```
;       CP/M 3.0 LOADER BIOS FOR THE S100Computers (or ITHACA INTERSYSTEMS) Z80 BOARDS
;       AND THE ZFDC FDC BOARD
;
; The only relevence to the Z80 board has to do with the fact that
; this CPU board has two ports that allow a window in the 64K RAM space to be re-mapped
; to anywhere within a 24 bit address space. This allows convinient bank switching
; for CPM3 in a CPM3 Banked system. In a non-banked CPM3 system any Z80 CPU card can be used.
;
; Note this simple version will assume we are only booting from an 8" SD,SS IBM format disk.
;
;               WRITTEN BY     JOHN MONAHAN   (11/27/2009)
;

BELL    EQU     07H
CR      EQU     0DH
LF      EQU     0AH
NBYTES  EQU     128                     ;We will assume an 8" SSSD IBM Format type disk (128 byte sectors)
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)


MPURR0  EQU     0D2H                    ;CPU BOARD PORT TO SWITCH IN MEMORY BANKS (ALSO BIT 0 OF D3 FOR EPROM Removal)
MPURR1  EQU     0D3H

CRTSTAT EQU     0H                      ;SD Systems Video Board Consol I/O
CRTOUT  EQU     1H


;       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

;Commands used in this module for the ZFDC Board:-

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$SEEK$TRACK EQU      0EH             ;Seek to track to (IY+DRIVE$TRACK) with the track verify bit set on CURRENT
drive/format
CMD$READ$SECTOR           EQU   10H             ;Read data from the CURRENT sector (on current track,side,drive).
CMD$HANDSHAKE   EQU     21H             ;Handshake command only sent during board initilization/testing
CMD$SET$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)

NO$ERRORS$FLAG EQU      00H             ;No Errors flag for previous cmd, sent back to S-100 BIOS
TIMEOUT$ERROR   EQU     3DH             ;Error flag to signify the previous command timed out

STD8IBM         EQU     1               ;IBM 8" SDSS Disk

        ; INCLUDE CP/M 3.0 DISK DEFINITION MACROS:
        MACLIB  CPM3
        MACLIB  Z80

;       CODE BEGINS HERE:

        JMP     BOOT                    ;INITIAL ENTRY ON COLD START
        JMP     WBOOT                   ;REENTRY ON PROGRAM EXIT, WARM START
        JMP     CONST                   ;RETURN CONSOLE INPUT STATUS
        JMP     CONIN                   ;RETURN CONSOLE INPUT CHARACTER
        JMP     CONOUT                  ;SEND CONSOLE OUTPUT CHARACTER
        JMP     LIST                    ;SEND LIST OUTPUT CHARACTER
        JMP     AUXOUT                  ;SEND AUXILLIARY OUTPUT CHARACTER
        JMP     AUXIN                   ;RETURN AUXILLIARY INPUT CHARACTER
        JMP     HOME                    ;SET DISKS TO LOGICAL HOME
        JMP     SELDSK                  ;SELECT DISK DRIVE, RETURN DISK PARAMETER INFO
        JMP     SETTRK                  ;SET DISK TRACK
        JMP     SETSEC                  ;SET DISK SECTOR
        JMP     SETDMA                  ;SET DISK I/O MEMORY ADDRESS
        JMP     READ                    ;<------------ READ PHYSICAL BLOCK(S)
        JMP     FWRITEA                 ;WRITE PHYSICAL BLOCK(S)
        JMP     LISTST                  ;RETURN LIST DEVICE STATUS
        JMP     SECTRN                  ;TRANSLATE LOGICAL TO PHYSICAL SECTOR
        JMP     CONOST                  ;RETURN CONSOLE OUTPUT STATUS
```

```
        JMP     AUXIST              ;RETURN AUXILLIARY INPUT STATUS
        JMP     AUXOST              ;RETURN AUXILLIARY OUTPUT STATUS
        JMP     DEVTBL              ;RETURN ADDRESS OF DEVICE DEFINITION TABLE
        JMP     ?CINIT              ;CHANGE BAUD RATE OF DEVICE
        JMP     GETDRV              ;RETURN ADDRESS OF DISK DRIVE TABLE
        JMP     MULTIO              ;SET MULTIPLE RECORD COUNT FOR DISK I/O
        JMP     FLUSH               ;FLUSH BIOS MAINTAINED DISK CACHING
        JMP     ?MOVE               ;BLOCK MOVE MEMORY TO MEMORY
        JMP     ?TIME               ;SIGNAL TIME AND DATE OPERATION
        JMP     BNKSEL              ;SEL BANK FOR CODE EXECUTION AND DEFAULT DMA
        JMP     SETBNK              ;SELECT DIFFERENT BANK FOR DISK I/O DMA OPS.
        JMP     ?XMOVE              ;SET SOURCE AND DEST. BANKS FOR ONE OPERATION
        JMP     0                   ;RESERVED FOR FUTURE EXPANSION
        JMP     0                   ;     DITTO
        JMP     0                   ;     DITTO


CONST:  RET                         ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

LISTST: RET                         ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

AUXIST: RET                         ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

AUXOST: RET                         ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

FLUSH:  XRA     A                   ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:
        RET                         ; RETURN A FALSE STATUS

LIST:   RET                         ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

AUXOUT: RET                         ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

DEVTBL: RET                         ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

?CINIT: RET                         ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

MULTIO: RET                         ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

?TIME:  RET                         ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

BNKSEL: RET                         ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

SETBNK: RET                         ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

?XMOVE: RET                         ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

CONIN:  MVI   A,'Z'-40H                     ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:
        RET

AUXIN:  MVI   A,'Z'-40H                     ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:
        RET

CONOUT: CALL    CONOST              ; ROUTINE OUTPUTS A CHARACTER IN [C] TO THE CONSOLE:
        JRZ     CONOUT
        MOV     A,C
        CPI     0                   ; SD BOARD DOES NOT LIKE NULLS
        RZ
        OUT     CRTOUT
        RET

CONOST: IN      CRTSTAT             ; RETURN CONSOLE OUTPUT STATUS:
        ANI     04H
        RZ                          ;0 IF NOT READY
        XRA     A
        DCR     A
        RET

?MOVE:  XCHG
        LDIR
        XCHG
        RET

SELDSK: LXI     H,DPH0              ; RETURN DPH ADDRESS FOR DRIVE A:
        RET

HOME:   LXI     B,0                 ; HOME SELECTED DRIVE -- TREAT AS SETTRK(0):

SETTRK: SBCD    @TRK                ; ROUTINE SETS TRACK TO ACCESS ON NEXT READ
        RET

SETSEC: SBCD    @SECT               ; ROUTINE SETS SECTOR TO ACCESS ON NEXT READ
        RET
```

```
SETDMA: SBCD    @DMA                    ; ROUTINE SETS DISK MEMORY ADDRESS FOR READ
        RET
                                        ;Arrive here with [BC] = Sec #,  [DE] Translation table address.
SECTRN: XCHG                            ;Swap [DE] (Translation table address) with [HL] (Sec#),  ([HL] now has address)
        DAD     B                       ;Add [BC] to [HL]
        MOV     L,M
        MVI     H,0                     ;Return with actual sector # in [HL]
        RET

GETDRV: LXI     H,@DTBL                 ; RETURN ADDRESS OF DISK DRIVE TABLE:
        RET

FINITA: RET                             ; ROUTINE HAS NO FUNCTION IN LOADER BIOS:

FLOGA:  RET                             ; DOES NOT SEEM TO BE USED IN LOADER BIOS

FWRITEA:XRA     A                       ; RETURN GOOD RESULT CODE
        RET


WBOOT:  RET                             ; WARM BOOT IS NOT USED IN LOADER BIOS

        ;;;;; BOOT
                                        ; ROUTINE DOES COLD BOOT INITIALIZATION
                                        ; (Used here to setup RAM banks and remove EPROM)

BOOT:                                   ;IF BANKED, we need to setup memory mapping

        LDA     BANK$FLAG               ;(RAM location 46H), Z if a non-banked CPMLDR file is to be run
        ORA     A                       ;NZ for a banked CPMLDR (the CPMLDR does not use this byte)
        JZ       RESET$ZFDC
                                        ;Note cannot seem to get CONOUT calls to work at this point
                                        ;seem things/stack? are not yet valid with CPM.
;=============================================================================================

        ; LETS RELOCATE OUR MEMORY IMAGE UP TO THE 10000H-17FFFH MEMORY
        ; REGION FOR EXECUTION -- CP/M 3.0 BANK 0 WILL BE THAT EXTENDED
        ; ADDRESS REGION AND THE TPA WILL BE PART OF THE NORMAL LOWER 64K

        MVI     A,11H                   ;<--- BIT 0 IS TO REMOVE THE EPROM AT 0F000H
        OUT     MPURR1                  ;THIS RELOCATES THE UPPER WINDOW TO 10000H-13FFFH
        LXI     B,2000H                 ;WE WILL MOVE 8K BYTES
        LXI     H,0                     ;STARTING FROM 00000H
        LXI     D,4000H                 ; UP TO 10000H
        LDIR                            ;Z-80 BLOCK MOVE
        MVI     A,11H
        OUT     MPURR0                  ;SWITCH OURSELVES IN TO THAT WINDOW.
                                        ;The CPMLOADER is now actully at 10100H upwards.
        ADI     4                       ;AND MAKE THE UPPER WINDOW CONTIGUOUS SO WE HAVE 16K STARTING
        OUT     MPURR1                  ;AT 10000H in RAM (i.e 10000H-17FFFH).
;=============================================================================================

RESET$ZFDC:                             ;Reset or Initilize the ZFDC Board hardware, just in case!
        OUT     RESET$ZFDC$PORT                 ;Do a hardware reset. Does not matter what is in [A]

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

        IN      S100$DATA$B             ;Check the board is there
        CPI     CMD$HANDSHAKE           ;Make sure we get HANDSHAKE byte back
        JNZ     BAD$ZFDC                ;If not there, just abort

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

        MVI     C,CMD$SET$FORMAT        ;Send Set Disk Format to Drive CMD
        CALL    S100OUT
        MVI     C,0                     ;Floppy Drive 0, (ZFDC Board expects a 0H, 1H, 2H or 3H)
        CALL    S100OUT
        MVI     C,STD8IBM               ;ZFDC Board expects a Disk Format Table Number (0,1,2...13H)
        CALL    S100OUT
```

```
        CALL    WAIT$FOR$ACK            ;Return Z (and NO$ERRORS$FLAG in [A]), or NZ with error # in [A]
        XRA     A
        LXI     H,SET$FORMAT$ERROR
        JNZ     ZFDC$ERROR              ;If not there, just abort

        LXI     H,NBYTES                ;128 bytes always, Size of a sector on this disk format
        SHLD    @SECTOR$BYTE$COUNT      ;Bytes/sector for this disk format

        MVI     C,CMD$SET$DRIVE                 ;Send a "Set Drive CMD" to ZFDC board
        CALL    S100OUT
        MOV     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]
        LXI     H,SET$DRIVE$ERROR
        JNZ     ZFDC$ERROR              ;If not there, just abort
        XRA     A                       ;Return Z if no error
        RET

BAD$ZFDC:
        LXI     H,INIT$ERROR
        CALL    SPECIAL$PMSG            ;Note we cannot use the normal @PMSG BIOS call. It appears not to be valid yet
        HLT                             ;Cannot recover easily, banks may be screwed up, just HALT



;       READ A SECTOR                   ;READ A SECTOR

READ:   SSPD    OLDSTACK                ;See bottom of this module
        LXI     SP,NEWSTACK
        CALL    RDSC                    ;No need for retrys,reseeks etc. The ZFDC board will do this
        LSPD    OLDSTACK                ;VIP!
        RET


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

        MVI     C,CMD$SET$SECTOR        ;Sector # to side A (or for DS disks also side B)
        CALL    S100OUT
        LDA     @SECT                   ;SKEW6: SKEW    26,6,0<----
        INR     A                       ;Disk sectors 1...MAXSEC
        MOV     C,A
        CALL    S100OUT                 ;Send Selected sector HEX number
        CALL    WAIT$FOR$ACK            ;Return Z (and NO$ERRORS$FLAG in [A]), or NZ with error # in [A]
        JNZ     READ$ERROR

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

        MVI     C,CMD$READ$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
        JNZ     READ$ERROR

        LHLD    @DMA                    ;Get DMA address
        LDED    @SECTOR$BYTE$COUNT      ;Bytes/sector for this disk format (128,256,512 or 1024)

RD$SEC: MVI     B,0FFH                  ;Put in a timeout count (Loop for status read at most 256 times)
RD$SEC1:DCR     B
        JNZ     RD$SEC2
        MVI     A,TIMEOUT$ERROR                 ;Send Timeout error
        JMP     READ$ERROR              ;Note JMP back to DSEG bank!

                                        ;Note: we cannot use S100IN here since we are no longer in the DSEG bank
RD$SEC2:IN      S100$STATUS$B           ;Check if ZFDC has any data for S-100 system
        BIT     DIRECTION$BIT,A                 ;Is ZFDC in input mode, if not wait.
        JZ      RD$SEC1                 ;If low then ZFDC board is still in input mode, wait
        BIT     DATA$IN$RDY,A           ;Is there a character available
        JZ      RD$SEC1

        IN      S100$DATA$A             ;Input byte in [A] from ZFDC port
        MOV     M,A                     ;Store it at [@DMA]
        INX     H                       ;[HL++] for [DE--] bytes in sector
        DCX     D
```

```
        MOV     A,E
        ORA     D
        JNZ     RD$SEC                  ;Next Byte, reset timeout count
                                        ;Else fall through to CHECK$RD
CHECK$RD:
        CALL    WAIT$FOR$ACK            ;Return Z (and NO$ERRORS$FLAG in [A]), or NZ with error # in [A]
        JRNZ    READ$ERROR
        RET

READ$ERROR:
        LXI     H,SEC$ZFDC$ERROR
ZFDC$ERROR:
        MOV     A,M
        INX     H
        ORA     A
        JZ      ERROR$DONE
        MOV     C,A
        CALL    CONOUT
        JP      ZFDC$ERROR
ERROR$DONE:
        XRA     A
        DCR     A                       ;RETURN NZ TO INDICATE AN ERROR
        RET


;============== SUPPORT ROUTINES FOR HARDWARE ================================
;
S100STAT:IN     S100$STATUS$B           ;Check if ZFDC has any data for S-100 system
        BIT     DATA$IN$RDY,A           ;Anything there ?
        RZ                              ;Return 0 if nothing
        XRA     A
        DCR     A                       ;Return NZ, & 0FFH in A if something there
        RET

S100IN:
        IN      S100$STATUS$B           ;Check if ZFDC has any data for S-100 system
        BIT     DIRECTION$BIT,A                 ;Is ZFDC in input mode, if not wait
        JZ      S100IN                  ;If low then ZFDC board is still in input mode, wait
        BIT     DATA$IN$RDY,A
        JZ      S100IN
        IN      S100$DATA$A             ;return with character in A
        RET
;
S100OUT:
        IN      S100$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
        JNZ     S100OUT
        BIT     DATA$OUT$RDY,A          ;Has previous (if any) character been read.
        JZ       S100OUT                        ;Z if not yet ready
        MOV     A,C
        OUT     S100$DATA$B
        RET

WAIT$FOR$ACK:                           ;Delay to wait for ZFDC to return data. There is a timeout of about 2 sec.
        PUSH    B                       ;This can be increased if you are displaying debugging info on the ZFDC
        PUSH    D                       ;HEX LED display.
        LXI     B,0
        MVI     E,STATUS$DELAY          ;Timeout, (about 2 seconds)
WAIT$1: IN      S100$STATUS$B           ;Check if ZFDC has any data for S-100 system
        BIT     DIRECTION$BIT,A                 ;Is ZFDC in input mode
        JZ      WAIT$2                  ;if low then ZFDC is still in input mode
        CALL    S100STAT                ;Wait until ZFDC Board sends something
        JZ      WAIT$2
        CALL    S100IN                  ;Get returned Error # (Note this releases the SEND$DATA routine on the ZFDC
board)
        CPI     NO$ERRORS$FLAG          ;Was SEND$OK/NO$ERRORS$FLAG sent back from ZFDC Board
        POP     DE                      ;Balance up stack
        POP     BC
        RET                             ;Return NZ if problem, Z if no problem
WAIT$2: DCR     B
        JNZ     WAIT$1                  ;Try for ~2 seconds
        DCR     B                       ;Reset B to 0FFH
        DCR     C
        JNZ     WAIT$1
        DCR     B                       ;Reset B to 0FFH
        DCR     C
        DCR     E
        JNZ     WAIT$1
        XRA     A
        DCR     A
        POP     D                       ;Balance up stack
        POP     B
```

```
        RET                             ;Return NZ flag set if timeout AND 0FFH in [A]


;Print the acccumulator value on CRT in HEX-ASCII
PACC:   PUSH    PSW
        PUSH    B

        PUSH    PSW
        RRC
        RRC
        RRC
        RRC
        CALL    ZCONV
        POP     PSW
        CALL    ZCONV
        POP     B
        POP     PSW
        RET

ZCONV:  ANI     0FH             ;HEX to ASCII
        ADI     90H
        DAA
        ACI     40H
        DAA
        MOV     C,A
        CALL    CONOUT
        RET

SPECIAL$PMSG:                           ;Cannot use @PMSG in LOADERBIOS
        MOV     A,M
        INX     H
        CPI     '$'
        RZ
        MOV     C,A
        CALL    CONOUT                  ;Hardware send to consol
        JMP     SPECIAL$PMSG



INIT$ERROR      DB      CR,LF,'Within CPMLDR, Error Initilizing the ZFDC Board.',0
SET$FORMAT$ERROR DB     CR,LF,'Within CPMLDR, Set Disk Format Error.',0
SET$DRIVE$ERROR         DB      CR,LF,'Within CPMLDR, Set Drive Error.',0
SEC$ZFDC$ERROR DB       CR,LF,'Within CPMLDR, Sector Read Error (On ZFDC Board).',0
LOADING$B$CPM  DB       CR,LF,'Loading BANKED Version of CPMLDR.',0
LOADING$NB$CPM DB       CR,LF,'Loading NON-BANKED Version of CPMLDR.',0


@TRK:           DW      0               ;2 BYTES FOR NEXT TRACK TO READ OR WRITE
@DMA:           DW      0               ;2 BYTES FOR NEXT DMA ADDRESS
@SECT           DW      0               ;2 BYTES FOR SECTOR
@SECTOR$BYTE$COUNT DW   0               ;2 BYTES FOR Bytes/Sector

OLDSTACK:DW     0
         DS     40
NEWSTACK:DW     0

                        ; DISK DRIVE TABLE:
@DTBL:  DW      DPH0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0


DPH0:   DPH     SKEW6,FLOPPY$DPB,16,31

        ; MAKE SURE DPB'S ARE IN COMMON MEMORY:
                                        ;IF     BANKED
        CSEG

FLOPPY$DPB:
        DPB     128,26,77,1024,64,2

SKEW6:  SKEW    26,6,0          ;<-- Note start with sector 0
        DB      '<----- END OF LOADER BIOS  >' ;Useful for debugging
```