```
; THIS IS A PROGRAM TO PLACE A CPM3 CPMLDR ON A SD 8" IBM DISK
; THE PROGRAM UTILIZES THE S100Computers ZFDC Board TO WRITE THE SECTORS TO THE DISK.

; The CPM3 boot load process is in two steps. The ROM monitor reads in the very first sector
; (independent of the operating sysyem) from the disk. This 128 byte (at the most), program
; reads in the rest of the sectors required for that operating system -- in this case CPM3.
; The loaction, number of sectors etc. required is set by flags in RAM (40-53H). These locations
; are not used by CPM and are set to be consistant with my older operating systems such as
; SD Systems version of CPM2.2 (SDOS). The above ROM monitor program is counting on them being set.
; Note the 128 byte boot loader sets a flag in memory (46H) to Z if a NON-BANKED version of CPM3
; is to be loaded, NZ if a banked version is to be loaded. This is require because the CPMLDR
; for BANKED systems requires banked memory hardware to be activated during the loading process.
; (Actully you can load and run a NON-BANKED CPM3 system in "BANKED" RAM hardware mode it's just
; the system will not know and still have to work in 64K or total RAM.)


;              JOHN J MONAHAN         VERSION 2.0 (02/21/2011)

;       V3.0    2/22/2011             No longer requires Master EEPROM at 0F800H
;       V3.1    2/23/2011             Combined Banked & Non Banked versions.
;       V3.2    2/24/2011             Added Y or N to CRT for Banking/Non Banking option

BELL            EQU     07H
CR              EQU     0DH
LF              EQU     0AH
ESC             EQU     1BH
NBYTES          EQU     128
STDSDT          EQU     26              ;STANDARD 8" 26 SECTORS/TRACK

EEPROM          EQU     0F000H          ;Base location of Z80 Monitor
?LOADER         EQU     EEPROM+839H     ;LOAD N RECORDS (See below)

BOOT_UNIT       EQU     42H             ;RAM area used by EEPROM monitor to store the disk "Unit Byte"
                                        ;This contains bits to select density, size and side of disk.
                                        ;00H for a SDSS 8" in Versafloppy's drive 0.
                                        ;(Not used with the newer versions of the MASTER.Z80 Monitor V4.5+)
BOOT_COUNT      EQU     53H             ;RAM area used by EEPROM monitor to store SECTORS/TRACK
BOOT_TADDR      EQU     40H             ;RAM area used by EEPROM monitor to store DMA address
BOOT_SCTR       EQU     43H             ;RAM area used by EEPROM monitor to store starting SECTOR
BOOT_TRK        EQU     44H             ;RAM area used by EEPROM monitor to store starting TRACK
BOOT_NREC       EQU     45H             ;RAM area used by EEPROM monitor to store number of SECTORS to R/W
BOOT_BANK_FLAG  EQU     46H             ;Z if a non-banked CPMLDR file is to be run
                                        ;NZ for a banked CPMLDR (the CPMLDR does not use this byte)



;       PORTS  FOR FOR SD Systems Video Board
                                        ;With this (non-banked system) the EEPROM Monitor at F000H
                                        ;is always present. This is used for Console I/O only
CIN             EQU     1H              ;IN Data SD Systems Video Board
COUT            EQU     1H              ;OUT Data SD Systems Video Board
CSTAT           EQU     0H              ;Status Port SD Systems Video Board


CPM     EQU     0H                      ;A jump to here reboots CPM (if present in RAM).


;--------------- 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_WRITE_SECTOR  EQU 11H           ;Write data to the CURRENT sector (on current track,side,drive).
CMD_HANDSHAKE  EQU    21H           ;Handshake command only sent during board initilization/testing


; This program expects the CPM LOADER image to be in RAM at 900H.
; It will assume an 8" single density drive standard IBM format.
; Starting on track 0 sector 1 it will write 2 tracks (52 sectors) to the disk
;
        ORG     100H

        LD      HL,0
        ADD     HL,SP
        LD      (OLDSTACK),HL
        LD      SP,NEWSTACK
                                    ;SETUP DISK PARAMATERS
        LD      HL,MSG_SIGNON       ;Signon and check to continue
        CALL    PSTRING
        CALL    ZCI
        JP      NZ,EXIT1
        LD      C,'Y'
        CALL    ZCO

CHECK_BANKS:
        LD      HL,MSG_BANKED       ;Banked or non banked CPMLDR?
        CALL    PSTRING
        CALL    ZCI
        AND     5FH                 ;CHANGE TO UPPER CASE ONLY
        CP      ESC
        JP      Z,EXIT2
        CP      'N'
        JP      Z,NON_BANKED
        LD      A,1
        LD      (904H),A            ;This is a sneaky way of getting a flag on to the boot sector
        LD      C,'Y'               ;Already have a 0 at position 904H below
        CALL    ZCO
        JP      START               ;which resides here in this program at 900H (see below)
NON_BANKED:
        LD      C,'N'               ;Already have a 0 at position 904H below
        CALL    ZCO
                                    ;SETUP DISK PARAMATERS
START:  LD      HL,MSG_WRITING      ;"Writing sectors"
        CALl    PSTRING

        LD      A,STDSDT            ;26 SECTORS/TRACK
        LD      (@COUNT),A
        LD      HL,900H             ;DMA address to read from
        LD      (@TADDR),HL
        LD      A,1                 ;Starting Sector
        LD      (@SCTR),A
        LD      A,0                 ;Start on track 0
        LD      (@TRK),A
        LD      A,52                ;NUMBER OF SECTORS TO WRITE
        LD      (@NREC),A
        LD      HL,NBYTES           ;Will assume 128 byte sectors for now
        LD      (@SEC_SIZE),HL

        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 ZFDC board is there
        CP      A,CMD_HANDSHAKE             ;Make sure we get HANDSHAKE byte back
        JP      NZ,NO_ZFDC_ERROR    ;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,NO_ZFDC_ERROR    ;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,FORMAT_ERROR                 ;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,SET_DRIVE_ERROR      ;If error, just abort

        LD      HL,CRLF
        CALL    PSTRING

MULTI_WRITE:
        CALL    ZFDC_WRITE_SECTOR       ;<<< WRITE A SECTOR
        JP      NZ,SEC_WRITE_ERROR
        CALL    INCP                    ;Increment sector, track, adjust NREC
        JR      NZ,MULTI_WRITE

        LD      HL,MSG_DONE
        CALL    PSTRING


ZFDC_WRITE_SECTOR
        LD      C,'.'                   ;Show progress on screen
        CALL    ZCO

        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,SET_TRK_ERROR        ;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,SET_SEC_ERROR        ;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,SEEK_ERROR           ;If error, just abort

        LD      C,CMD_WRITE_SECTOR      ;Routine assumes required Drive Table,Drive,Side,Track, and sector are already
sent to board
        CALL    S100OUT                 ;(Note [HL]-> Sector DMA address)
        CALL    WAIT_FOR_ACK            ;Wait for NO_ERRORS_FLAG to come back
        JP      NZ,SEC_WRITE_ERROR      ;If error, just abort

        LD      HL,(@TADDR)             ;Set DMA address
        LD      DE,(@SEC_SIZE)          ;For CPM this will be 128 Byte sector(s)

WR_SEC: LD      C,(HL)
        CALL    S100OUT                 ;Note potential to lockup here & below (but unlightly)
        INC     HL
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,WR_SEC
        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]


;       INC SECTOR AND TRACK
INCP:   LD      HL,(@TADDR)
        LD      DE,(@SEC_SIZE)
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                      ;>>> NOTE, ASSUMES @TRK=SECTOR+1 IE 44H
        INC     (HL)                    ;Now points to TRACK#
        OR      A                       ;MAKE SURE TO RETURN NZ
        RET


ZCI:    IN      A,(CSTAT)               ;console input
        AND     A,02H
        JR      Z,ZCI
        IN      A,(CIN)                 ;return with character in A
        AND     A,7FH                   ;Strip parity in case ASCII keyboard sends one
        RET

ZCO:    IN      A,(CSTAT)               ;console output (arrive with character in C)
        AND     A,04H                   ;Note character is in C and A on return.
        JR      Z,ZCO
        LD      A,C
        OUT     (COUT),A
```

```
        RET


EXIT1:  LD      C,'N'
        CALL    ZCO
EXIT2:  LD      HL,MSG_ABORT            ;Did not write image msg
        JP      PRINT_ERROR

NO_ZFDC_ERROR:
        LD      HL,NO_ZFDC
        JP      PRINT_ERROR

FORMAT_ERROR:
        LD      HL,FORMAT_MSG
        JP      PRINT_ERROR

SET_DRIVE_ERROR:
        LD      HL,SET_DRIVE_MSG
        JP      PRINT_ERROR


SEC_WRITE_ERROR:
        LD      HL,WRITE_ERROR
        JP      PRINT_ERROR

WP_ERROR:
        LD      HL,WRITE_PROTECT
        JP      PRINT_ERROR

SEEK_ERROR:
        LD      HL,SEEK_ERROR_MSG
        JP      PRINT_ERROR

SET_TRK_ERROR:
        LD      HL,SET_TRK_MSG
        JP      PRINT_ERROR

SET_SEC_ERROR:
        LD      HL,SET_SEC_MSG
        JP      PRINT_ERROR

PRINT_ERROR:
        CALL    PSTRING                 ;Print error message in [HL]
        JP      EXIT

EXIT:   LD      HL,OLDSTACK
        LD      SP,HL                   ;Get back old stack before returning
        JP      CPM                     ;Assuming we have a valid CPM system running

PSTRING:PUSH    BC
PSTR2:  LD      A,(HL)
        CP      '$'
        JP      Z,PSTR1
        LD      C,A
        CALL    ZCO
        INC     HL
        JP      PSTR2
PSTR1:  POP     BC
        RET

                DS      20H             ;<- Space for new stack

NEWSTACK:       DW      0
OLDSTACK:       DW      0

@TADDR          DW      0               ;DMA address
@SCTR           DB      0               ;SECTOR
@TRK            DB      0               ;TRACK
@NREC           DB      0               ;SECTORS to Write
@COUNT          DB      0               ;SEC/Track
@SEC_SIZE       DW      0               ;Bytes per sector

MSG_SIGNON:     DB      LF,CR,LF,'This program, ZSYSGEN.COM V2.3, (02/23/2011) '
                DB      'will write a CPM3 CPMLDR '
                DB      CR,LF,'to the system tracks of an 8" SD Disk in Drive A:'
                DB      CR,LF,'It assumes a ZFDC FDC Board is present to do the'
                DB            ' multi-sector write.'
                DB      CR,LF,'Do you want to continue (Y/N)...$'
CRLF:           DB      CR,LF,'$'
MSG_BANKED:     DB      CR,LF,'Is this for a BANKED version of CPM3 (Y/N)...$'
MSG_WRITING     DB      CR,LF,'Writing 52 sectors$'
MSG_DONE:       DB      CR,LF,LF,'CPMLDR has been written to disk correctly. (52 X 128 byte sectors)'
```

```
               DB      CR,LF,'Returing to CPM.$'
MSG_ABORT:     DB      CR,LF,LF,BELL,'CPM Loader will NOT be written.'
               DB      CR,LF,'Returning to CPM.$'
WRITE_ERROR:   DB      CR,LF,BELL,'Error writing CPMLDR to disk',0DH,0AH,'$'
LOAD_ERROR:    DB      CR,LF,BELL,'Loader not written correctly to Tracks 0 & 1 of disk!$'
VF_HUNG:       DB      CR,LF,BELL,'ZFDC Board Controller Hung',CR,LF,LF,'$'
WRITE_PROTECT:  DB     CR,LF,BELL,'Disk is Write Protected.',CR,LF,LF,'$'
SEEK_ERROR_MSG:        DB      CR,LF,BELL,'Track Seek Error.',CR,LF,LF,'$'
NO_ZFDC:       DB      CR,LF,BELL,'The ZFDC Board is not responding.',CR,LF,LF,'$'
FORMAT_MSG:    DB      CR,LF,BELL,'Could not set disk Format on ZFDC board.',CR,LF,LF,'$'
SET_DRIVE_MSG: DB      CR,LF,BELL,'Could not set Select Drive on ZFDC board.',CR,LF,LF,'$'
SET_TRK_MSG:   DB      CR,LF,BELL,'Set Track error on ZFDC board.',CR,LF,LF,'$'
SET_SEC_MSG:   DB      CR,LF,BELL,'Set Sector error on ZFDC board.',CR,LF,LF,'$'


       ORG     900H                    ;This is code that will be in the disk first sector
                                       ;It will actually be run at 80H in RAM by the EEPROM loader
       LD      SP,80H                  ;Do not change the 900H location
       LD      A,0                     ;<-- this '0' can be changed to a '1' by CHECK_BANKS: above
       LD      (BOOT_BANK_FLAG),A      ;z if non banked, 1 if banked CPMLDR
       LD      A,02H
       LD      (BOOT_SCTR),A           ;The EEPROM bootstrap has already selected Drive A:, Track 0
       LD      A,51                    ;<---- We need only obtain the next 51 sectors for an 8", 128 byte sector
       LD      (BOOT_NREC),A
       LD      HL,100H                 ;Start loading them here
       LD      (BOOT_TADDR),HL
       CALL    ?LOADER                 ; TO EEPROM TO LOAD IN N CONTIGOUS SECTORS
       JP      100H                    ; Transfer control to here when done
;END
```