```
TITLE   'CPM3 MODULE FOR ZFDC CONTROLLER.'

; Assume a BANKED version of CPM3
;
;       12/17/1010      V1.0            John Monahan (monahan@vitasoft.org)
;       12/24/2010      V1.1            Added Timeout loop in Sector R/W.

FALSE   EQU     0
TRUE    EQU     1
;
LF      EQU     0AH
CR      EQU     0DH
BELL    EQU     07H
CLEAR   EQU     1AH                     ;SD Systems Video Board Clear Screen
TAB     EQU     09H
ESC     EQU     1BH
EOL     EQU     1CH                     ;SD Systems Video Board Clear to end of line



;       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

ZFDC$UNINITILIZED EQU   0FFH            ;If ZFDC is not yet initilized
ZFDC$NOT$WORKING EQU    0FEH            ;If ZFDC is not working
ZFDC$NOT$PRESENT EQU    0FDH            ;If ZFDC board is absent
ZFDC$INITILIZED EQU     000H            ;If ZFDC is initilized OK


;       PORTS  FOR FOR SD Systems Video Board
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


;Commands to the ZFDC Board:-

CMD$PIO$TEST    EQU     0H              ;Simple loop hardware test of PIO #1 Ports
```

```
CMD$MONITOR     EQU     1H              ;Jump to internal monitor here.
CMD$SHOW$SIGNON EQU     2H              ;This will "rotate" in the Sector Display TIL's as a hardware test
CMD$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$GET$FORMAT  EQU     6H              ;Return to S100 System the current Disk paramater format table number
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$STEP$OUT    EQU     0CH             ;Step head out one track of CURRENT drive
CMD$SEEK$NV     EQU     0DH             ;Seek to track with NO verify of CURRENT drive

CMD$SEEK$TRACK  EQU     0EH             ;Seek to track to (IY+DRIVE$TRACK) with the track verify bit set on CURRENT drive/ ↙
        format
CMD$GET$TRACK$ID  EQU   0FH             ;Read the CURRENT TRACK ID

CMD$READ$SECTOR   EQU   10H             ;Read data from the CURRENT sector (on current track,side,drive).
CMD$WRITE$SECTOR  EQU   11H             ;Write data to the CURRENT sector (on current track,side,drive).

CMD$GET$WD$TRACK  EQU   12H             ;Get the WD2793 Track register value
CMD$GET$WD$SECTOR EQU   13H             ;Get the WD2793 Sector register value
CMD$GET$WD$STATUS EQU   14H             ;Get the WD2793 Status register value

CMD$TRACK$DUMP  EQU     15H             ;Dump complete CURRENT track to S-100 system
CMD$FORMAT$TRACK EQU    16H             ;Format the disk in the of the CURRENT drive using the current format assigned to ↙
        that disk

CMD$SET$DEBUG$ON EQU    17H             ;Turn on Debug display mode
CMD$SET$DEBUG$OFF EQU   18H             ;Turn off Debug display mode
CMD$RAM$DUMP    EQU     19H             ;Command to pass back to S-100 system all memory variables and flags on ZFDC board

CMD$ABORT       EQU     20H             ;Generalized Abort of the current process command.
CMD$HANDSHAKE   EQU     21H             ;Handshake command only sent during board initilization/testing
CMD$GET$DRIVE   EQU     22H             ;Get the current selected drive number (0..3)
CMD$SET$TRACK$DS EQU    23H             ;Set Track (If a DS Disk, EVEN tracks on Side A, ODD tracks on Side B. Used by  ↙
        CPM)
CMD$GET$ERROR$STRING EQU 24H            ;Return a string explaining the last Error Code sent


;ERROR codes returned from the ZFDC Board:-

NO$ERRORS$FLAG  EQU     00H             ;No Errors flag for previous cmd, sent back to S-100 BIOS
BUSY$ERR        EQU     01H             ;WD2793 Timeout Error before CMD was started
```

```
HUNG$ERR        EQU     02H             ;General WD2793 Timeout Error after CMD was sent
TABLE$ERR       EQU     03H             ;Disk parameter table error
DRIVE$ERR       EQU     04H             ;Drive not 0-3
TRACK$RANGE$ERR EQU     05H             ;Drive track not valid for this disk
SECTOR$RANGE$ERR EQU    06H             ;Drive sector not valid for this disk
SIDE$ERR        EQU     07H             ;No B side on this disk
SIDE$ERR1       EQU     08H             ;Invalid Side Paramater
SECTOR$SIZE$ERR EQU     09H             ;Size of sector > 1024 Bytes

RESTORE$HUNG    EQU     0AH             ;WD2793 Timeout Error after RESTORE Command
RESTORE$ERR     EQU     0BH             ;Restore to track 0 error

STEP$IN$HUNG    EQU     0CH             ;WD2793 Timeout Error after STEP-IN Command
STEP$IN$ERR     EQU     0DH             ;Head Step In Error, DRIVE NOT READY ERROR
STEP$OUT$HUNG   EQU     0EH             ;WD2793 Timeout Error after STEP-OUT Command
STEP$OUT$ERR    EQU     0FH             ;Head Step Out Error, NOT READY ERROR

SEEK$NV$HUNG    EQU     10H             ;WD2793 Timeout Error after SEEK-NV Command
SEEK$NV$ERR1    EQU     11H             ;Seek with No Verify Error, NOT READY ERROR
SEEK$NV$ERR2    EQU     12H             ;Seek with No Verify with SEEK error bit set

SEEK$TRK$HUNG   EQU     13H             ;WD2793 Timeout Error after SEEK with Verify Command
SEEK$TRK$ERR1   EQU     14H             ;Seek to a track with Verify error, DRIVE NOT READY ERROR bit set
SEEK$TRK$ERR2   EQU     15H             ;Seek to a track with Verify error with SEEK ERROR bit set
SEEK$REST$HUNG  EQU     16H             ;WD2793 Timeout Error after RESTORE within SEEK with Verify Command
SEEK$REST$ERR   EQU     17H             ;Restore to track 0, DRIVE NOT READY ERROR within SEEK with Verify Command

ID$ERR$HUNG     EQU     18H             ;WD2793 Timeout Error after READ TRACK ID Command
ID$ERR1         EQU     19H             ;Track ID Error, DRIVE NOT READY ERROR
ID$ERR2         EQU     1AH             ;Track ID Error, RNF ERROR
ID$ERR3         EQU     1BH             ;Track ID Error, LOST DATA ERROR
ID$ERR4         EQU     1CH             ;Track ID Error, CRC ERROR

SEC$READ$HUNG   EQU     1DH             ;WD2793 Timeout Error after Read Sector Command was sent
SEC$READ$ERR1   EQU     1EH             ;Sector read error, DRIVE NOT READY ERROR
SEC$READ$ERR2   EQU     1FH             ;Sector read error, RNF ERROR
SEC$READ$ERR3   EQU     20H             ;Sector read error, LOST DATA ERROR
SEC$READ$ERR4   EQU     21H             ;Sector read error, CRC ERROR
RS$SEEK$TRK$HUNG EQU    22H             ;WD2793 Timeout Error after SEEK within READ SECTOR Command
RS$RESTORE$HUNG EQU     23H             ;WD2793 Timeout Error after RESTORE command within READ SECTOR Command
RS$RESTORE$ERR  EQU     24H             ;Restore to track 0 Error, within READ SECTOR Command
RS$SEEK$TRK$ERR1 EQU    25H             ;Seek to track error, within READ SECTOR Command
RS$SEEK$TRK$ERR2 EQU    26H             ;Seek to track error with SEEK ERROR bit set within READ SECTOR Command

SEC$WRITE$HUNG  EQU     27H             ;WD2793 Timeout Error after Read Sector Command was sent
SEC$WRITE$ERR1  EQU     28H             ;Sector write error, DRIVE NOT READY ERROR
SEC$WRITE$ERR2  EQU     29H             ;Sector write error, RNF ERROR
SEC$WRITE$ERR3  EQU     2AH             ;Sector write error, LOST DATA ERROR
```

```
SEC$WRITE$ERR4  EQU     2BH             ;Sector write error, CRC ERROR
WS$SEEK$TRK$HUNG EQU    2CH             ;WD2793 Timeout Error after SEEK within WRITE SECTOR Command
WS$RESTORE$HUNG EQU     2DH             ;WD2793 Timeout Error after RESTORE command within WRITE SECTOR Command
WS$RESTORE$ERR  EQU     2EH             ;Restore to track 0 Error, within WRITE SECTOR Command
WS$SEEK$TRK$ERR1 EQU    2FH             ;Seek to track error, within WRITE SECTOR Command
WS$SEEK$TRK$ERR2 EQU    30H             ;Seek to track error with SEEK ERROR bit set within WRITE SECTOR Command
DISK$WP$ERR     EQU     31H             ;Sector write error, Disk is write protected

CONFIRM$FORMAT  EQU     32H             ;Confirm disk format cmd request
FORMAT$HUNG     EQU     33H             ;WD2793 Timeout Error after Track Format Command was sent
FORMAT1$ERR     EQU     34H             ;Disk format request error
FORMAT2$ERR     EQU     35H             ;Track format error (Side A)
FORMAT3$ERR     EQU     36H             ;Track format error (Side B)
FORMAT4$ERR     EQU     37H             ;Restore error after formatting disk
RT$ERR$HUNG     EQU     38H             ;Disk Read Track hung error
RT$ERR          EQU     39H             ;Disk Read track error
ABORT$FLAG      EQU     3AH             ;Special error flag to signify the user aborted a command

ZFDC$ABSENT     EQU     3BH             ;If ZFDC Board is absent
ZFDC$INIT$ERROR EQU     3CH             ;If ZFDC initilization error

TIMEOUT$ERROR   EQU     3DH             ;Error flag to signify the previous command timed out

CMD$RANGE$ERR   EQU     3EH             ;CMD out or range


;------------  DISK PARAMATER TABLE NUMBERS ------------------

STD8IBM         EQU     1               ;IBM 8" SDSS Diak
STDDDT          EQU     2
DDT256          EQU     3               ;IBM System-36 DDDS Disk
DDT512          EQU     4
DDT1K           EQU     5
DDT1K2          EQU     6
MINSDT          EQU     7
MINDDT          EQU     8
MINCPM          EQU     9               ;CPM-86 5" DDDS Disk
DEC             EQU     0AH
TOSHIBA         EQU     0BH
CDOS            EQU     0CH
CDOSDD          EQU     0DH
EPSON           EQU     0EH
MORROW          EQU     0FH
ZENITH          EQU     10H
SUPER           EQU     11H
MSDOS           EQU     12H
MSDOS2          EQU     13H
TRS80           EQU     14H
```

```
        ; DEFINE PUBLIC LABELS:
        PUBLIC  DPH1,DPH2,DPH3          ;FLOPPY DISK PARAMETER HEADERS

        ; DEFINE EXTERNAL LABELS:
        EXTRN   @ADRV,@RDRV
        EXTRN   @DMA,@TRK,@SECT
        EXTRN   @CBNK
        EXTRN   @DBNK                   ;BANK FOR DMA OPERATION
        EXTRN   @ERMDE                  ;BDOS ERROR MODE
        EXTRN   ?WBOOT                  ;WARM BOOT VECTOR
        EXTRN   ?PMSG                   ;PRINT MESSAGE @<HL> UP TO 00, SAVES [BC] AND [DE]
        EXTRN   ?PDERR                  ;PRINT BIOS DISK ERROR HEADER
        EXTRN   ?CONIN,?CONO            ;CONSOLE IN AND OUT
        EXTRN   ?CONST                  ;CONSOLE STATUS
        EXTRN   ?BNKSL                  ;SELECT PROCESSOR MEMORY BANK
        EXTRN   ?SMSG                   ;MY ROUTINE TO SPEAK A MESSAGE

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

        ; INCLUDE Z-80 MACRO LIBRARY:
        MACLIB  Z80

        DSEG                            ;PUT IN OP SYS BANK IF BANKING

        ; EXTENDED DISK PARAMETER HEADER FOR DRIVE 0: (B:)
        DW      WRITE$SECTOR            ;FD SEC WRITE ROUTINE
        DW      READ$SECTOR             ;FD SEC READ ROUTINE
        DW      FLOPPY$LOGIN$0          ;FLOPPY DISK "B:" LOGIN PROCEDURE
        DW      FLOPPY$INIT$0           ;FLOPPY DISK "B:" DRIVE INITIALIZATION ROUTINE
        DB      0                       ;RELATIVE DRIVE 0 ON THIS CONTROLLER
        DB      1                       ;MEDIA TYPE KNOWN SSSD 8"
                                        ;HI BIT SET : DRIVE NEEDS RECALIBRATING

DPH1:   DPH     SD128$trans,SDSS128$dpb,,

        ; EXTENDED DISK PARAMETER HEADER FOR DRIVE 1: (C:)
        DW      WRITE$SECTOR            ;FD SEC WRITE ROUTINE
        DW      READ$SECTOR             ;FD SEC READ ROUTINE
        DW      FLOPPY$LOGIN$1          ;FLOPPY DISK "C:" LOGIN PROCEDURE
        DW      FLOPPY$INIT$1           ;FLOPPY DISK DRIVE "C:" INITIALIZATION ROUTINE
        DB      0                       ;RELATIVE DRIVE 1 ON THIS CONTROLLER
        DB      1                       ;MEDIA TYPE KNOWN SSSD 8"
                                        ;HI BIT SET : DRIVE NEEDS RECALIBRATING

DPH2:   DPH     SD128$trans,SDSS128$dpb,,
```

```
        ; EXTENDED DISK PARAMETER HEADER FOR DRIVE 2: (D:)
        DW      WRITE$SECTOR            ;FD SEC WRITE ROUTINE
        DW      READ$SECTOR             ;FD SEC READ ROUTINE
        DW      FLOPPY$LOGIN$2          ;FLOPPY DISK "D:" LOGIN PROCEDURE
        DW      FLOPPY$INIT$2           ;FLOPPY DISK "D:" DRIVE INITIALIZATION ROUTINE
        DB      0                       ;RELATIVE DRIVE 2 ON THIS CONTROLLER
        DB      0                       ;MEDIA TYPE KNOWN 5" (CPM-86 Format)
                                        ;HI BIT SET : DRIVE NEEDS RECALIBRATING

DPH3:   DPH     DD512$trans,MINI$dpb,,

        ; MAKE SURE DPB'S ARE IN COMMON MEMORY:

        CSEG

        ; 128 byte sectors, SD, SS disk parameter block (IBM 3740):
SDSS128$dpb:    dpb     128,26,77,1024,64,2     ;8" Disks
MINI$dpb:       dpb     512,16,40,2048,64,1     ;5" Disks  (16=8X2, because 1 Track, but both sides)


        DSEG                            ;CAN SET BACK TO BANKED SEGMENT IF BANKING

SD128$trans:    skew    26,6,0          ;8" sector skew
DD512$trans:    skew    16,4,0          ;5" sector skew (16 = 8X2)


        ;;;;; INIT:                     ;Initilization routines for each floppy drive
FLOPPY$INIT$0:                          ;For the first floppy also inutilize the ZFDC board
        CALL    RESET$ZFDC              ;First drive so initilize the ZFDC board as well.
        JRNZ    BAD$ZFDC                ;If problem say so
        CALL    SET$DISK$FORMATS        ;Set Disk Formats for all 3 drives here so current drive is 0.
        RET

FLOPPY$INIT$1:                          ;Next the second floppy drive, Just return
        RET

FLOPPY$INIT$2:                          ;Then the third floppy drive, Just return
        RET


BAD$ZFDC:
        LXI     H,ZFDC$RESET$ERROR
        CALL    ?PMSG
        RET
```

```
RESET$ZFDC:                             ;Routine the Reset or Initilize the ZFDC Board hardware
        OUT     RESET$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$HANDSHAKE           ;Make sure we get HANDSHAKE byte back
        MVI     A,ZFDC$ABSENT           ;If not then no ZFDC board present
        STA     INITILIZED$FLAG         ;Flag the board as being absent
        RNZ                             ;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
        MVI     A,ZFDC$INIT$ERROR       ;If not then no ZFDC board present
        STA     INITILIZED$FLAG         ;Flag the board as having an error
        RNZ                             ;and abort

        MVI     A,ZFDC$INITILIZED       ;Flag the ZFDC is initilized
        STA     INITILIZED$FLAG
        XRA     A
        RET

SET$DISK$FORMATS:                       ;Do reverse order so we end up with Drive 0 selected
        MVI     C,CMD$SET$FORMAT        ;Send Set Disk Format to Drive CMD
        CALL    S100OUT
        MVI     C,2                     ;Floppy Drive 2, (ZFDC Board expects a 0H, 1H, 2H or 3H)
        CALL    S100OUT
        MVI     C,MINCPM                ;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]

        MVI     C,CMD$SET$FORMAT        ;Send Set Disk Format to Drive CMD
        CALL    S100OUT
        MVI     C,1                     ;Floppy Drive 1, (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]

        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
        RET


;-----------------LOGIN OF FLOPPY DISK ---------------------------------------
;
;       on entry:       [de] = pointer to XDPH of drive to be logged-in
;
;       on exit:        XDPH address on the stack at entry has been
;                       replaced with a null address if the drive has
;                       been determined to be an invalid one

FLOPPY$LOGIN$0:                         ;Restore Floppy Drive 0, to Track 0,
        LXI     H,128                   ;Size of a sector on this disk format
        SHLD    SECTOR$BYTE$COUNT       ;Bytes/sector for this disk format
        MVI     D,0                     ;Select Drive 0
        JR      LOGIN$DISK

FLOPPY$LOGIN$1:                         ;Restore Floppy Drive 1, to Track 0,
        LXI     H,128                   ;Size of a sector on this disk format
        SHLD    SECTOR$BYTE$COUNT       ;Bytes/sector for this disk format
        MVI     D,1                     ;Select Drive 1
        JR      LOGIN$DISK

FLOPPY$LOGIN$2:                         ;Restore Floppy Drive 2, to Track 0,
        LXI     H,512                   ;Size of a sector on this disk format
        SHLD    SECTOR$BYTE$COUNT       ;Bytes/sector for this disk format
        MVI     D,2                     ;Select Drive 2
        JR      LOGIN$DISK

LOGIN$DISK:
        MVI     C,CMD$SET$DRIVE         ;Send a "Set Drive CMD" to ZFDC board
        CALL    S100OUT
        MOV     C,D                     ;Floppy Drive #, (ZFDC Board expects a 0H, 1H, 2H or 3H)
        CALL    S100OUT
        CALL    WAIT$FOR$ACK            ;Return Z (and NO$ERRORS$FLAG in [A]), or NZ with error # in [A]
        JNZ     LOGIN$ERROR
        XRA     A                       ;Return Z if no error
        RET
```

```
LOGIN$ERROR:                            ;If a problem logging into a drive return an error to the system
        LXI     H,FLOPPY$LOGIN$ERR
        CALL    GET$ERROR$STRING        ;Get Error String from ZFDC Board
        POP     D                       ;[DE] has calling routine's return address
        LXI     H,0                     ;This will be invalid drive from now on in tthe CPM BDOS
        XTHL                            ;put on the stack in place of the XDPH ptr.
        XCHG
        PCHL                            ;Back to caller...



;---------------FLOPPY SECTOR WRITE ROUTINE ---------------------------
;               ROUTINE WRITES 1 SECTOR TO THE DISK:
;
;       on entry:       [de] = XDPH address for current drive
;
;       on exit:        [a] =   0 --> successful read operation
;                               1 --> unsuccessful read operation
;                             255 --> media change
;
; Assumes valid track in (@TRK) & (@SECT), Address in (@DMA).
; If the disk is double sided then the ZFDC board assumes 1 to NSCTRS+1 on Side A
; and NSCTRS+1 to (NSCTRS X 2)+1 on Side B
; Note: The XDPH table is unused since we only need the sector byte count.
; The sector byte count was placed in SECTOR$BYTE$COUNT in the above FLOPPY$LOGIN routines.
;
WRITE$SECTOR:
        MVI     C,CMD$SET$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]
        JRNZ    WRITE$ERROR

        MVI     C,CMD$SET$SECTOR        ;Sector # to side A (or for DS disks also side B)
        CALL    S100OUT
        LDA     @SECT
        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]
        JRNZ    WRITE$ERROR

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

```
        JRNZ    WRITE$ERROR

        MVI     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
        JRNZ    WRITE$ERROR

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

        JMP     ADJ$BANK1
        CSEG
;=====================================
ADJ$BANK1:
        LDA     @CBNK
        PUSH    PSW
        LDA     @DBNK
        CALL    ?BNKSL

WR$SEC: MVI     B,0FFH                  ;Put in a timeout count (Loop for status read at most 256 times)
WR$SEC1:DJNZ    WR$SEC2
        MVI     A,TIMEOUT$ERROR         ;Send Timeout error
        POP     PSW
        CALL    ?BNKSL
        JMP     WRITE$ERROR             ;Note JMP to DSEG bank!
                                        ;Note we cannot use S100OUT here since we are no longer in the DSEG bank
WR$SEC2:IN      S100$STATUS$B           ;Send data to ZFDC output (arrive with byte to be sent in M)
        BIT     DIRECTION$BIT,A         ;Is ZFDC in output mode, if not wait
        JRNZ    WR$SEC1
        BIT     DATA$OUT$RDY,A          ;Has previous (if any) character been read.
        JRZ     WR$SEC1                 ;Z if not yet ready

        MOV     A,M                     ;Get the byte
        OUT     S100$DATA$B             ;Send it
        INX     H                       ;[HL++] for [DE--] bytes in sector
        DCX     D
        MOV     A,E
        ORA     D
        JRNZ    WR$SEC

        POP     PSW
        CALL    ?BNKSL
        JMP     CHECK$WR
;=====================================
        DSEG

CHECK$WR:
```

```
        CALL    WAIT$FOR$ACK            ;Return Z (and NO$ERRORS$FLAG in [A]), or NZ with error # in [A]
        JRNZ    WRITE$ERROR
        RET

WRITE$ERROR:
        LXI     H,FLOPPY$WRITE$ERR      ;"Sector Read Error"
        CALL    GET$ERROR$STRING        ;Get Error String from ZFDC Board
        JP      COMMON$RW$ERROR

;
;
;---------------FLOPPY SECTOR READ ROUTINE ----------------------------
;               ROUTINE READS 1 SECTOR TO THE DISK:
;
;       on entry:       [de] = XDPH address for current drive
;
;       on exit:        [a] =   0 --> successful read operation
;                               1 --> unsuccessful read operation
;                             255 --> media change
;
; Assumes valid track in (@TRK) & (@SECT), Address in (@DMA).
; If the disk is double sided then the ZFDC board assumes 1 to NSCTRS+1 on Side A
; and NSCTRS+1 to (NSCTRS X 2)+1 on Side B
; Note: The XDPH table is unused since we only need the sector byte count.
; The sector byte count was placed in SECTOR$BYTE$COUNT in the above FLOPPY$LOGIN routines.
;
READ$SECTOR:
        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]
        JRNZ    READ$ERROR

        MVI     C,CMD$SET$SECTOR        ;Sector # to side A (or for DS disks also side B)
        CALL    S100OUT
        LDA     @SECT
        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]
        JRNZ    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]
```

```
        JRNZ    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
        JRNZ    READ$ERROR

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

        JMP     ADJ$BANK2
        CSEG
;=====================================
ADJ$BANK2:
        LDA     @CBNK                   ;get current bank
        PUSH    PSW
        LDA     @DBNK                   ;Get Destination Bank. MUST HAVE THIS CODE IN COMMON
        CALL    ?BNKSL                  ;NOW DMA ADDRESS IS AT THE CORRECT BANK


RD$SEC: MVI     B,0FFH                  ;Put in a timeout count (Loop for status read at most 256 times)
RD$SEC1:DJNZ    RD$SEC2
        MVI     A,TIMEOUT$ERROR         ;Send Timeout error
        POP     PSW
        CALL    ?BNKSL
        JMP     READ$ERROR              ;Note JMP to DSEG bank!
                                        ;Note: we cannot use S100IN here since we are no longer in the DSEG bank
RD$SEC2:IN      S100$STATUS$B           ;Check if ZFDC has any data for S-100 system
        BIT     DIRECTION$BIT,A         ;Is ZFDC in input mode, if not wait.
        JRZ     RD$SEC1                 ;If low then ZFDC board is still in input mode, wait
        BIT     DATA$IN$RDY,A           ;Is there a character available
        JRZ     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
        JRNZ    RD$SEC                  ;Next Byte, reset timeout count

        POP     PSW
        CALL    ?BNKSL
        JMP     CHECK$RD
;=====================================
        DSEG
```

```
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$ERROR:
        LXI     H,FLOPPY$READ$ERR       ;"Sector Read Error"
        CALL    GET$ERROR$STRING        ;Get Error String from ZFDC Board

COMMON$RW$ERROR
        LXI     H,TRACK$MSG
        CALL    ?PMSG
        LDA     @TRK
        CALL    PACC
        LXI     H,SECTOR$MSG
        CALL    ?PMSG
        LDA     @SECT
        INR     A                       ;Because sectors are numbered 1,2,3...
        CALL    PACC
        LXI     H,H$MSG                 ;Finish with a "H."
        CALL    ?PMSG
        POP     D                       ;Get back [DE]
        XRA     A
        INR     A                       ;Set to NZ & 1 for read 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]



GET$ERROR$STRING:                       ;General Print error string routine.
        PUSH    PSW
        CALL    ?PMSG                   ;Print the Header in [HL]

        POP     PSW
        CPI     0FFH                    ;Was it a timeout error
        JRNZ    GET$STRING
```

```
        LXI     H,TIMEOUT$MSG
        CALL    ?PMSG
        RET

GET$STRING:
        PUSH    PSW
        MVI     C,CMD$GET$ERROR$STRING  ;See if we can get the Text String about the error
        CALL    S100OUT
        POP     PSW
        MVI     C,A
        CALL    S100OUT                 ;Send Error Number

RD$STRING:                              ;Get Error String returned back
        CALL    S100IN                  ;Note potential to lockup here
        ORA     A
        RZ                              ;return when done
        MOV     C,A
        CALL    ?CONO
        JR      RD$STRING



;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    CO
        RET
;
CI:     IN      0H              ;Direct console input (used only for debugging)
```

```
        ANI     02H
        JZ      CI
        IN      1H              ;return with character in A
        RET
;
CO:     IN      0H              ;console output (arrive with character in C)
        ANI     04H             ;Note character is in C and A on return.
        JZ      CO
        MOV     A,C
        OUT     1H
        RET


        DSEG
;-------------------------------------------------------------------------------
;
INITILIZED$FLAG         DB      ZFDC$UNINITILIZED       ;Flag set when ZFDC is initilized
SECTOR$BYTE$COUNT       DW      128                     ;Bytes/sector for current selected disk

LOGIN$ERROR$0           DB      CR,LF,'Floppy Disk Drive 0 Initilization Error.',0
LOGIN$ERROR$1           DB      CR,LF,'Floppy Disk Drive 1 Initilization Error.',0
LOGIN$ERROR$2           DB      CR,LF,'Floppy Disk Drive 2 Initilization Error.',0
ZFDC$RESET$ERROR        DB      'ZFDC Board Initilization Error.',0
FLOPPY$LOGIN$ERR        DB      CR,LF,'Floppy Disk Drive Login Error. ZFDC Error code = ',0
FLOPPY$WRITE$ERR        DB      CR,LF,'Sector Write Error.',0
FLOPPY$READ$ERR         DB      CR,LF,'Sector Read Error.',0
TIMEOUT$MSG             DB      CR,LF,'The ZFDC Board Timed Out.',0
HMSG                    DB      'H.',0
TRACK$MSG               DB      CR,LF,'Track = ',0
SECTOR$MSG              DB      'H.   Sector = ',0
CRLF$MSG                DB      CR,LF,0
```