CR, LF
              DB    'Read Error.', CR, LF, LF, '$'
BOOT_LD1_ERR: DB    BELL, CR, LF
              DB    'BOOT error.', CR, LF, LF, '$'

```

```

H_ROM_MSG:   DB    CR, LF, 'ROM to HIGH ROM Page', CR, LF, '$'
L_ROM_MSG:   DB    CR, LF, 'ROM to LOW ROM Page', CR, LF, '$'

```

```

MSG14:       DB    BELL, CR, LF
              DB    'TMA line 0 Activated'
              DB    CR, LF, LF, '$'

```

```

MSG68K:      DB    BELL, CR, LF
              DB    'TMA line 1 Activated'
              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, '$'
SP_MSG       DB    'SP=$'
CR_SMSG_SP:  DB    CR, CR, CR, CR, '$'

```


;This code will read a download file from another computer (usually a PC) sent via a serial port and place it in RAM at a specified location. The file must be sent via the XModem protocol. On PC's there are many programs that do this. The "Absolute Telnet" program (by Celestial Software) is what I use.

```

BOOT_XMODEMH:
    LD    HL,MODEM_SIGNON          ;Send Modem signon message
    CALL  PRINT_STRING

    LD    HL,0                    ;SP to HL
    ADD  HL,SP
    LD    DE,40H                  ;Drop 40H bytes below to be safe
    SBC  HL,DE
    PUSH HL
    POP  IX                      ;Will store data below in RAM 40H bytes below SP to be safe

    XOR  A,A                      ;Initilize flags
    LD   (IX+RECVD_SECT_NO),A
    LD   (IX+SECTNO),A
    LD   (IX+ERRCT),A

    LD   HL,MODEM_RAM_LOC        ;Get RAM location for where to place code
    CALL PRINT_STRING
    LD   C,1
    CALL ZHEXSP                  ;Get 16 bit value, put on stack
    POP  IY                      ;DMA Value now in IY
    CALL ZCRLF

    CALL INIT_SCC_A              ;INITIALIZE THE SCC SERIAL PORT A

    LD   B,1                      ;TIMEOUT DELAY
    CALL RECV                    ;GOBBLE UP GARBAGE CHARS FROM THE LINE

    LD   HL,DOWNLOAD_SP          ;Speak downloading file
    CALL SPEAK$

RECV_LOOP:
    XOR  A,A                      ; ---- MAIN RECIEVE LOOP ----
    LD   (IX+ERRCT),A            ;GET 0
    LD   (IX+ERRCT),A            ;INITIAL ERROR COUNT SET TO 0

RECV_HDR:
    LD   HL,RMSG
    CALL PRINT_STRING
    LD   A,(IX+SECTNO)           ;Get current sector number
    INC  A
    CALL LBYTE                   ;Show Sector Number on Console

    LD   HL,MODEM_RAM_MSG       ;"H. IF OK, will write to RAM location"
    CALL PRINT_STRING
    PUSH IY
    POP  HL                      ;IY to HL
    CALL LADR                    ;Show DMA Address
    CALL ZCRLF

```

```

LD      B,5                ;5 SEC TIMEOUT
CALL    RECV
JP      NC,RHNTO          ;IF ALL OK (NO TIMEOUT), THEN DROP DOWN TO RHNTO TO GET DATA

RECV_HDR_TIMEOUT:
CALL    TOUT              ;PRINT TIMEOUT

RECV_SECT_ERR:
                                ;PURGE THE LINE OF INPUT CHARS
LD      B,1                ;1 SEC W/NO CHARS
CALL    RECV
JP      NC,RECV_SECT_ERR  ;LOOP UNTIL SENDER DONE
LD      A,NAK
CALL    SEND              ;SEND NAK
LD      A,(IX+ERRCT)
INC     A
LD      (IX+ERRCT),A
CP      A,MODEM_ERR_LIMIT
JP      C,RECV_HDR
CALL    CHECK_FOR_QUIT
JP      Z,RECV_HDR
LD      HL,BAD_HEADER_MSG
CALL    PRINT_STRING
JP      EXIT

RHNTO:  CP      A,SOH        ;GOT CHAR - MUST BE SOH
JP      Z,GOT_SOH          ;Z IF OK
OR      A,A                ;00 FROM SPEED CHECK?
JP      Z,RECV_HDR
CP      A,EOT
JP      Z,GOT_EOT

                                ;DIDN'T GET SOH -
CALL    LBYTE
LD      HL,ERRSOH
CALL    PRINT_STRING
JP      RECV_SECT_ERR

GOT_SOH:
LD      B,1
CALL    RECV
JP      C,RECV_HDR_TIMEOUT
LD      D,A                ;D=BLK #
LD      B,1
CALL    RECV              ;GET CMA'D SECT #
JP      C,RECV_HDR_TIMEOUT
CPL
CP      A,D                ;GOOD SECTOR #?
JP      Z,RECV_SECTOR

LD      HL,ERR2            ;GOT BAD SECTOR #

```

```

CALL PRINT_STRING
JP RECV_SECT_ERR

RECV_SECTOR:                                ;Sector is OK, so read data and place in RAM
LD A,D                                     ;GET SECTOR #
LD (IX+RECV_SECT_NO),A
LD C,0                                     ;INIT CKSUM
LD E,80H                                   ;Sector Byte Count
PUSH IY
POP HL                                     ;DMA address (IY) to HL

RECV_CHAR:
LD B,1                                     ;1 SEC TIMEOUT
CALL RECV                                  ;GET CHAR
JP C,RECV_HDR_TIMEOUT
LD (HL),A                                  ;STORE CHAR
INC HL
DEC E                                       ;Next sector byte
JP NZ,RECV_CHAR

LD D,C                                     ;VERIFY CHECKSUM
LD B,1                                     ;SAVE CHECKSUM
CALL RECV                                  ;TIMEOUT
JP C,RECV_HDR_TIMEOUT                     ;GET CHECKSUM
CP A,D                                     ;CHECK
JP NZ,RECV_CKSUM_ERR

LD A,(IX+RECV_SECT_NO)                     ;GOT A SECTOR, WRITE IF = 1+PREV SECTOR
LD B,A                                     ;SAVE IT
LD A,(IX+SECTNO)                           ;GET PREV
INC A                                       ;CALC NEXT SECTOR #
CP B                                       ;MATCH?
JP NZ,DO_ACK
LD (IX+SECTNO),A                           ;UPDATE SECTOR #
DO_ACK: LD A,ACK
CALL SEND

PUSH HL                                     ;ALL OK SO SAVE DMA Address in IY
POP IY
JP RECV_LOOP                               ;Back to Top recieve loop

RECV_CKSUM_ERR:
LD HL,ERR3
CALL PRINT_STRING
JP RECV_SECT_ERR

GOT_EOT:
LD A,ACK                                    ;ACK THE EOT

```

```

CALL SEND
JP XFER_CPLT

;-----
; XMODEM SERIAL PORT GET CHARACTER ROUTINE
;-----

RECV: PUSH DE ;SAVE D,E
      LD A,5H ;Lower RTS line
      OUT MODEM_CTL_PORT,A ;Sel Reg 5
      LD A,11101010B ;EAH
      OUT MODEM_CTL_PORT,A
      NOP
      NOP
MSEC: LD DE,0BBBBH ;1 SEC DCR COUNT
MWTI: IN A,MODEM_CTL_PORT
      AND A,MODEM_RECV_MASK
      CP A,RECV_READY
      JP Z,MCHAR ;GOT CHAR
      DEC E ;COUNT DOWN
      JP NZ,MWTI ;FOR TIMEOUT
      DEC D
      JP NZ,MWTI
      DEC B ;DCR # OF SECONDS
      JP NZ,MSEC ;MODEM TIMED OUT RECEIVING
      POP DE ;RESTORE D,E
      SCF ;CARRY SHOWS TIMEOUT
      RET

;GOT MODEM CHAR
MCHAR: IN A,MODEM_DATA_PORT
      POP DE ;RESTORE DE
      PUSH AF ;CALC CHECKSUM
      ADD A,C
      LD C,A
      POP AF
      OR A,A ;TURN OFF CARRY TO SHOW NO TIMEOUT
      RET

;-----
; XMODEM SERIAL PORT SEND CHARACTER ROUTINE
;-----

SEND: PUSH AF ;CHECK IF MONITORING OUTPUT
      ADD A,C ;CALC CKSUM
      LD C,A
SENDW: IN A,MODEM_CTL_PORT ;Don't worry PC is always fast enough!
      AND A,MODEM_SEND_MASK
      CP A,SEND_READY
      JP NZ,SENDW
      POP AF ;GET CHAR
      OUT MODEM_DATA_PORT,A

```



```

                                ;Raise RTS line to prevent the next character arriving
LD      A,5H                    ;while the Z80 is busy processing info
OUT     MODEM_CTL_PORT,A        ;Sel Reg 5
LD      A,11101000B            ;E8H
OUT     MODEM_CTL_PORT,A
RET

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

TOUT:  LD      HL,TOUTM          ;PRINT TIMEOUT MESSAGE
       CALL    PRINT_STRING
       LD      A,(IX+ERRCT)
       CALL    LBYTE
       CALL    ZCRLF
       RET

CO_A:  PUSH    AF                ;PRINT VALUE of [A] on CRT
       PUSH    BC
       LD      C,A
       CALL    CO
       POP     BC
       POP     AF
       RET

CHECK_FOR_QUIT:                  ;MULTIPLE ERRORS, ASK IF TIME TO QUIT
XOR     A,A                      ;GET 0
LD      (IX+ERRCT),A            ;RESET ERROR COUNT
LD      HL,QUITM
CALL    PRINT_STRING

CI3:   IN      A,CONSOL_STATUS    ;NEED CONSTAT TO CLEAN UP SHIFT KEYS ETC
AND     A,02H
JP      Z,CI3
IN      A,CONSOL_IN
AND     A,7FH
PUSH    AF                        ;SAVE CHAR
CALL    ZCRLF
POP     AF
CP      A,'R'
RET     Z                          ;RETURN IF RETRY
CP      A,'r'
RET     Z
CP      A,'Q'                      ;QUIT?
JP      NZ,LCQ
OR      A,A                        ;TURN OFF ZERO FLAG
RET

LCQ:   CP      A,'q'
JP      NZ,CHECK_FOR_QUIT
OR      A,A                        ;TURN OFF ZERO FLAG
RET

```



```

UP8086: JP    UP8086L
BOOT_XMODEM:
        JP    NOT_DONE

```

;Basic test for 8086 on reset. The 8086 will jump to FFFF0H. If there is a ROM there nothing will change and the 8086 will jump from F000:FFF0 to the start of the ROM monitor. If however no ROM is present the 8086 will find the RAM code below and continuously put "3" on Consol via port 01H

```

UP8086L:  LD    A,0FCH                ;Point to 8086 Reset location
          OUT   (Z80PORT+2),A        ;Re-map to 0000H to FC000H
          LD    HL,3FF0H

          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            ;Console port (note no status check)
          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

```

;VBOOT or ZBOOT 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

```

```

VBOOTL:  XOR    A,A                ;0 = Flag as Boot from Versafloppy II FDC
          JR    BOOT_COLD
ZBOOTL:  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_SP      ;Announce on speaker
          CALL  SPEAK$
          XOR   A
          LD    (CDISK),A           ;MAKE CURRENT DISK A:
          LD    (CIOBYTE),A        ;CLEANUP IOBYTE

```



```

        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

```



```

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:
PUSH    BC                 ;Delay to wait for ZFDC to return data. There is a timeout of about 2 sec.
PUSH    DE                 ;This can be increased if you are displaying debugging info on the ZFDC
LD      BC, 0              ;HEX LED display.
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
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
        ;
        ;      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

MENUMSG:  DB     CR,LF
          DB     'A=Memmap B=68000      C=CP/M(F) 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) QI,O=Port R=Ports S=Subs  T=Type'
          DB     CR,LF
          DB     'V=Verify W=Port EDH XH=PAGE  Z=Top  @=Flush Printer'
          DB     CR,LF,LF,'$'

RESTORE_ERR: DB     BELL,CR,LF
             DB     'Restore Failed.',CR,LF,LF,'$'
SEEK_ERROR_MSG: DB     BELL,CR,LF
                DB     'Seek Error.',CR,LF,LF,'$'
VF_HUNG:      DB     'VF Controller Hung',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,'$'
             NOP

ENDIF
;----- End of LOW PAGE
;END

```