```
;THIS IS A BIOS FOR CPM-86 USEING THE ZFDC S100 FDC CONTROLLER BOARD
;      AUTHOR:      JOHN J. MONAHAN           monahan@vitasoft.org             6/5/2011

;      V1.0                       ;Initial version    6/5/2100


CR     EQU    0DH                 ;CARRIAGE RETURN
LF     EQU    0AH                 ;LINE FEED
BELL   EQU    7                   ;DING

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

S100DATAA     EQU    10H          ;IN,  S100 Data port to GET data to from FDC Board
S100DATAB     EQU    10H          ;OUT, S100 Data port to SEND data to FDC Board
S100STATUSA   EQU    11H          ;Status port for A
S100STATUSB   EQU    11H          ;Status port for B
RESETZFDCPORT EQU    13H          ;Port to reset ZFDC Z80 CPU.

STATUSDELAY   EQU    10           ;Time-out for waiting for ZFDC Board handshake signal (Now, ~0.5 seconds @ 10MHz)

ZFDCUNINITILIZED EQU       0FFH         ;If ZFDC is not yet initilized
ZFDCNOTWORKING      EQU    0FEH         ;If ZFDC is not working
ZFDCNOTPRESENT      EQU    0FDH         ;If ZFDC board is absent
ZFDCINITILIZED      EQU    000H         ;If ZFDC is initilized OK

STD8IBM             EQU    1            ;IBM 8" SDSS Diak


;Commands to the ZFDC Board (Only a few are used here):-

CMDPIOTEST    EQU    0H           ;Simple loop hardware test of PIO #1 Ports
CMDMONITOR    EQU    1H           ;Jump to internal monitor here.
CMDSHOWSIGNON EQU    2H           ;This will "rotate" in the Sector Display TIL's as a hardware test
CMDRESETZFDC  EQU    3H           ;Reset the WD2793 chip and Board software

CMDSETFORMAT  EQU    4H           ;This will select a specified drive and assign a disk format table to that drive
CMDSETDRIVE   EQU    5H           ;This will select a specified drive (0,1,2,3)
CMDGETFORMAT  EQU    6H           ;Return to S100 System the current Disk paramater format table number
CMDSETTRACK   EQU    7H           ;This will set head request to a specified track
CMDSETSIDE    EQU    8H           ;This will set side request to a specified side
CMDSETSECTOR  EQU    9H           ;This will set sector request to a specified sector

CMDSETHOME    EQU    0AH          ;This will set head request to Track 0 of CURRENT drive
CMDSTEPIN     EQU    0BH          ;Step head in one track of CURRENT drive
CMDSTEPOUT    EQU    0CH          ;Step head out one track of CURRENT drive
CMDSEEKNV     EQU    0DH          ;Seek to track with NO verify of CURRENT drive

CMDSEEKTRACK  EQU    0EH          ;Seek to track to (IY+DRIVETRACK) with the track verify bit set on CURRENT drive/format
CMDGETTRACKID EQU    0FH          ;Read the CURRENT TRACK ID
```

```
CMDREADSECTOR EQU    10H          ;Read data from the CURRENT sector (on current track,side,drive).
CMDWRITESECTOR  EQU 11H           ;Write data to the CURRENT sector (on current track,side,drive).

CMDGETWDTRACK EQU    12H          ;Get the WD2793 Track register value
CMDGETWDSECTOR      EQU    13H          ;Get the WD2793 Sector register value
CMDGETWDSTATUS      EQU    14H          ;Get the WD2793 Status register value

CMDTRACKDUMP  EQU    15H          ;Dump complete CURRENT track to S-100 system
CMDFORMATTRACK      EQU    16H          ;Format the disk in the of the CURRENT drive using the current format assigned to that disk

CMDSETDEBUGON EQU    17H          ;Turn on Debug display mode
CMDSETDEBUGOFF      EQU    18H          ;Turn off Debug display mode
CMDRAMDUMP    EQU    19H          ;Command to pass back to S-100 system all memory variables and flags on ZFDC board

CMDABORT      EQU    20H          ;Generalized Abort of the current process command.
CMDHANDSHAKE  EQU    21H          ;Handshake command only sent during board initilization/testing
CMDGETDRIVE   EQU    22H          ;Get the current selected drive number (0..3)
CMDSETTRACKDS EQU    23H          ;Set Track (If a DS Disk, EVEN tracks on Side A, ODD tracks on Side B. Used by CPM)
CMDGETERRORSTRING EQU 24H         ;Return a string explaining the last Error Code sent
CMDCHECKDRIVE EQU    28H          ;Check there is a valid drive present on specified drive (0,1,2,3)

;Possible ERROR codes returned from the ZFDC Board (Not a complete List):-

NOERRORSFLAG  EQU    00H          ;No Errors flag for previous cmd, sent back to S-100 BIOS
ZFDCABSENT    EQU    3BH          ;If ZFDC Board is absent
ZFDCINITERROR EQU    3CH          ;If ZFDC initilization error
TIMEOUTERROR  EQU    3DH          ;Error flag to signify the previous command timed out
DRIVEINACTIVE EQU    3AH          ;Drive is inactive
DRIVEDOOR     EQU    3BH          ;Drive door open
DISKWPERR     EQU    31H          ;Sector write error, Disk is write protected



;-------- PORTS USED IN IO DRIVERS FOR CONSOLE ETC.------------------

CONSTAT       EQU    0H                ;PORT TO SEE IF ANYTHING AT KEYBOARD
CONOUT EQU    1H
CONIN  EQU    1H

CENTOUT       EQU    5H                ;CENTRONICS PRINTER PORT
CENTSTAT EQU  5H
CENTSTROBE EQU       4H




       CSEG
```

```
       ORG    0H
CCP:
;
;=============================================================================
;
;      THE CPM-86 CCP AND BDOS WILL GO HERE ( ABSOLUTE 500H TO 25FFH)
;
;             NOTE WITH THIS 8089 VERSION I HAVE EVERYTHING 100H HIGHER
;             THAN EVERYTHING IN THE DIGITAL RESEARCH MANUAL.
;             SO THE CCP AND BDOS WILL BE FROM 500H TO 25FFH AND 8089
;             TABLE AT 400H TO 500H.
;
;=============================================================================
;
       ORG    2500H               ;NOW THE START OF THE CUSTOM BIOS
                                  ;NOTE ACTUAL ADDRESS IS 2A00H BECAUSE
                                  ;OF [CS] OFFSET
;      ---JUMP TABLE---
;
CPMINIT:JMP   INIT                ;  0 - COLD BOOT
WBX:   JMP    WBOOT               ;  1 - WARM BOOT
       JMP    CSTS                ;  2 - CONSOLE STATUS REQUEST
ZCI:   JMP    CI                  ;  3 - CONSOLE INPUT
ZCO:   JMP    CO                  ;  4 - CONSOLE OUTPUT
ZLO:   JMP    LO                  ;  5 - LIST OUTPUT
       JMP    POO                 ;  6 - PUNCH OUTPUT
       JMP    RI                  ;  7 - READER INPUT
       JMP    HOME                ;  8 - TRACK ZERO SEEK
       JMP    SETDR               ;  9 - SET DRIVE #
       JMP    SETTK               ; 10 - SET TRACK ADR
       JMP    SETSEC              ; 11 - SET SECTOR ADR
       JMP    SETDMA_OFFSET       ; 12 - SET BUFFER ADDRESS
       JMP    READ                ; 13 - READ A SECTOR
       JMP    WRITE               ; 14 - WRITE A SECTOR
ZLISTS:       JMP    LSTAT              ; 15 - LIST OUTPUT READY TEST
       JMP    SXR                 ; 16 - SECTOR XLATE ROUTINE
       JMP    SETDMA_SEG          ; 17 - SET SEG BASE FOR BUFFER
       JMP    GETSEGT                   ; 18 - GET MEM DESC TABLE OFFSET
       JMP    GETIOBF                   ; 19 - RETURN IO BYTE
       JMP    SETIOBF                   ; 20 - SET IO BYTE

       DB     '<----- End of XBIOS JMP Table  ' ;For debugging purposes



;             =====================
;             ** CBIOS FUNCTIONS **
;             =====================
```

```
;              ---COLD BOOT---

INIT:  MOV    AX,CS
       MOV    SS,AX
       MOV    DS,AX
       MOV    ES,AX
       MOV    SP, OFFSET STKBASE  ;USE A LOCAL STACK
       CLD
       PUSH   DS
       MOV    AX,0
       MOV    DS,AX
       MOV    ES,AX
       MOV    INT0OFFSET, OFFSET INTTRAP ;INT0 TO ADDRESS TRAP ROUTINE
       MOV    INT0SEGMENT, CS
       MOV    DI,4
       MOV    SI,0
       MOV    CX,510              ;TRAP VECTOR TO ALL 256 INTS
   REP MOVS   AX,AX
       MOV    BDOSOFFSET, 0B06H   ;BDOS OFFSET TO PROPER INT0
       MOV    BDOSSEGMENT,CS
       MOV    INT0OFFSET, OFFSET INT0TRAP
       MOV    INT4OFFSET, OFFSET INT4TRAP
       POP    DS


       MOV    BX,OFFSET STORAGE   ;CLEAR RAM STORAGE AREA
       MOV    CH, FLGSIZ
       XOR    AL,AL
INIT1: MOV    BYTE PTR [BX],AL    ;CLEAR FLAGS & VARIABLES
       INC    BX
       DEC    CH
       JNZ    INIT1
       MOV    BYTE PTR IOBYT, AL  ;CLEAR IOBYTE

       OUT    CENTSTROBE,AL       ;CLEAR PRINTER PORT JUST IN CASE

       CALL   RESETZFDC           ;<<<<< Initilize the ZFDC Board >>>>>>>>>>

       MOV    WORD PTR DMA_SEGMENT,CS    ;SET DEFAULT SEGMENT DMA TO HERE

       MOV    BX,OFFSET SIGNON
       CALL   PMSG
       MOV    CL,0                ;DEFAULT TO DRIVE A:
       JMP    CCP

;------------------ WARM BOOT-------------------------------------------------
;
WBOOT: JMP    CCP + 6                   ;GOTO CPM
```

```
;------------------- INT TRAP ROUTINES -------------------------------------
INT0TRAP:
       CLI
       MOV    BX,OFFSET INT0TRP   ;DIVIDE TRAP HALT
       JMPS   INTHALT

INT4TRAP:
       CLI
       MOV    BX,OFFSET INT4TRP   ;OVERFLOW TRAP HALT
       JMPS   INTHALT

INTTRAP:
       CLI
       MOV    BX,OFFSET INTTRP    ;INTERRUPT TRAP HALT

INTHALT:MOV   AX,CS
       MOV    DS,AX
       CALL   PMSG
       POP    BX                  ;GET SEGMENT
       POP    AX                  ;PRINT SEGMENT
       PUSH   BX
       CALL   PHEX
       MOV    CL,':'
       CALL   ZCO                 ;PRINT OFFSET
       POP    AX
       CALL   PHEX
       HLT                        ;HOLD EVERYTHING
;
PHEX:  PUSH   AX                  ;Print [AX] (CX destroyed)
       MOV    AL,AH
       CALL   PHXB
       POP    AX
PHXB:  PUSH   AX
       MOV    CL,4
       SHR    AL,CL
       CALL   PHXD
       POP    AX
       AND    AL,0FH              ;ISOLATE LOWER NIBBLE
PHXD:  ADD    AL,90H              ;DISPLAY A NIBBLE
       DAA
       ADC    AL,40H
       DAA
       MOV    CL,AL
       CALL   ZCO
       RET
;
GETIOBF:
       MOV    AL,IOBYT
       RET
```

```
SETIOBF:
       MOV    IOBYT,CL
       RET


PMSG:  MOV    AL,[BX]                  ;PRINT A STRING
       TEST   AL,AL
       JZ     RETURN
       MOV    CL,AL
       CALL   ZCO
       INC    BX
       JMP    PMSG
RETURN:       RET


;--------------- SECTOR TRANSLATE ROUTINE -------------------------------
SXR:   TEST   DX,DX
       JNZ    SXR1               ;IF Z THEN NO TRANSLATION IS REQ
       MOV    BX,CX
       RET

SXR1:  MOV    BX,CX              ;TRANS SEC [CX] USING TABLE AT [DX]
       ADD    BX,DX              ;WILL HAVE NO TRANSLATION FOR MDISK
       MOV    BL,[BX]
       RET

;--------------- HOME ---------------------------------------------------
HOME:  MOV    CX,0               ;RETURN ZERO JUST IN CASE
       JMP    SETTK

;-------------- SET DRIVE NUMBER ----------------------------------------
SETDR: MOV    BYTE PTR RRDSK,CL  ;SAVE IT
       CMP    CL,1               ;First or second floppy drive
       JZ     DRIVE1
       MOV    BX,OFFSET DPE0           ;Don't actully select the drive yet
       RET                       ;Too much disk trashing if you do

DRIVE1: MOV   BX,OFFSET DPE1
       RET


;--------------- SET TRACK ADDRESS --------------------------------------
SETTK: MOV    WORD PTR RRTRK,CX  ;SAVE TRACK ADR
       RET

;--------------- SET SECTOR ADR -----------------------------------------
SETSEC:       MOV    WORD PTR RRSEC,CX   ;SAVE SECTOR ADR (NEED CX BECAUSE THE
       RET                       ;MDISK HAS UP TO 480 SECTORS/"TRACK"
```

```
;----------------- SET DMA ADDRESS ---------------------------------------
SETDMA_OFFSET:
       MOV    WORD PTR DMA_OFFSET,CX     ;SAVE DMA ADR
       RET

;------------------ SET DMA SEGMENT ---------------------------------------
SETDMA_SEG:
       MOV    WORD PTR DMA_SEGMENT,CX
       RET

;------------------ GET MEMORY MAP ----------------------------------------
GETSEGT:MOV   BX, OFFSET SEGTABLE
       RET

;-------------- READ SECTOR -------------------------------------------
READ:  CALL   READSECTOR          ;<<<<<<< Read sector on Floppy disk >>>>>>>>>>
       RET

;--------------- WRITE -------------------------------------------------
WRITE: CALL   WRITESECTOR         ;<<<<<<<<<< Write sector to floppy >>>>>>>>>>>>>
       RET



;<<<<<<<<<<<<<<<<<<<<<< ZFDC CORE ROUTINES >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
RESETZFDC:
       OUT    RESETZFDCPORT,AL    ;Do a hardware reset. Does not matter what is in [AL]

       MOV    AX,5                ;~0.5 second at 10 MHz
       MOV    CX,0                ;Delay to allow board to setup hardware
WAITD: LOOP   WAITD               ;Delay for ~0.5 seconds
       DEC    AX
       JNZ     WAITD

       IN     AL,S100DATAB        ;Check the board is there
       CMP    AL,CMDHANDSHAKE            ;Make sure we get HANDSHAKE byte back
       MOV    AL,ZFDCABSENT       ;If not then no ZFDC board present
       MOV    BX,Offset INITFLAG
       MOV    [BX],AL
       JNZ    BADZFDC                    ;If not there, just abort

       MOV    AL,CMDHANDSHAKE            ;Send another byte just to be sure.
       OUT    S100DATAB,AL        ;This clears up ints on ZFDC board
       CALL   WAITFORACK          ;Wait to make sure all is well.

       OR     AL,AL
       MOV    AL,ZFDCINITERROR    ;If not then no ZFDC board present
       MOV    BX,Offset INITFLAG
       MOV    [BX],AL                    ;Flag the board as having an error
```

```
        JNZ     BADZFDC                 ;and abort

        MOV     AL,ZFDCINITILIZED       ;Flag the ZFDC is initilized
        MOV     BX,Offset INITFLAG
        MOV     [BX],AL                 ;Flag the board as having an error

SETDISKFORMATS:                         ;Do reverse order so we end up with Drive 0 selected
        MOV     CL,CMDSETFORMAT         ;Send Set Disk Format to Drive CMD
        CALL    S100OUT
        MOV     CL,1                    ;Floppy Drive 1, (ZFDC Board expects a 0H, 1H, 2H or 3H)
        CALL    S100OUT
        MOV     CL,STD8IBM              ;ZFDC Board expects a Disk Format Table Number (0,1,2...13H)
        CALL    S100OUT
        CALL    WAITFORACK              ;Return Z (and NOERRORSFLAG in [A]), or NZ with error # in [A]

        MOV     CL,CMDSETFORMAT         ;Send Set Disk Format to Drive CMD
        CALL    S100OUT
        MOV     CL,0                    ;Floppy Drive 0, (ZFDC Board expects a 0H, 1H, 2H or 3H)
        CALL    S100OUT
        MOV     CL,STD8IBM              ;ZFDC Board expects a Disk Format Table Number (0,1,2...13H)
        CALL    S100OUT
        CALL    WAITFORACK              ;Return Z (and NOERRORSFLAG in [A]), or NZ with error # in [A]
        XOR     AL,AL
        RET                             ;Return Z

BADZFDC:
        MOV     BX,OFFSET ZFDCRESETERROR
        CALL    PMSG
        XOR     AL,AL
        DEC     AL
        RET                             ;Return NZ



ZFDCDRIVELOGIN:
        MOV     CL,CMD_GET_DRIVE        ;Send a "Get Current Drive CMD" to ZFDC board
        CALL    S100OUT
        CALL    S100IN                  ;Note potential to lockup here (but unlightly)
        MOV     AH,AL                   ;ZFDC Board will return a 1 byte drive number in [AL]
        CALL    WAITFORACK              ;Return Z (and NOERRORSFLAG in [AL]), or NZ with error # in [AL]
        JNZ     LOGINERROR
        CMP     AH,BYTE PTR RRDSK       ;Is it the same as the current disk
        JZ      SAME_DRIVE

        MOV     CL,CMDSETDRIVE          ;Send a "Set Drive CMD" to ZFDC board
        CALL    S100OUT
        MOV     CL,BYTE PTR RRDSK       ;Get Requested Disk
        CALL    S100OUT

        CALL    WAITFORACK              ;Return Z (and NOERRORSFLAG in [AL]), or NZ with error # in [AL]
```

```
       JNZ    LOGINERROR

       MOV    CL,CMDCHECKDRIVE    ;Send a "Check a Drive is valid" to ZFDC board
       CALL   S100OUT
       MOV    CL,BYTE PTR RRDSK   ;Get Requested Disk
       CALL   S100OUT
       CALL   WAITFORACK          ;Return Z (and NOERRORSFLAG in [A]), or NZ with error # in [A]
       JNZ    LOGINERROR
SAME_DRIVE:
       RET                        ;Return Z as no error

LOGINERROR:                       ;If a problem logging into a drive return an error to the system
       CMP    AL,DRIVEINACTIVE    ;"No detected disk in drive"
       JNZ    LOGINERROR2
       MOV    BX,OFFSET FLOPPYNODISKERR
       CALL   PMSG
       JMP    LOGINERROR4
LOGINERROR2:
       CMP    AL,DRIVEDOOR        ;"Drive Door Open"
       JNZ    LOGINERROR3
       MOV    BX,OFFSET FLOPPYDOORERR
       CALL   PMSG
       JMP    LOGINERROR4
LOGINERROR3:
       MOV    BX,OFFSET FLOPPYLOGINERR ;"Unknown drive error"
       CALL   PMSG                ;Write error message
LOGINERROR4:
       MOV    BX,0                ;RIF INVALID DRIVE # WITH [BX]=0
       RET


;-------------------------- READ SECTOR ---------------------------------
READSECTOR:
       MOV    CL,CMDSETDRIVE             ;<<< Set Drive.     Will just return if current drive
       CALL   S100OUT                    ;We need this each time because commands like PIP do not Login a drive for each sector
       MOV    CL,BYTE PTR RRDSK   ;Get Requested Disk
       CALL   S100OUT
       CALL   WAITFORACK          ;Return Z (and NOERRORSFLAG in [AL]), or NZ with error # in [AL]
       JZ     RS1
       JMP    READERROR

RS1:   MOV    CL,CMDSETTRACK             ;<<< Set Track
       CALL   S100OUT
       MOV    CL,BYTE PTR RRTRK   ;Get Requested Disk (Only CL used here)
       CALL   S100OUT                    ;Send Selected track HEX number
       CALL   WAITFORACK          ;Wait for NOERRORSFLAG to come back
       JZ     RS2
       JMP    READERROR

RS2:   MOV    CL,CMDSETSECTOR            ;<<< Set Sector (No INC because this is done with translation table)
```

```
        CALL   S100OUT
        MOV    CL,BYTE PTR RRSEC   ;Get Requested Disk (Only CL used here)
        CALL   S100OUT                    ;Send Selected sector HEX number
        CALL   WAITFORACK          ;Wait for NOERRORSFLAG to come back
        JZ     RS3
        JMP    READERROR

RS3:    MOV    CL,CMDSEEKTRACK            ;<<< Seek to that track if not already there
        CALL   S100OUT
        CALL   WAITFORACK          ;Wait for NOERRORSFLAG to come back
        JZ     RS4
        JMP    READERROR

RS4:    MOV    CL,CMDREADSECTOR    ;<<<< Routine assumes required Drive Table,Drive,(Side),Track, and sector are already sent to board
        CALL   S100OUT
        CALL   WAITFORACK          ;Wait for NOERRORSFLAG to come back
        JZ     RS5
        JMP    READERROR

RS5:    CLI                        ;Just in case
        CLD
        PUSH   ES
        PUSH   DI

        MOV    DI, DS:WORD PTR DMA_OFFSET
        MOV    ES, DS:WORD PTR DMA_SEGMENT      ;GET CORRECT SEGMENT
        MOV    CX,128              ;Sector size in this case always

RDSEC:  MOV    BX,0400H            ;Put in a timeout count (Loop for status read at most 256X4 times)
RDSEC1:        DEC    BX                  ;Dec BC
        JNZ    RDSEC2              ;Will wait 400H times before timing out
        MOV    AL,TIMEOUTERROR             ;Send Timeout error
        JMP    READERROR
                                   ;Note we cannot use S100OUT here since we are no longer in the DSEG bank
RDSEC2:        IN     AL,S100STATUSB             ;Send data to ZFDC output (arrive with byte to be sent in M)
        TEST   AL,80H              ;Is ZFDC in INPUT mode, if not wait
        JZ     RDSEC1
        TEST   AL,01H              ;Has previous (if any) character been read.
        JZ     RDSEC1              ;Z if not yet ready

        IN     AL,S100DATAB        ;Get data
        STOS   AL                  ;>>>>>>>>> READ 1 BYTE BYTE <<<<<<<<<
        LOOP   RDSEC

        POP    DI
        POP    ES                  ;GET BACK OLD VALUE OF [DS]
        STI
        CALL   WAITFORACK          ;Return Z (and NOERRORSFLAG in [A]), or NZ with error # in [A]
        JNZ    READERROR
        RET                        ;Ret Z (from WAITFORACK)
```

```
READERROR:
       MOV    BX,OFFSET FLOPPYREADERR ;"Sector Read Error"
       CALL   PMSG
       CALL   RESETZFDC
       XOR    AL,AL
       INC    AL                  ;Set to NZ & 1 for read error
       RET


;------------------ WRITE SECTOR ----------------------------------------
WRITESECTOR:
       MOV    CL,CMDSETDRIVE             ;<<< Set Drive.     Will just return if current drive
       CALL   S100OUT                    ;We need this each time because commands like PIP do not Login a drive for each sector
       MOV    CL,BYTE PTR RRDSK   ;Get Requested Disk
       CALL   S100OUT
       CALL   WAITFORACK          ;Return Z (and NOERRORSFLAG in [AL]), or NZ with error # in [AL]
       JZ     WS1
       JMP    WRITEERROR

WS1:   MOV    CL,CMDSETTRACK             ;<<< Set Track
       CALL   S100OUT
       MOV    CL,BYTE PTR RRTRK   ;Get Requested Disk (Only CL used here)
       CALL   S100OUT                    ;Send Selected track HEX number
       CALL   WAITFORACK          ;Wait for NOERRORSFLAG to come back
       JZ     WS2
       JMP    WRITEERROR

WS2:   MOV    CL,CMDSETSECTOR            ;<<< Set Sector (No INC because this is done with translation table)
       CALL   S100OUT
       MOV    CL,BYTE PTR RRSEC   ;Get Requested Disk (Only CL used here)
       CALL   S100OUT                    ;Send Selected sector HEX number
       CALL   WAITFORACK          ;Wait for NOERRORSFLAG to come back
       JZ     WS3
       JMP    WRITEERROR

WS3:   MOV    CL,CMDSEEKTRACK            ;<<< Seek to that track if not already there
       CALL   S100OUT
       CALL   WAITFORACK          ;Wait for NOERRORSFLAG to come back
       JZ     WS4
       JMP    WRITEERROR

WS4:   MOV    CL,CMDWRITESECTOR   ;<<<< Routine assumes required Drive Table,Drive,(Side),Track, and sector are already sent to board
       CALL   S100OUT
       CALL   WAITFORACK          ;Wait for NOERRORSFLAG to come back
       JZ     WS5
       JMP    WRITEERROR

WS5:   CLI                        ;Just in case
```

```
        CLD
        PUSH    DS
        PUSH    SI

        MOV     SI, DS:WORD PTR DMA_OFFSET
        MOV     DS, DS:WORD PTR DMA_SEGMENT      ;GET CORRECT SEGMENT
        MOV     CX,128              ;Sector size in this case always

WRSEC:  MOV     BX,0400H            ;Put in a timeout count (Loop for status read at most 256X4 times)
WRSEC1:         DEC     BX                  ;Dec BC
        JNZ     WRSEC2              ;Will wait 400H times before timing out
        MOV     AL,TIMEOUTERROR             ;Send Timeout error
        JMP     WRITEERROR
                                    ;Note we cannot use S100OUT here since we are no longer in the DSEG bank
WRSEC2:         IN      AL,S100STATUSB              ;Send data to ZFDC output (arrive with byte to be sent in M)
        TEST    AL,80H              ;Is ZFDC in OUTPUT mode, if not wait
        JNZ     WRSEC1
        TEST    AL,02H              ;Has previous (if any) character been read.
        JZ      WRSEC1              ;Z if not yet ready

        LODS    AL                  ;>>>>>>>>> WRITE 1 BYTE BYTE <<<<<<<<<
        OUT     S100DATAB,AL        ;Send it
        LOOP    WRSEC

        POP     SI
        POP     DS                  ;GET BACK OLD VALUE OF [DS]
        STI
        CALL    WAITFORACK          ;Return Z (and NOERRORSFLAG in [A]), or NZ with error # in [A]
        JZ      WS6
        JMP     WRITEERROR
WS6:    RET                         ;Ret Z (from WAITFORACK)




WRITEERROR:
        MOV     BX,OFFSET FLOPPYWRITEERR ;"Sector Write Error"
        CMP     AL,DISKWPERR        ;Special case for Write Protected Disk error
        JNZ     SECERROR            ;All other errors
        MOV     BX,OFFSET DISKWPERROR
SECERROR:
        CALL    PMSG
        CALL    RESETZFDC
        XOR     AL,AL
        INC     AL                  ;Set to NZ & 1 for read error
        RET

INVALIDDRIVE:
        MOV     BX,OFFSET INVALIDDRIVEMSG
```

```
       CALL   PMSG
       CALL   RESETZFDC           ;Re-initilize the ZFDC board just in case it is hung
       JMP    SECERROR            ;All other errors


;------------------- ZFDC CORE INTERFACE ROUTINES ----------------------
S100STAT:
       IN     AL,S100STATUSB             ;Check if ZFDC has any data for S-100 system
       TEST   AL,01H              ;Anything there ?
       JZ     S100ST1                    ;Return 0 if nothing
       XOR    AL,AL
       DEC    AL                  ;Return NZ, & 0FFH in AL if something there
S100ST1:RET


S100IN:       IN     AL,S100STATUSB             ;Check if ZFDC has any data for S-100 system
       TEST   AL,80H              ;Is ZFDC in input mode, if not wait
       JZ     S100IN              ;If low then ZFDC board is still in input mode, wait
       TEST   AL,01H
       JZ     S100IN
       IN     AL,S100DATAA        ;return with character in AL
       RET

S100OUT:IN    AL,S100STATUSB             ;Send data to ZFDC output (arrive with character to be sent in C)
       TEST   AL,80H              ;Is ZFDC in output mode, if not wait
       JNZ    S100OUT
       TEST   AL,02H              ;Has previous (if any) character been read.
       JZ      S100OUT                   ;Z if not yet ready
       MOV    AL,CL
       OUT    S100DATAB,AL
       RET

WAITFORACK:                       ;Delay to wait for ZFDC to return data. There is a timeout of about 2 sec.
       PUSH   BX                  ;This can be increased if you are displaying debugging info on the ZFDC
       PUSH   DX                  ;HEX LED display.
       MOV    BX,0
       MOV    DL,STATUSDELAY             ;Timeout, (about 2 seconds)
WAIT1: IN     AL,S100STATUSB             ;Check if ZFDC has any data for S-100 system
       TEST   AL,80H              ;Is ZFDC in input mode
       JZ     WAIT2               ;if low then ZFDC is still in input mode
       CALL   S100STAT            ;Wait until ZFDC Board sends something
       JZ     WAIT2
       CALL   S100IN              ;Get returned Error # (Note this releases the SENDDATA routine on the ZFDC board)
       CMP    AL,NOERRORSFLAG            ;Was SENDOK/NOERRORSFLAG sent back from ZFDC Board
       POP    DX                  ;Balance up stack
       POP    BX
       RET                        ;Return NZ if problem, Z if no problem
WAIT2: DEC    BH
       JNZ    WAIT1               ;Try for ~2 seconds
```

```
       DEC    BH
       DEC    BL
       JNZ    WAIT1
       DEC    BH
       DEC    BL
       DEC    DL
       JNZ    WAIT1
       XOR    AL,AL
       DEC    AL
       POP    DX                  ;Balance up stack
       POP    BX
       RET                        ;Return NZ flag set if timeout AND 0FFH in [A]


;<<<<<<<<<<<<<<<<<<<<<<< MAIN CONSOL OUTPUT ROUTINE >>>>>>>>>>>>>>>>>>>>>>>>>
CO:    IN     AL,CONSTAT          ;SD SYSTEMS OR PROPELLER VIDIO BOARD PORT
       AND    AL,4H
       JZ     CO
       MOV    AL,CL
       OUT    CONOUT,AL
       RET
;
;
;<<<<<<<<<<<<<<<<<<<< MAIN CONSOL STATUS ROUTINE >>>>>>>>>>>>>>>>>>>>>
;
CSTS:  IN     AL,CONSTAT
       AND    AL,02H
       JNZ    CST1
       RET                        ;RETURN WITH 0 IN [AL] IF NOTHING THERE
CST1:  DEC    AL
       RET                        ;RETURN WITH 0FFH IN [AL] IF SOMETHING
;
;
;<<<<<<<<<<<<<<<<<<<< MAIN CONSOL INPUT ROUTINE >>>>>>>>>>>>>>>>>>>>>
;
CI:    CALL   CSTS
       JZ     CI
       IN     AL,CONIN
       AND    AL,7FH              ;Just in case strip parity
       RET
;
;<<<<<<<<<<<<<<<<<<<<<<< MAIN PRINTER STATUS ROUTINE >>>>>>>>>>>>>>>>>>>>>>>>>
LSTAT: IN     AL,CENTSTAT         ;FIRST FIND WHICH PRINTER IS SELECTED
       AND    AL,00001111B        ;XXXX0110 IS READY (BIT 3=PAPER BIT 2=FAULT
       CMP    AL,00000110B        ;BIT 1=SELECT  BIT 0=BUSY
       JZ     LSTAT1
       XOR    AL,AL
       RET
```

```
LSTAT1:       XOR    AL,AL               ;PUT 0FFH IN [A] IF READY & NO ZERO FLAG
       DEC    AL
       RET
;
;<<<<<<<<<<<<<<<<<<<<<< MAIN PRINTER OUTPUT ROUTINE (Char in CL) >>>>>>>>>>>>>>>>>>>>>>>>
LO:    PUSH   CX
       MOV    CX,0FFH
LO2:   CALL   LSTAT               ;Do not hang on status
       JNZ    LO1
       LOOP   LO2
       XOR    AL,AL
       POP    CX
       MOV    AL,CL
       RET                        ;RET Z if problem

LO1:   POP    CX
       MOV    AL,0FFH
       OUT    CENTSTROBE,AL
       MOV    AL,CL
       OUT    CENTOUT,AL
       XOR    AL,AL
       OUT    CENTSTROBE,AL
       MOV    AL,0FFH                    ;Raise strobe again
       OUT    CENTSTROBE,AL
       OR     AL,AL
       RET                        ;Ret NZ if OK


POO:   RET                        ;NO PUNCH OUTPUT AT THE MOMENT
RI:    MOV    AL,1AH              ;NO READER AT THE MOMENT
       RET

LAB57  EQU    $

;---------------------> START OF DATA SEGMENT <----------------------


              DSEG
              ORG OFFSET LAB57


;      ---MESSAGES---

SIGNON              DB     1AH,1H,10H,11H,LF
                    DB     '    512K CP/M-86 V1.1  (With ZFDC & 8" SSSD Floppy Disks)',CR,LF
CLEANUP                    DB     1H,10H,11H,17H,07H,0H
DRNRDY              DB     CR,LF,'DRIVE NOT READY',0
INTTRP              DB     CR,LF,'INTERRUPT TRAP HALT AT:- ',0H
INT0TRP                    DB     CR,LF,'DIVIDE TRAP HALT AT:- ',0H
INT4TRP                    DB     CR,LF,'OVERFLOW TRAP HALT AT:- ',0H
```

```
ZFDCRESETERROR              DB     'ZFDC Board Initilization Error.',CR,LF,0
FLOPPYNODISKERR             DB     CR,LF,BELL,'Floppy Disk Drive Login Error. (No detected disk in drive).',0
FLOPPYDOORERR       DB      CR,LF,BELL,'Floppy Disk Drive Login Error. (Door Open?)',0
FLOPPYLOGINERR              DB     CR,LF,BELL,'Floppy Disk Drive Login Error.',0
FLOPPYWRITEERR              DB     CR,LF,'ZFDC Board Sector Write Error.',0
FLOPPYREADERR       DB      CR,LF,'ZFDC Board Sector Read Error.',0
TIMEOUTMSG          DB      CR,LF,'The ZFDC Board Timed Out.',0
INVALIDDRIVEMSG             DB     CR,LF,'Invalid Drive.',0
DISKWPERROR         DB      CR,LF,'This Disk is Write Protected.',0
ERRORCODEMSG        DB      CR,LF,'ZFDC Board Error Code = ',0


;       ---PROGRAM STORAGE---

SEGTABLE      DB     1                    ;SYSTEM MEMORY TABLE
              DW     TPASEG               ;FIRST SEGMENT STARTS AFTER BIOS
              DW     TPALEN               ;AND GOES UP TO 01FFFH


;       --- DISK DEFINITIONS TABLE ---

DPHDR  EQU    OFFSET $                    ;BASE OF DISK PARAMETER BLOCKS
DPE0          DW     XLT1,0000H           ;TRANSLATE TABLE (IBM 8" SD FLOPPY)
              DW     0000H,0000H          ;SCRATCH AREA
              DW     DIRBUF,DPB0          ;DIR BUFF, PARM BLOCK
              DW     CSV1,ALV1            ;CHECK, ALLOC VECTORS
DPE1          DW     XLT1,0000H           ;TRANSLATE TABLE (IBM 8" FLOPPY)
              DW     0000H,0000H          ;SCRATCH AREA
              DW     DIRBUF,DPB1          ;DIR BUFF, PARM BLOCK
              DW     CSV2,ALV2            ;CHECK, ALLOC VECTORS


;         DISKDEF 1,1,26,6,1024,243,64,64,2
;        DISKDEF 2,1

DPB0   EQU    OFFSET $                    ;DISK PARAMETER BLOCK
               DW     26                  ;SECTORS PER TRACK
               DB     3                   ;BLOCK SHIFT
               DB     7                   ;BLOCK MASK
               DB     0                   ;EXTNT MASK
               DW     242                 ;DISK SIZE - 1 (75tracks x 3.25bkks/trk -1)
               DW     63                  ;DIRECTORY MAX
               DB     192                 ;ALLOC0
               DB     0                   ;ALLOC1
               DW     16                  ;CHECK SIZE
               DW     2                   ;OFFSET
               db     0                   ;physical sector shift

DPB1   EQU    OFFSET $                    ;DISK PARAMETER BLOCK
               DW     26                  ;SECTORS PER TRACK
```

```
            DB      3                   ;BLOCK SHIFT
            DB      7                   ;BLOCK MASK
            DB      0                   ;EXTNT MASK
            DW      242                 ;DISK SIZE - 1 (75tracks x 3.25bkks/trk -1)
            DW      63                  ;DIRECTORY MAX
            DB      192                 ;ALLOC0
            DB      0                   ;ALLOC1
            DW      16                  ;CHECK SIZE
            DW      2                   ;OFFSET
            db      0                   ;physical sector shift
            db      0                   ;physical sector mask

XLT1   EQU  OFFSET $                    ;TRANSLATE TABLE
            DB      1,7,13,19
            DB      25,5,11,17
            DB      23,3,9,15
            DB      21,2,8,14
            DB      20,26,6,12
            DB      18,24,4,10
            DB      16,22

XLT2   EQU  XLT1            ;SAME TRANSLATE TABLE


INITFLAG    DB      ZFDCUNINITILIZED    ;Flag set when ZFDC is initilized

STORAGE             DB      0H
RRDSK       RS      1                   ;CP/M REQUESTED DRIVE #
RRTRK       RS      2                   ;CP/M REQUESTED TRACK ADDRESS
RRSEC       RS      2                   ;CP/M REQUESTED SECTOR

IOBYT       RS      1                   ;STORAGE FOR IOBYTE
DMA_SEGMENT RS      2                   ;STORAGE FOR CURRENT SEGMENT ADDRESS
DMA_OFFSET  RS      2                   ;BUFFER (DMA) ADDRESS


FLGSIZ EQU  (OFFSET $) - (OFFSET STORAGE)   ;DEFINES SIZE OF VARIABLE STORAGE

LOCSTK      RW      64                  ;LOCAL STACK FOR INITILIZATION
STKBASE             EQU     OFFSET $


;       UNINITIALIZED SCRATCH MEMORY FOLLOWS:
;
BEGDAT EQU  OFFSET      $                   ;START OF SCRATCH AREA
DIRBUF      RS      128                 ;DIRECTORY BUFFER
ALV1        RS      31                  ;ALLOC VECTOR
CSV1        RS      16                  ;CHECK VECTOR
ALV2        RS      31                  ;ALLOC VECTOR
CSV2        RS      16                  ;CHECK VECTOR
```

```
ENDDAT EQU    OFFSET $                  ;END OF SCRATCH AREA
DATSIZ EQU    (OFFSET $) - BEGDAT       ;SIZE OF SCRATCH AREA
              DB     0                  ;MARKS END OF MODULE

XLAST  EQU    OFFSET $
;
TPASEG EQU    (XLAST + 0500H+15)/16            ;
TPALEN  EQU   07FFFH - TPASEG                  ;<---- TOP PARAGRAPH OF RAM FOR CP/M
              DB     0                  ;FOR GENCMD
;
;       ----- LOW MEMORY -------

       DSEG   0H
       ORG    0H                        ;AT LOW MEMORY
;
INT0OFFSET    RW     1
INT0SEGMENT   RW     1
                                        ;PAD TO OVERFLOW TRAP VECTOR
              RW     6
INT4OFFSET    RW     1
INT4SEGMENT   RW     1
                                        ;PAD TO SYSTEM CALL VECTOR

       ORG    380H

BDOSOFFSET    RW     1
BDOSSEGMENT   RW     1

END
```