```
;THIS IS A PROGRAM TO EXAMINE & FORMAT CPM DISKS USEING THE S100COMPUTERS Z80/WD2793 (ZFDC) Controller board

;       JOHN MONAHAN     (monahan@vitasoft.org)    Started 9/3/2010
;       VERSION  0.1     Initial code
;
;       V0.1                    Streamlined error handeling
;       V0.2                    Random Seek track for Track ID done
;       V0.3                    Random sector reads working
;       V0.4                    Track formatting added
;       V0.5                    Combined set tarck and set format into one command
;       V2.0                    Setup for V2 ZFDC board. Communications check at start
;       V2.1                    Added Side,Density etc. equates.
;       V2.2                    Cleaned up menu. Can now select different drives.
;       V2.3                    Added multi sector R/W capability
;       V2.5                    Added multi-sector R/W test
;       V2.6                    Added multi-sector R/W test with fast track buffering
;       V2.7    03/08/2011      Corrected Drive Select Menu option, set default disk formats for drives 0,1,2.
;       V2.8    04/30/2011      Added support for IBM 1.2M/5" and 1.44M/3.5" disk formats
;       V2.9    05/1/2011       Added after format modifications of MSDOS disks

;Please see the notes in ZFDC.Z80 to understand how this Diagnostic routine works

FALSE           EQU     0
TRUE            EQU     NOT FALSE

                                        ;Equates for display on SD Systems Video Board (Used In CPM Debugging mode only)
SCROLL          EQU     01H             ;Set scrool direction UP.
LF              EQU     0AH
CR              EQU     0DH
BS              EQU     08H             ;Back space (required for sector display)
BELL            EQU     07H
SPACE           EQU     20H
TAB             EQU     09H             ;TAB ACROSS (8 SPACES FOR SD-BOARD)
ESC             EQU     1BH
QUIT            EQU     11H             ;Turns off any screen enhancements (flashing, underline etc).
NO_ENHANCEMENT  EQU     17H             ;Turns off whatever is on
FAST            EQU     10H             ;High speed scrool


; BDOS EQUATES (VERSION 3)
CPM_RESET_DISKS EQU     13              ;Reset all CPM disks
BDOS            EQU     5               ;All CPM BDOS calls to there

STATUS_DELAY    EQU     50              ;Time-out for waiting for ZFDC Board handshake signal (~5 seconds @ 4MHz)
CPM86_FLAG      EQU     01              ;Flag to indicate after 5" disk formating CPM86 first sector needs to be modified
MAX_TRACK_SIZE  EQU     3000H           ;Maximum number of bytes there will be on a track. Used for formatting and reading a
                                        ;a disk track

                                        ;Will place these values here for easy RAM analysis if a crash
                                        ;The rest go to the end of the program.

@DRIVE          EQU     42H             ;CURRENT DISK DRIVE NUMBER (0,1,2,3), (old UNIT)
@SEC            EQU     43H             ;NEW SECTOR
@TRK            EQU     44H             ;NEW TRACK
@REC_COUNT      EQU     45H             ;Count for multi sector R/W
@FORMAT         EQU     47H             ;CURRENT DISK FORMAT NUMBER (0,1,2,3...13H)
```

```
@SIDE            EQU     49H             ;NEW SIDE


; EQUATES FOR [IX] REGISTER OFFSETS INTO DISK FORMAT PARAMATER TABLES
; Each of the many disk formats have their own table with things like sector size, tracks/disk, sectors/track etc.
; You can add more.
;
; Note: [IX] will ALWAYS point to the current disk parameter table in this program <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

; Note Also: It is very important that the BIOS "Disk Paramater Tables in any S-100 BIOS are EXACTLY the
; same as that on the ZFDC Board. There can be less disk formats (actully only one is required), but a drive format
; number (FORMAT_NUM) on the S-100 BIOS has to be the same drive format on the ZFDC board. This is because this
; table is used to determine sector size, sectors/track etc.

HW_BYTE                  EQU     0               ;Will contain bit flags for:-
                                                 ;Bits 0,1 Unused
                                                 ;Bit 2, 0 if single sided disk, 1 if double sided
                                                 ;Bit 3, 1 Hardware is an 8" disk, 0 = 5" disk
                                                 ;Bit 4, 1 R/W in Single density, 0 = Double density
                                                 ;Bits 5-7 Unused

NSCTRS                   EQU     1               ;Sectors/Track for this disk format
NTRKS                    EQU     2               ;Tracks/Side
HEADR                    EQU     3               ;For Formatting
GAP1                     EQU     4               ;       "
GAP2                     EQU     5               ;       "
GAP3                     EQU     6               ;       "
GAP4                     EQU     7               ;       "
GAP4R                    EQU     8               ;       "
SEC_SIZE_FLAG            EQU     9               ;0=128 Byte sectors, 1 = 256, 2 = 512, 4=1024 Byte sectors
GAP_FILL_CHAR            EQU     10              ;Byte used in disk formating
DATA_FILL_CHAR           EQU     11              ;     "       "       "
SPECIAL_FLAG             EQU     12              ;Flag byte for cases where after formatting disk need to be initilized. Normally 0, CPM86_FLAG = 1
SEC_SKEW_TABLE           EQU     13              ;Two Bytes.    Address of sector skew table
FORMAT_NUM               EQU     15              ;Each format will have a unique number in the table list below.
SYS_TRKS         EQU     16              ;How many tracks for system (usually 2 for 8-inch CPM disks)
SEC_SIZE_BYTES           EQU     17              ;Two Bytes.    (128,256,512 or 1024)
TRACK_SIZE               EQU     19              ;Two Bytes.    Track size (in bytes) of that disks format (used in formatting disk)
TITLE                    EQU     21              ;Text string describing the disk format (must end with 0)

DIRECTION_BIT            EQU     7               ;Bits for the above flags 0 = IN, 1 = OUT
WD_INTRQ_BIT             EQU     6
WATCHDOG_BIT             EQU     5
DENSITY_BIT              EQU     4
SIZE_BIT         EQU     3
SIDE_BIT         EQU     2


;       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.

;       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              ;Set the current Disk number (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 a track 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
CMD_GET_SEC_SIZE EQU    25H             ;Return currently selected disk sector size (X*128)
CMD_GET_SEC_COUNT EQU   26H             ;Return currently selected disk sector sectors/track
CMD_ZFDC_ALIVE  EQU     27H             ;Command sent to see if the ZFDC board is present and responding.
CMD_CHECK_DRIVE  EQU    28H             ;Check there is a valid drive present on specified drive (0,1,2,3)


CMD_RD_MULTI_SECTOR EQU 29H             ;Read data from multiple sectors starting at the CURRENT sector (on current track,side,drive).
CMD_WR_MULTI_SECTOR EQU 2AH             ;Write data to multiple sectors starting at the CURRENT sector (on current track,side,drive).

;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
DRIVE_INACTIVE   EQU     3AH             ;Drive is inactive
DRIVE_DOOR       EQU     3BH             ;Drive door open
ABORT_FLAG       EQU     3CH             ;Special error flag to signify the user aborted a command
CMD_RANGE_ERR    EQU     3DH             ;CMD out or range

TIMEOUT_ERROR    EQU     0FFH            ;Special error flag to signify the previous command timed out




;-------------------------------------------------------------------------------
         ORG     100H

         LD      (SP_SAVE),SP
         LD      SP,STACK
         JP      START
;-------------------------------------------------------------------------
;                HARDWARE DEPENDENT STUFF
; The only hardware port links for the above ZFDC board.

CONSTAT:IN       A,(CSTAT)               ;console status for SD Systems 8024 Video board
         AND     A,02H                   ;anything there
         RET     Z                       ;return 0 if nothing
         XOR     A,A
         DEC     A                       ;return NZ, & 0FFH in A if something there
         RET

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

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

```
S100STAT:IN      A,(S100_STATUS_B);Check if ZFDC has any data for S-100 system
         BIT     0,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);Send data to ZFDC output (arrive with character to be sent in C)
         BIT     DIRECTION_BIT,A         ;Is ZFDC in input mode, if not wait
         JR      Z,S100IN         ;if low then ZFDC is in input mode, wait
         AND     A,01H
         JR      Z,S100IN
         IN      A,(S100_DATA_A)         ;return with character 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     1,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




;-------------------------------------------------------------------------
START:   LD      HL,SIGNON
         CALL    PMSG
         XOR     A,A                     ;set everything to zero
         LD      (@TRK),A
         LD      (@SIDE),A
         LD      (@DRIVE),A
         LD      (DISPLAY_FLAG),A ;Detailed sector data display ON
         LD      (DEBUG_FLAG),A          ;Debug flag off initially on ZFDC board
         INC     A
         LD      (@SEC),A         ;Track 0, side A, sector 1
         LD      IX,STD8IBM              ;Set as default drive table: IBM 8" SSSD Disk

         OUT     RESET_ZFDC_PORT,A;Do a hardware reset. Does not matter what is in [A]
         CALL    DELAY_1S         ;Allow Board time to re-initilize

         IN      A,(S100_DATA_B)         ;Check the board is there
         CP      A,CMD_HANDSHAKE
         JR      NZ,BOARD_ERROR

         LD      A,CMD_HANDSHAKE
         OUT     (S100_DATA_B),A         ;This clears up ints on ZFDC board
         CALL    WAIT_FOR_ACK            ;Check any error flags
         JR      Z,GOT_DRIVE

BOARD_ERROR:
         LD      HL,NO_INITILIZE_MSG
         CALL    PMSG
```

```
        JP      0H

GOT_DRIVE:
        LD      A,2                     ;Third ZFDC Drive
        LD      B,9                     ;(MINCPM) Set initially for an 5" IBM SSSD
        CALL    BIOS_SET_FORMAT         ;First select the initial drive and DISKS format
        CALL    NZ,SHOW_ERRORS

        LD      A,1                     ;Second ZFDC Drive
        LD      B,1                     ;(STD8IBM) Set initially for an 8" IBM SSSD
        CALL    BIOS_SET_FORMAT         ;First select the initial drive and DISKS format
        CALL    NZ,SHOW_ERRORS

        LD      A,0                     ;First ZFDC Drive
        LD      B,1                     ;(STD8IBM) Set initially for an 8" IBM SSSD
        CALL    BIOS_SET_FORMAT         ;First select the initial drive and DISKS format
        CALL    NZ,SHOW_ERRORS

LOOP:   LD      HL,MAIN_MENU            ;Main Menu
        CALL    PMSG
        LD      HL,DISK_TBL_MSG
        CALL    PMSG
        CALL    SHOW_DRIVE_NAME         ;show current info
        CALL    CRLF
        LD      HL,MENU_CHOICE_MSG      ;"Menu = "
        CALL    PMSG
                                        ;-----MAIN MENU----
        CALL    GETHEX                  ;Find out Menu Option in [A] Carry set if Abort/ESC
        JP      C,FINISH        ;Abort if ESC character

        CP      A,MAX_MENU_TBL
        JP      C,TO_MENU_TABLE         ;Is it within range
        LD      HL,CMD_ERROR_MSG
        CALL    PMSG
        JP      LOOP

TO_MENU_TABLE:
        ADD     A,A                     ;X2 for table
        LD      HL,MENU_TBL
        ADD     A,L
        LD      L,A
        LD      A,(HL)
        INC     HL
        LD      H,(HL)
        LD      L,A
        JP      (HL)                    ;JUMP TO COMMAND TABLE (A-Z for now)
                                        ;Do 1,2,3.... later

;       NOTE TABLE MUST BE WITHIN 0-FFH BOUNDRY

MENU_TBL        EQU     ($ & 0FF00H) + 100H
        ORG     MENU_TBL

        DW      FORMAT_DISK             ;00     Special Disk Format section...
        DW      SET_FORMAT              ;01     This will set request to a specified drive. Drive Byte (0-3)
        DW      SET_DRIVE               ;02     Select a new disk drive in hardware (Note no assumptions about disk format)
```

```
        DW      SET_HOME        ;03     This will set head request to Track 0 of current drive
        DW      SET_TRACK               ;04     This will set head request to a specified track. Track Byte (0-NTRKS)
        DW      SET_SECTOR              ;05     This will set head request to a specified sector.  Sector Byte (0-NSCTRS)
        DW      SET_SIDE        ;06     This will set head request to a specified side. Side Byte (0=A, 1=B)
        DW      STEP_IN                 ;07     Test to step head IN one track at a time of current drive (continously)
        DW      GET_TRACK_ID            ;08     Read the current track ID (6 Bytes of data returned)
        DW      R_GET_TRACK_ID          ;09     RANDOM Read the current track ID (6 Bytes of data returned)
        DW      SEEK_NV                 ;0A     Seek to track with NO verify of current drive
        DW      SEEK_TRACK              ;0B     Seek to track (set by SET_TACK) and verify (on current drive)
        DW      R_SEEK_TRACK            ;0C     RANDOM Seek to tracks (on current drive).
        DW      READ_SECTOR             ;0D     Read data from specific sector (need track,side, use current drive).
        DW      S_SEC_READ              ;0E     Sequental Read sector test starting at track 0, sector 1
        DW      R_SEC_READ              ;0F     RANDOM Read sector test
        DW      WRITE_SECTOR            ;10     Write data to a specific sector (need track,side, use current drive).
        DW      S_SEC_RDWR              ;11     Sequental Sector R/W
        DW      R_SEC_RDWR              ;12     RANDOM sector Read/write
        DW      WR_BOOT                 ;13     Write boot sector data to current disk
        DW      GET_FORMAT              ;14     Get current drives Disk format number
        DW      READ_TRACK              ;15     Read and display a specified track on current drive
        DW      GET_ERROR_STRING ;16    Get string associate with an error number
        DW      DETAIL_SEC_DATA         ;17     Turn on/off detailed sector contents display
        DW      PIO_TEST        ;18     Hardware loop test of PIO's
        DW      MONITOR                 ;19     No extra data (Note in this case no SEND_OK is sent back to CPM system)
        DW      SHOW_SIGNON             ;1A     This will put 11 in Display TIL's as a test
        DW      RESET_ZFDC              ;1B     Reset the board and WD2793 chip
        DW      GET_WD_STATUS           ;1C     Return the WD2793 Status value
        DW      RAM_DUMP        ;1D     Display ZFDC board RAM & disk variables
        DW      DEBUG_MODE              ;1E     Set Debug display mode off
        DW      MULTI_READ              ;1F     Multi-sector read test
        DW      MULTI_WRITE             ;20     Multi-sector write test
        DW      FINISH                  ;21     Back to CP/M

MENU_TBL_END
MAX_MENU_TBL    EQU     (MENU_TBL_END - MENU_TBL)/2

FINISH: CALL    CRLF
        LD      C,CPM_RESET_DISKS;Must reset all disks to sync ZFDC board back with CPM
        CALL    BDOS
        LD      SP,(SP_SAVE)
        JP      Z,0H                    ;Return to CPM



; ++++++++++++++++++++++++++  CORE CPM BIOS TYPE ROUTINES ++++++++++++++++++++++++++++++++++++
; These modules are called by various parts of the program. They can be used directly in any CPM BIOS
;
; Commands to ZFDC board are always in [C].  Single byte paramaters (if any), are always sent in [A].
; In the few cases where a second extra paramater is needed, it is always in [B].
; Format a disk requires extra 3 paramaters. Drive # [A], format in [B], starting tarck in [D]
; Track ID's, Sector R/W's  will use [HL] (the "DMA adderss") to place the relevent data

; In all cases commands return either a Z (and NO_ERRORS_FLAG in [A]), or NZ with an error # in [A]
; In the few cases where a paramater is also returned, it will be in [D].


BIOS_SHOW_SIGNON:               ;Will incriment Track TIL's on ZFDC board to 0FFH as a diagnostic check
```

```
        LD      C,CMD_SHOW_SIGNON
        CALL    S100OUT
        CALL    WAIT_FOR_ACK            ;return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET

BIOS_RESET_ZFDC                         ;Note this is not the same as HOME. It resets a hung WD2793 chip only
        OUT     RESET_ZFDC_PORT,A;Do a hardware reset. Does not matter what is in [A]
        CALL    DELAY_1S        ;Allow Board time to re-initilize

        IN      A,(S100_DATA_B)         ;Check the board is there
        CP      A,CMD_HANDSHAKE
        JP      NZ,BOARD_ERROR

        LD      A,CMD_HANDSHAKE
        OUT     (S100_DATA_B),A         ;This clears up ints on ZFDC board
        CALL    WAIT_FOR_ACK            ;Ignore any error flags

        XOR     A,A                     ;set everything to zero
        LD      (@TRK),A
        LD      (@SIDE),A
        LD      (@DRIVE),A
        LD      (DEBUG_FLAG),A          ;Debug flag off initially on ZFDC board
        INC     A
        LD      (@SEC),A        ;Track 0, side A, sector 1
        LD      IX,STD8IBM              ;Set as default drive table: IBM 8" SSSD Disk
        LD      A,0
        LD      B,1                     ;Set initially for an 8" IBM
        CALL    BIOS_SET_FORMAT ;First select the initial drive
        CALL    NZ,SHOW_ERRORS
        RET

;       LD      C,CMD_RESET_ZFDC ;If we just want a software reset
;       CALL    S100OUT
;       CALL    WAIT_FOR_ACK            ;return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET



BIOS_SET_FORMAT:                ;Store Drive selection in [A], (A:=0, B:=1, C:=2, D:=3)
        PUSH    AF                      ;VIP, The disk's format is in [B] (0,1,2,3... 13H, or 0FFH)
        LD      C,CMD_SET_FORMAT
        CALL    S100OUT

        POP     AF                      ;ZFDC Board expects a 0H, 1H, 2H or 3H (Only)
        LD      C,A
        CALL    S100OUT

        LD      C,B                     ;ZFDC Board expects a Disk Format Table Number (0,1,2...13H, or 0FFH)
        CALL    S100OUT

        CALL    WAIT_FOR_ACK            ;return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET



BIOS_SET_DRIVE:                         ;Set the currently selected drive (0H, 1H, 2H or 3H)
```

```
        PUSH    AF
        LD      C,CMD_SET_DRIVE
        CALL    S100OUT

        POP     AF                      ;ZFDC Board expects a 0H, 1H, 2H or 3H (Only)
        LD      C,A
        CALL    S100OUT

        CALL    WAIT_FOR_ACK            ;return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET



BIOS_GET_DRIVE:                         ;Get the currently selected drive (1H, 2H, 3H or 4H)
        LD      C,CMD_GET_DRIVE
        CALL    S100OUT
        CALL    S100IN                  ;Note potential to lockup here (but unlightly)
        LD      D,A                     ;ZFDC Board will return a 1 byte drive number in [D]
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET


BIOS_GET_FORMAT                         ;Table Format number is returned in [D]
        LD      C,CMD_GET_FORMAT
        CALL    S100OUT
        CALL    S100IN                  ;Note potential to lockup here (but unlightly)
        LD      D,A                     ;ZFDC Board will return a 1 byte table number in [D]
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET


BIOS_SET_TRACK:                         ;Set Track for later sector read/write
        PUSH    AF                      ;Store the Track [A]
        LD      C,CMD_SET_TRACK
        CALL    S100OUT
        POP     AF
        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]
        RET


BIOS_SET_SIDE:                          ;Select Disk Side A (Default) or Side B
        PUSH    AF                      ;[A] = 0 for side A, [A] = 1 for side B
        LD      C,CMD_SET_SIDE
        CALL    S100OUT
        POP     AF
        LD      C,A
        CALL    S100OUT                 ;Send Selected side HEX number
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET


BIOS_SET_SECTOR:                ;Set Sector for later sector read/write
        PUSH    AF                      ;[A] contains the sector number
        LD      C,CMD_SET_SECTOR
```

```
        CALL    S100OUT
        POP     AF
        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]
        RET


BIOS_SET_HOME:                          ;Set current selected drive head to track 0
        LD      C,CMD_SET_HOME          ;Note this is a restore with NO verify. (disk my be for formatting)
        CALL    S100OUT
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET


BIOS_STEP_IN:                           ;Step head IN one track at a time on current drive
        LD      C,CMD_STEP_IN
        CALL    S100OUT
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET

BIOS_STEP_OUT:                          ;;Step head OUT one track at a time on current drive
        LD      C,CMD_STEP_OUT
        CALL    S100OUT
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET


BIOS_SEEK_NV:                           ;Seek (with no verify) to track already specified with BIOS_SET_TRACK
        LD      C,CMD_SEEK_NV
        CALL    S100OUT
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET


BIOS_SEEK_TRACK:                ;Seek (with verify) to track already specified with BIOS_SET_TRACK
        LD      C,CMD_SEEK_TRACK
        CALL    S100OUT
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET


BIOS_GET_TRACK_ID:                      ;Get the ID of track already specified by BIOS_SET_TRACK
        LD      C,CMD_GET_TRACK_ID      ;Note [HL] will point to a 6 byte Track ID info
        CALL    S100OUT
        CALL    WAIT_FOR_ACK            ;Wait for NO_ERRORS_FLAG to come back
        RET     NZ                      ;Check if ZFDC completed the command
        LD      B,6                     ;Else, 6 bytes total are then expected
ID_DATA:CALL    S100IN                  ;Note potential to lockup here (but unlightly)
        LD      (HL),A
        INC     HL
        DJNZ    ID_DATA
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET
```

```
;Note this is different than the next command where multiple sectors were read in one at a time.
;Here we read everything in one gulp - very fast

BIOS_MULTI_READ_SECTOR:                     ;<<< CORE BIOS MULTI-SECTOR READ ROUTINE >>
        LD      C,CMD_RD_MULTI_SECTOR       ;Routine assumes required Drive Table,Drive,Side,Track, and sector are already sent to board
        CALL    S100OUT                     ;(Note [HL]-> Sector DMA address)

        LD      A,(@REC_COUNT)              ;Number of sectors INCLUDING the current selected sector to read
        LD      C,A
        CALL    S100OUT

        CALL    WAIT_FOR_ACK                ;Wait for NO_ERRORS_FLAG to come back
        RET     NZ

MULTI_RD_DATA1:
        LD      E,(IX+SEC_SIZE_BYTES)       ;128,256,512 or 1024 byte sector size
        LD      D,(IX+SEC_SIZE_BYTES+1)

MULTI_RD_DATA:
        CALL    S100IN                      ;Note potential to lockup here & below (but unlightly)
        LD      (HL),A
        INC     HL                          ;Note [HL] will increase for each sector
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,MULTI_RD_DATA

        LD      C,'.'                       ;Show progress
        CALL    CO

        LD      A,(@REC_COUNT)
        DEC     A
        LD      (@REC_COUNT),A              ;For next time
        JP      NZ,MULTI_RD_DATA1

        CALL    WAIT_FOR_ACK                ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET



BIOS_READ_SECTOR:                   ;<<< CORE BIOS SECTOR READ ROUTINE >>
        LD      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
        RET     NZ

        LD      E,(IX+SEC_SIZE_BYTES)       ;128,256,512 or 1024 byte sector size
        LD      D,(IX+SEC_SIZE_BYTES+1)

RD_DATA:CALL    S100IN                      ;Note potential to lockup here & below (but unlightly)
        LD      (HL),A
        INC     HL
        DEC     DE
```

```
        LD      A,E
        OR      A,D
        JR      NZ,RD_DATA
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET


;Note this is different than the next command where multiple sectors were written one at a time.
;Here we write everything in one gulp - very fast

BIOS_MULTI_WRITE_SECTOR:          ;<<< CORE BIOS MULTI-SECTOR READ ROUTINE >>
        LD      C,CMD_WR_MULTI_SECTOR   ;Routine assumes required Drive Table,Drive,Side,Track, and sector are already sent to board
        CALL    S100OUT                 ;(Note [HL]-> Sector DMA address)

        LD      A,(@REC_COUNT)          ;Number of sectors INCLUDING the current selected sector to read
        LD      C,A
        CALL    S100OUT

        CALL    WAIT_FOR_ACK            ;Wait for NO_ERRORS_FLAG to come back
        RET     NZ

MULTI_WR_DATA1:
        LD      E,(IX+SEC_SIZE_BYTES)   ;128,256,512 or 1024 byte sector size
        LD      D,(IX+SEC_SIZE_BYTES+1)

MULTI_WR_DATA:
        LD      C,(HL)                  ;Remember S100OUT always requires data in [C]
        CALL    S100OUT                 ;(Note [HL]-> ZFDC Board)
        INC     HL
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,MULTI_WR_DATA

        LD      C,'.'                   ;Show progress
        CALL    CO

        LD      A,(@REC_COUNT)
        DEC     A
        LD      (@REC_COUNT),A          ;For next time
        JP      NZ,MULTI_WR_DATA1

        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET



BIOS_WRITE_SECTOR:                      ;<<< CORE BIOS SECTOR WRITE ROUTINE >>
        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
        RET     NZ
                                        ;If no errors, then we are ready to SEND the actual sector data bytes
        LD      E,(IX+SEC_SIZE_BYTES)   ;128,256,512 or 1024 byte sector size
        LD      D,(IX+SEC_SIZE_BYTES+1)
```

```
WR_DATA:LD      C,(HL)                  ;Remember S100OUT always requires data in [C]
        CALL    S100OUT                 ;(Note [HL]-> ZFDC Board)
        INC     HL
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,WR_DATA

        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET


BIOS_READ_TRACK:                ;MAX_TRACK_SIZE of Track data will be placed at [HL]
        LD      C,CMD_TRACK_DUMP ;Routine assumes required Drive is already selected on ZFDC board
        CALL    S100OUT

        CALL    WAIT_FOR_JOB_DONE;Wait for second NO_ERRORS_FLAG to come back, or user ABORT
        RET     NZ

        LD      DE,MAX_TRACK_SIZE;Will expect exactly MAX_TRACK_SIZE of data (3K)
RTD_DATA:
        CALL    S100IN                  ;Note potential to lockup here & below (but unlightly)
        LD      (HL),A
        INC     HL
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,RTD_DATA
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET

BIOS_GET_WD_SECTOR:                     ;Return WD2793 Sector Register in [D]
        LD      C,CMD_GET_WD_SECTOR
        CALL    S100OUT
        CALL    S100IN                  ;Note potential to lockup here (but unlightly)
        LD      D,A
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET

BIOS_GET_WD_TRACK:                      ;Return WD2793 Track Register in [D]
        LD      C,CMD_GET_WD_TRACK
        CALL    S100OUT
        CALL    S100IN                  ;Note potential to lockup here (but unlightly)
        LD      D,A
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET

BIOS_GET_WD_STATUS:                     ;Return WD2793 Status Register in [D]
        LD      C,CMD_GET_WD_STATUS
        CALL    S100OUT
        CALL    S100IN                  ;Note potential to lockup here (but unlightly)
        LD      D,A
        CALL    WAIT_FOR_ACK            ;Return Z (and NO_ERRORS_FLAG in [A]), or NZ with error # in [A]
        RET
```

```
BIOS_DEBUG_ON:                          ;Turn ZFDC Hex display to Debug mode
        LD      C,CMD_SET_DEBUG_ON
        CALL    S100OUT
        CALL    WAIT_FOR_ACK
        RET


BIOS_DEBUG_OFF:                         ;Turn ZFDC Hex display to normal Track/Sector display mode
        LD      C,CMD_SET_DEBUG_ON
        CALL    S100OUT
        CALL    WAIT_FOR_ACK
        RET


BIOS_ERROR_STRING:                      ;Get string associate with last error sent
        PUSH    AF                      ;Save Error Number
        LD      C,CMD_GET_ERROR_STRING
        CALL    S100OUT
        POP     AF

        LD      C,A                     ;Remember data out in [C]
        CALL    S100OUT                 ;Send Error Number

RD_STRING:
        CALL    S100IN                  ;Note potential to lockup here
        OR      A,A
        RET     Z
        LD      C,A
        CALL    CO
        JR      RD_STRING

;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;               END OF BIOS LIKE MODULES
;
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++




;----------- Echo whatever is sent from S-100 system to ZFDC board BYPASSING SOFTWARE.---------------
; (A simple hardware loop test for POI's only)

PIO_TEST:
        LD      HL,PIO_MSG
        CALL    PMSG
        LD      C,CMD_PIO_TEST
        CALL    S100OUT                 ;Send it to ZFDC Board

PIO_TEST1:
        CALL    CI                      ;Get a keyboard character
        LD      C,A
        CALL    S100OUT                 ;Send to ZFDC board
        CP      A,ESC
```

```
        JP      Z,LOOP
        CALL    S100IN                  ;Get back returned character
        LD      C,A
        AND     7FH                     ;Strip high bit just in case errors
        CALL    CO                      ;display returned character on CRT
        JR      PIO_TEST1



; Interact with the Monitor on board the ZFDC Board
MONITOR:LD      HL,MONITOR_MSG
        CALL    PMSG
        LD      C,CMD_MONITOR
        CALL    S100OUT                 ;Send it to ZFDC Board

MON1:   CALL    S100STAT        ;Anything at FDC Board
        JP      Z,TEST_SEND
        CALL    S100IN
        AND     7FH                     ;Strip high bit just in case
        LD      C,A
        CALL    CO                      ;Send it to video board
        JP      MON1                    ;Continue until nothing more
TEST_SEND:
        CALL    CONSTAT                 ;Anything from Keyboard to be sent
        JP      Z,MON1
        CALL    CI
        LD      C,A
        CALL    S100OUT                 ;ESC will abort the ZFDC monitor
        CP      A,ESC
        JP      Z,LOOP1                 ;Brake out of monitor loop here as well
        JP      MON1
LOOP1:  CALL    S100IN                  ;Need to flush port
        JP      LOOP



;-------------  Simple test of ZFDC Board handshaking  ----------------------------
; Will incriment the Track TIL's on ZFDC board to 0FFH as a diagnostic indicator

SHOW_SIGNON:
        LD      HL,SIGNON_MSG
        CALL    PMSG
        CALL    BIOS_SHOW_SIGNON ;Call the Command
        CALL    NZ,SHOW_ERRORS          ;If error returned display error string
        JP      LOOP


;-----------------  Reset the WD2793 Chip  ----------------------------------------

RESET_ZFDC:
        LD      HL,RESET_MSG
        CALL    PMSG
        CALL    BIOS_RESET_ZFDC
        JP      LOOP
```

```
;----------- Select Drive in hardware and assign a disk format to it ---------------
; This is a simple but very critical command. Do not change details if possible.

SET_FORMAT:
         LD       A,(@DRIVE)
         LD       (OLD_DRIVE),A             ;Store old drive in case an abort

         LD       HL,DRIVE_MSG
         CALL     PMSG
         CALL     GETCMD                    ;Get 0,1,2 or 3 for selected drive in [A]
         CP       A,'0'
         JP       Z,DRIVE_A
         CP       A,'1'
         JP       Z,DRIVE_B
         CP       A,'2'
         JP       Z,DRIVE_C
         CP       A,'3'
         JP       Z,DRIVE_D
INVALID_DRIVE:
         LD       HL,INVALID_DRIVE_MSG
         CALL     PMSG
         JP       LOOP

DRIVE_A:
         LD       HL,SEL_DRIVE_A_MSG
         CALL     PMSG
         LD       A,0
         JP       SET_DRIVE1

DRIVE_B:
         LD       HL,SEL_DRIVE_B_MSG
         CALL     PMSG
         LD       A,1
         JP       SET_DRIVE1

DRIVE_C:
         LD       HL,SEL_DRIVE_C_MSG
         CALL     PMSG
         LD       A,2
         JP       SET_DRIVE1

DRIVE_D:
         LD       HL,SEL_DRIVE_D_MSG
         CALL     PMSG
         LD       A,3
         JP       SET_DRIVE1

SET_DRIVE1:
         LD       (@DRIVE),A                ;Store the new selected drive here

         LD       HL,DRIVE_SIZE_MSG;We have the drive. Now we need to assign a FORMAT
         CALL     PMSG                      ;table to that drive
         CALL     GETCMD
         CP       ESC
```

```
        JP      Z,LOOP
        CP      A,'A'
        JR      Z,TABLE_8
        CP      A,'B'
        JR      Z,TABLE_8
        CP      A,'D'
        JR      Z,TABLE_8
        CP      A,'C'
        JR      Z,TABLE_5
        LD      HL,INVALID_CMD_MSG
        CALL    PMSG
        JP      LOOP

TABLE_8:
        LD      HL,TABLE_MSG_8
        JR      GOT_TABLE
TABLE_5:
        LD      HL,TABLE_MSG_5
GOT_TABLE:
        CALL    PMSG
        CALL    CRLF
        LD      HL,SEL_TBL_NUM_MSG      ;"Select table # from list above"
        CALL    PMSG

        CALL    GETHEX                  ;Get 00,01,02,03..... for selected table in [A]
        CALL    CRLF                    ;CRLF preserves AF!

        CP      A,DPL_COUNT             ;Is it within range ?
        JP      C,GOT_TABLE1
BAD_TABLE:
        LD      A,(OLD_DRIVE)
        LD      (@DRIVE),A              ;Get back old drive number
        LD      HL,BAD_TABLE_MSG
        CALL    PMSG
        JP      LOOP

GOT_TABLE1:                             ;We have a valid table number. Assign it
        LD      (@FORMAT),A             ;Now update the local IX table pointer here
        LD      B,A                     ;Must send requested format in [B]
        LD      A,(@DRIVE)              ;Must send requested disk in [A]

        CALL    BIOS_SET_FORMAT         ;Send the Command Drive in [A], Format in [B]
        CALL    NZ,SHOW_ERRORS
        JR      NZ,BAD_TABLE

        LD      A,(@FORMAT)             ;Now update the local IX table pointer here
        LD      HL,DPL_POINTERS         ;Pointers to table list are here
        ADD     A,A                     ;X2
        LD      C,A                     ;Add in offset
        LD      B,0                     ;High byte first in table
        ADD     HL,BC                   ;[HL] now points to correct pointer
        LD      E,(HL)
        INC     HL
        LD      D,(HL)
        PUSH    DE                      ;[DE]->[IX]
        POP     IX                      ;IX now points to correct Drive table
```

```
        LD      HL,NEW_TABLE_MSG ;"New Table will be..."
        CALL    PMSG
        CALL    CRLF
        CALL    SHOW_DRIVE_NAME          ;show current info
        CALL    CRLF
        JP      LOOP


;----------------------- Select Drive  ---------------

SET_DRIVE:                               ;0=A:,1=B:,2=C: & 3=D:
        LD      A,(@DRIVE)
        LD      (OLD_DRIVE),A            ;Store old drive in case an abort

        LD      HL,DRIVE_MSG
        CALL    PMSG
        CALL    GETCMD                   ;Get A,B,C or D for selected drive in [A]
        SUB     A,30H                    ;0H to 3H only
        CP      A,4H
        JP      NC,INVALID_DRIVE

        LD      (@DRIVE),A               ;Store it

        CALL    BIOS_SET_DRIVE           ;Send the Command Drive in [A]
        CALL    NZ,SHOW_ERRORS

        CALL    BIOS_GET_FORMAT          ;Must update locally the Drive table here as well
        CALL    NZ,SHOW_ERRORS
        LD      A,D                      ;Format number in in [D]
        LD      (@FORMAT),A              ;Now update the local IX table pointer here

        LD      HL,DPL_POINTERS          ;Pointers to table list are here
        ADD     A,A                      ;X2
        LD      C,A                      ;Add in offset
        LD      B,0                      ;High byte first in table
        ADD     HL,BC                    ;[HL] now points to correct pointer
        LD      E,(HL)
        INC     HL
        LD      D,(HL)
        PUSH    DE                       ;[DE]->[IX]
        POP     IX                       ;IX now points to correct Drive table
        JP      LOOP


;---------------------  Get Current Disk Paramater Table Number ------------------------

GET_FORMAT:
        LD      HL,GETTING_TABLE
        CALL    PMSG
        CALL    BIOS_GET_FORMAT          ;Return with Table number in [D]
        JP      Z,GET_TABLE1
        CALL    SHOW_ERRORS
        JP      LOOP
GET_TABLE1:
        LD      HL,DISK_TABLE_MSG
```

```
        CALL    PMSG                    ;[C] is not altered
        LD      A,D
        CALL    PACC
        CALL    CRLF
        JP      LOOP


;---------------------- Set Track for later sector read/write -------------------------

SET_TRACK:
        LD      HL,TRACK_MSG
        CALL    PMSG
        CALL    GETHEX                  ;Get 00,01,02,03..... for selected table
        PUSH    AF                      ;Save for later
        LD      (@TRK),A        ;Store for SEEK_TRACK Diagnostic
        LD      HL,WILL_SET_TRK_MSG
        CALL    PMSG
        POP     AF
        CALL    BIOS_SET_TRACK          ;Call the Command
        CALL    NZ,SHOW_ERRORS
        JP      LOOP



;-------------------  Select Disk Side A (Default) or Side B ----------------------

SET_SIDE:
        LD      HL,SIDE_MSG
        CALL    PMSG
        CALL    GETCMD                  ;Get A or B for selected side
        CP      A,'B'
        LD      A,1
        JP      Z,GOT_SIDE
        XOR     A,A                     ;Side A if anything else!
GOT_SIDE:
        PUSH    AF                      ;Save for later
        LD      HL,WILL_SET_SIDE_MSG
        CALL    PMSG
        POP     AF
        CALL    BIOS_SET_SIDE           ;Call the Command
        CALL    NZ,SHOW_ERRORS
        JP      LOOP


;------------------- Set Sector for later sector read/write ------------------

SET_SECTOR:
        LD      HL,SEC_MSG
        CALL    PMSG
        CALL    GETHEX                  ;Get selected sector
        PUSH    AF                      ;Save for later
        LD      HL,WILL_SET_SEC_MSG
        CALL    PMSG
        POP     AF
        CALL    BIOS_SET_SECTOR         ;Call the Command
        CALL    NZ,SHOW_ERRORS
```

```
        JP      LOOP


;--------------------  Set current selected drive head to track 0 ---------------

SET_HOME:
        LD      HL,HOME_MSG
        CALL    PMSG
        CALL    BIOS_SET_HOME           ;Call the Command
        CALL    NZ,SHOW_ERRORS
        JP      LOOP



;------------ Step head in one track at a time on current drive (Repeat until ESC)

STEP_IN:
        LD      HL,STEP_IN_MSG
        CALL    PMSG

STEP_IN3:
        CALL    BIOS_SET_HOME           ;Start at track 0
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        LD      HL,AT_TRK_0_MSG
        CALL    PMSG
        LD      D,0                     ;Store Track Count here

STEP_IN1:
        LD      HL,STEP_TO_MSG
        CALL    PMSG
        LD      A,D
        CALL    PACC
        LD      HL,H_MSG
        CALL    PMSG
        CALL    CRLF

        CALL    BIOS_STEP_IN            ;<<<<<<<< Call the Step-In Command
        CALL    NZ,SHOW_ERRORS

        CALL    CONSTAT                 ;Check for abort at Consol
        JP      Z,STEP_IN2
        CALL    CI
        CP      A,ESC
        JP      Z,LOOP
STEP_IN2:
        LD      A,D
        INC     A
        LD      B,A
        LD      A,(IX+NTRKS)            ;Store Track Count here
        CP      A,B
        JP      Z,STEP_IN3
        LD      D,B                     ;Store new track number
        JP      STEP_IN1
```

```
;----- Use Seek NV to step head in one track at a time on current drive (Repeat until ESC)

SEEK_NV:
        LD      HL,CMD_SEEK_NV_MSG
        CALL    PMSG

SEEK_IN3:
        CALL    BIOS_SET_HOME           ;Start at track 0
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        LD      HL,AT_TRK_0_MSG
        CALL    PMSG
        LD      D,0                     ;Store Track Count here

SEEK_IN1:
        LD      HL,R_SEEK_TO_MSG
        CALL    PMSG
        LD      A,D
        CALL    PACC
        LD      HL,H_MSG
        CALL    PMSG
        CALL    CRLF

        LD      A,D                     ;Track sent from[A]
        CALL    BIOS_SET_TRACK          ;Call the Command to set track number
        CALL    NZ,SHOW_ERRORS

        CALL    BIOS_SEEK_NV            ;<<<<<<<< Call the Seek NV Command
        CALL    NZ,SHOW_ERRORS

        CALL    CONSTAT                 ;Check for abort at Consol
        JP      Z,SEEK_IN2
        CALL    CI
        CP      A,ESC
        JP      Z,LOOP
SEEK_IN2:
        LD      A,D
        INC     A
        CP      A,(IX+NTRKS)            ;Store Track Count here
        JP      Z,SEEK_IN3
        LD      D,A                     ;Store new track number
        JP      SEEK_IN1



;---------------------- Step head out one track on current drive -----------------
STEP_OUT:
        LD      HL,STEP_OUT_MSG
        CALL    PMSG
        CALL    BIOS_STEP_OUT           ;Call the Command
        CALL    NZ,SHOW_ERRORS
        JP      LOOP
```

```
;----- Use Seek With Verify to step head in one track at a time (Repeat until ESC)

SEEK_TRACK:
         LD       HL,INPUT_TRACK_MSG
         CALL     PMSG
         CALL     GETHEX                    ;Get 00,01,02,03..... for selected table
         JP       C,LOOP                    ;Abort if ESC
         LD       (@TRK),A          ;Store for SEEK_TRACK Diagnostic

         LD       HL,R_SEEK_TO_MSG
         CALL     PMSG
         LD       A,(@TRK)          ;Get stored track #
         CALL     PACC
         LD       HL,H_MSG
         CALL     PMSG
         CALL     CRLF

         LD       A,(@TRK)          ;Get stored track #
         CALL     BIOS_SET_TRACK            ;Call the Command to set track number
         CALL     NZ,SHOW_ERRORS
         CALL     BIOS_SEEK_TRACK           ;<<<<<<<< Call the Seek V Command
         CALL     NZ,SHOW_ERRORS
         JP       LOOP



;----- RANDOM Track ID information Test, step head in one track at a time (Repeat until ESC)

R_SEEK_TRACK:
         LD       HL,R_SEEK_TRACK_MSG
         CALL     PMSG

         CALL     BIOS_SET_HOME             ;Start at track 0
         CALL     NZ,SHOW_ERRORS
         JP       NZ,LOOP
         JP       RSEEK_TRACK2

RSEEK_TRACK1:
         LD       HL,R_SEEK_TO_MSG
         CALL     PMSG
         LD       A,(@TRK)          ;Get stored track #
         CALL     PACC
         LD       HL,H_MSG
         CALL     PMSG
         CALL     CRLF

         LD       A,(@TRK)          ;Get stored track #
         CALL     BIOS_SET_TRACK            ;Call the Command to set track number
         CALL     NZ,SHOW_ERRORS
         CALL     BIOS_SEEK_TRACK           ;<<<<<<<< Call the Seek V Command
         CALL     NZ,SHOW_ERRORS

         CALL     CONSTAT                   ;Check for abort at Consol
         JP       Z,RSEEK_TRACK2
         CALL     CI
```

```
        CP      A,ESC
        JP      Z,LOOP
RSEEK_TRACK2:
        CALL    RANDOM_TRK_SEC          ;Get next random Track(/Sector)
        JP      RSEEK_TRACK1



;-------- Get Current Track ID  ------------

GET_TRACK_ID:
        LD      HL,TRACK_ID_MSG
        CALL    PMSG

        LD      HL,R_SEEK_TO_MSG
        CALL    PMSG
        LD      A,(@TRK)          ;Get stored track #
        CALL    PACC
        LD      HL,H_MSG
        CALL    PMSG
        CALL    CRLF

        LD      A,(@TRK)          ;Track sent from[A]
        CALL    BIOS_SET_TRACK          ;Call the Command to set track number
        CALL    NZ,SHOW_ERRORS
        CALL    BIOS_SEEK_TRACK         ;<<<<<<<< Call the Seek V Command
        CALL    NZ,SHOW_ERRORS

        LD      HL,TRACK_ID_BUFFER
        CALL    BIOS_GET_TRACK_ID;<<<GET Track ID
        CALL    NZ,SHOW_ERRORS

        CALL    DISPLAY_TRACK_ID ;Display the 6 Bytes on the CRT
        JP      LOOP



;--- RANDOM Track ID information Test, step head in one track at a time Repeat until ESC)

R_GET_TRACK_ID:
        LD      HL,R_TRACK_ID_MSG
        CALL    PMSG

        CALL    BIOS_SET_HOME           ;Start at track 0
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        JP      RTRACK_ID2

RTRACK_ID1:
        LD      HL,R_SEEK_TO_MSG
        CALL    PMSG
        LD      A,(@TRK)          ;Get stored track #
        CALL    PACC
        LD      HL,H_MSG
```

```
        CALL    PMSG
        CALL    CRLF

        LD      A,(@TRK)        ;Get stored track #
        CALL    BIOS_SET_TRACK          ;Call the Command to set track number
        CALL    NZ,SHOW_ERRORS
        CALL    BIOS_SEEK_TRACK         ;<<<<<<<< Call the Seek V Command
        CALL    NZ,SHOW_ERRORS

        LD      HL,TRACK_ID_BUFFER
        CALL    BIOS_GET_TRACK_ID;<<<GET Track ID
        CALL    NZ,SHOW_ERRORS

        CALL    DISPLAY_TRACK_ID ;Display the 6 Bytes on the CRT

        CALL    CONSTAT                 ;Check for abort at Consol
        JP      Z,RTRACK_ID2
        CALL    CI
        CP      A,ESC
        JP      Z,LOOP
RTRACK_ID2:
        CALL    RANDOM_TRK_SEC          ;Get next random Track(/Sector)
        JP      RTRACK_ID1


;--------------------- MULTI-SECTOR READ TEST ----------------------------------------------
;Test from ZFDC Board Multi-sector read CMD.  Note this is different than the previous commands
;where multiple sectors were read in one at a time. Here we read everything in one gulp - very fast
;

MULTI_READ:
        LD      HL,MULTI_RD_SECTOR_MSG
        CALL    PMSG
MULTI_READ_SEC1:
        LD      HL,INPUT_TRACK_MSG
        CALL    PMSG
        CALL    GETHEX                  ;Get 00,01,02,03..... for selected table
        JP      C,LOOP                  ;Abort if ESC
        LD      (@TRK),A        ;Store for SEEK_TRACK Diagnostic
        CALL    BIOS_SET_TRACK          ;Call the Command
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP                 ;Abort if error

MULTI_READ_SEC2:
        LD      HL,INPUT_SECTOR_MSG
        CALL    PMSG
        CALL    GETHEX                  ;Get 00,01,02,03..... for selected table
        JP      C,LOOP                  ;Abort if ESC
        OR      A,A
        JR      NZ,RMULTI_SEC_VALID
        LD      HL,NO_SEC_0_MSG
        CALL    PMSG
        JR      MULTI_READ_SEC2         ;Try again

RMULTI_SEC_VALID:
        LD      (@SEC),A        ;Store for SEEK_TRACK Diagnostic
```

```
        CALL    BIOS_SET_SECTOR          ;Call the Command
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP                  ;Abort if error

RMULTI_SEC1_VALID:
        BIT     SIDE_BIT,(IX+HW_BYTE)    ;Is it a DS drive
        JR      Z,MULTI_READ_SEC3;no then get number of sectors

        LD      HL,INPUT_SIDE_MSG
        CALL    PMSG
        CALL    GETCMD                   ;Get A or B
        CP      A,ESC
        JP      Z,LOOP                   ;Abort if ESC

        CP      A,'A'
        JR      NZ,RMULTI_NOT_A_SIDE
        XOR     A,A
        LD      (@SIDE),A
        CALL    BIOS_SET_SIDE
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        JR      MULTI_READ_SEC3          ;go to get number of sectors

RMULTI_NOT_A_SIDE:
        CP      A,'B'
        JR      NZ,RMULTI_NOT_B_SIDE
        LD      A,1
        LD      (@SIDE),A
        CALL    BIOS_SET_SIDE
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        JR      MULTI_READ_SEC3          ;go to get number of sectors

RMULTI_NOT_B_SIDE:
        LD      HL,INVALID_SIDE_MSG
        CALL    PMSG
        JR      RMULTI_SEC1_VALID

MULTI_READ_SEC3:
        LD      HL,NUM_OF_RD_SEC ;"Enter number of sectors to read"
        CALL    PMSG
        CALL    GETHEX                   ;Get number of sectors to read
        JP      C,LOOP                   ;Abort if ESC
        LD      (@REC_COUNT),A           ;Store for Multi sector read routine

        LD      HL,SECTOR_BUFFER ;Location where data will be placed

        CALL    BIOS_MULTI_READ_SECTOR  ;<<<<<<<<<<<<<<<<<<<<<<<<

        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP                  ;Abort if error

        LD      HL,MULTI_SEC_ROK ;Message that things went OK.
        CALL    PMSG
        JP      LOOP
```

```
;---------------------- SECTOR READ ROUTINE -----------------------------------------------
;Read sector data from ZFDC Board selected disk, side, track, sector and send it to S-100 System

READ_SECTOR:
        LD      HL,READ_SECTOR_MSG
        CALL    PMSG
READ_SEC1:
        LD      HL,INPUT_TRACK_MSG
        CALL    PMSG
        CALL    GETHEX                  ;Get 00,01,02,03..... for selected table
        JP      C,LOOP                  ;Abort if ESC
        LD      (@TRK),A        ;Store for SEEK_TRACK Diagnostic
        CALL    BIOS_SET_TRACK          ;Call the Command
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
READ_SEC2:
        LD      HL,INPUT_SECTOR_MSG
        CALL    PMSG
        CALL    GETHEX                  ;Get 00,01,02,03..... for selected table
        JP      C,LOOP                  ;Abort if ESC
        OR      A,A
        JR      NZ,SEC_VALID
        LD      HL,NO_SEC_0_MSG
        CALL    PMSG
        JR      READ_SEC2

SEC_VALID:
        LD      (@SEC),A        ;Store for SEEK_TRACK Diagnostic
        CALL    BIOS_SET_SECTOR         ;Call the Command
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

SEC_VALID1:
        BIT     SIDE_BIT,(IX+HW_BYTE)   ;Is it a DS drive
        JR      Z,READ_SEC3
        LD      HL,INPUT_SIDE_MSG
        CALL    PMSG
        CALL    GETCMD                  ;Get A or B
        CP      A,ESC
        JP      Z,LOOP                  ;Abort if ESC
        CP      A,'A'
        JR      NZ,NOT_A_SIDE
        XOR     A,A
        LD      (@SIDE),A
        CALL    BIOS_SET_SIDE
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        JR      READ_SEC3
NOT_A_SIDE:
        CP      A,'B'
        JR      NZ,NOT_B_SIDE
        LD      A,1
        LD      (@SIDE),A
        CALL    BIOS_SET_SIDE
        CALL    NZ,SHOW_ERRORS
```

```
        JP      NZ,LOOP
        JR      READ_SEC3
NOT_B_SIDE:
        LD      HL,INVALID_SIDE_MSG
        CALL    PMSG
        JR      SEC_VALID1

READ_SEC3:
        LD      HL,SECTOR_BUFFER ;Location where data will be placed
        CALL    BIOS_READ_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      HL,SECTOR_DATA_RD
        CALL    PMSG

        LD      HL,SECTOR_BUFFER ;Location where data is
        LD      A,0FFH
        LD      (HALT_CHECK_FLAG),A       ;Set so we have a keyboard check for CRT display
        CALL    DISPLAY_SECTOR
        JP      LOOP


;-------- RANDOM Sector Read Test (Repeat until ESC)

R_SEC_READ:
        LD      HL,R_SEC_RD_MSG
        CALL    PMSG

        CALL    BIOS_SET_HOME             ;Start at track 0
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        LD      A,1
        LD      (@SEC),A
        XOR     A,A
        LD      (@TRK),A
        LD      (@SIDE),A
        JP      RSEC_READ2

RSEC_READ1:
        LD      HL,R_SEEK_TO_MSG ;Will seek to track XXH sector yyH.
        CALL    PMSG
        LD      A,(@TRK)
        CALL    PACC
        LD      HL,R_SECTOR_MSG
        CALL    PMSG
        LD      A,(@SEC)
        CALL    PACC
        LD      HL,H_MSG
        CALL    PMSG
        CALL    CRLF

        LD      A,(@TRK)          ;Track sent from[A]
        CALL    BIOS_SET_TRACK            ;Call the Command to set track number
        CALL    NZ,SHOW_ERRORS
        LD      A,(@SEC)          ;Sector sent from[A]
```

```
        CALL    BIOS_SET_SECTOR         ;Call the Command to set sector number
        CALL    NZ,SHOW_ERRORS

        LD      A,(@SIDE)               ;Sector sent from[A]
        CALL    BIOS_SET_SIDE           ;Call the Command to set side
        CALL    NZ,SHOW_ERRORS

;       CALL    BIOS_SEEK_TRACK         ;To bypass re-seek module in read sector code
;       CALL    NZ,SHOW_ERRORS          ;For fixing bugs only

        LD      HL,SECTOR_BUFFER ;Location where data will be placed
        CALL    BIOS_READ_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      HL,SECTOR_DATA_RD
        CALL    PMSG
        LD      HL,SECTOR_BUFFER ;Location where data is
        LD      A,0H
        LD      (HALT_CHECK_FLAG),A     ;Set so we have NO keyboard check for CRT display
        CALL    DISPLAY_SECTOR

        CALL    CONSTAT                 ;Check for abort at Consol
        JR      Z,RSEC_READ2
        CALL    CI
        CP      A,ESC
        JP      Z,LOOP

RSEC_READ2:
        CALL    RANDOM_TRK_SEC          ;Get next random number for Track AND Sector
        JR      RSEC_READ1



;-------- SEQUENTAL Sector Read Test (Repeat until ESC) -----------------

S_SEC_READ:
        LD      HL,S_SEC_RD_MSG
        CALL    PMSG

        CALL    BIOS_SET_HOME           ;Start at track 0
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        LD      A,1
        LD      (@SEC),A
        XOR     A,A
        LD      (@TRK),A
        LD      (@SIDE),A

SSEC_READ1:
        LD      HL,R_SEEK_TO_MSG ;Will seek to track XXH sector yyH
        CALL    PMSG
        LD      A,(@TRK)
        CALL    PACC
        LD      HL,R_SECTOR_MSG
        CALL    PMSG
```

```
        LD      A,(@SEC)
        CALL    PACC
        LD      HL,H_MSG
        CALL    PMSG
        CALL    CRLF

        LD      A,(@TRK)
        CALL    BIOS_SET_TRACK          ;Call the Command to set track number
        CALL    NZ,SHOW_ERRORS

        LD      A,(@SEC)        ;Sector sent from[A]
        CALL    BIOS_SET_SECTOR         ;Call the Command to set track number
        CALL    NZ,SHOW_ERRORS

        LD      A,(@SIDE)               ;Side sent from[A]
        CALL    BIOS_SET_SIDE           ;Call the Command to set side
        CALL    NZ,SHOW_ERRORS

;       CALL    BIOS_SEEK_TRACK         ;To bypass re-seek module in read sector code
;       CALL    NZ,SHOW_ERRORS          ;For fixing bugs only

        LD      HL,SECTOR_BUFFER ;Location where data will be placed
        CALL    BIOS_READ_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      HL,SECTOR_DATA_RD
        CALL    PMSG
        LD      HL,SECTOR_BUFFER ;Location where data is
        LD      A,0H
        LD      (HALT_CHECK_FLAG),A     ;Set so we have NO keyboard check for CRT display
        CALL    DISPLAY_SECTOR

        CALL    CONSTAT                 ;Check for abort at Consol
        JR      Z,SSEC_READ2
        CALL    CI
        CP      A,ESC
        JP      Z,LOOP

SSEC_READ2:
        CALL    NEXT_TRK_SEC            ;Get next number for Track/Sector.
        JR      SSEC_READ1


;--------------------- MULTI-SECTOR READ ROUTINE -----------------------------------------------
;Multi-sector read test from ZFDC Board using selected disk, side, track, sector

MULTI_WRITE:
        LD      HL,MULTI_WR_SECTOR_MSG
        CALL    PMSG
MULTI_WRITE_SEC1:
        LD      HL,INPUT_TRACK_MSG
        CALL    PMSG
        CALL    GETHEX                  ;Get 00,01,02,03..... for selected table
        JP      C,LOOP                  ;Abort if ESC
        LD      (@TRK),A        ;Store for SEEK_TRACK Diagnostic
```

```
        CALL    BIOS_SET_TRACK          ;Call the Command
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP                 ;Abort if error

MULTI_WRITE_SEC2:
        LD      HL,INPUT_SECTOR_MSG
        CALL    PMSG
        CALL    GETHEX                  ;Get 00,01,02,03..... for selected table
        JP      C,LOOP                  ;Abort if ESC
        OR      A,A
        JR      NZ,WMULTI_SEC_VALID
        LD      HL,NO_SEC_0_MSG
        CALL    PMSG
        JR      MULTI_WRITE_SEC2 ;Try again

WMULTI_SEC_VALID:
        LD      (@SEC),A         ;Store for SEEK_TRACK Diagnostic
        CALL    BIOS_SET_SECTOR         ;Call the Command
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP                 ;Abort if error

WMULTI_SEC1_VALID:
        BIT     SIDE_BIT,(IX+HW_BYTE)   ;Is it a DS drive
        JR      Z,MULTI_WRITE_SEC3      ;no then get number of sectors

        LD      HL,INPUT_SIDE_MSG
        CALL    PMSG
        CALL    GETCMD                  ;Get A or B
        CP      A,ESC
        JP      Z,LOOP                  ;Abort if ESC

        CP      A,'A'
        JR      NZ,WMULTI_NOT_A_SIDE
        XOR     A,A
        LD      (@SIDE),A
        CALL    BIOS_SET_SIDE
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        JR      MULTI_WRITE_SEC3        ;go to get number of sectors

WMULTI_NOT_A_SIDE:
        CP      A,'B'
        JR      NZ,WMULTI_NOT_B_SIDE
        LD      A,1
        LD      (@SIDE),A
        CALL    BIOS_SET_SIDE
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        JR      MULTI_WRITE_SEC3        ;go to get number of sectors

WMULTI_NOT_B_SIDE:
        LD      HL,INVALID_SIDE_MSG
        CALL    PMSG
        JR      WMULTI_SEC1_VALID

MULTI_WRITE_SEC3:
```

```
        LD      HL,NUM_OF_RD_SEC ;"Enter number of sectors to read"
        CALL    PMSG
        CALL    GETHEX                  ;Get number of sectors to read
        JP      C,LOOP                  ;Abort if ESC
        LD      (@REC_COUNT),A          ;Store for Multi sector read routine

        LD      HL,SECTOR_BUFFER ;Location where data will be placed

        CALL    BIOS_MULTI_WRITE_SECTOR  ;<<<<<<<<<<<<<<<<<<<<<<<<

        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP                 ;Abort if error

        LD      HL,MULTI_SEC_WOK ;Message that things went OK.
        CALL    PMSG
        JP      LOOP


;--------------------- SECTOR WRITE ROUTINE ------------------------------------------------
;Write sector data from S-100 system to a selected disk, side, track, sector via the ZFDC Board

WRITE_SECTOR:
        LD      HL,WRITE_SECTOR_MSG
        CALL    PMSG
WRITE_SEC1:
        LD      HL,INPUT_TRACK_MSG
        CALL    PMSG
        CALL    GETHEX                  ;Get 00,01,02,03..... for selected table
        JP      C,LOOP                  ;Abort if ESC
        LD      (@TRK),A        ;Store for SEEK_TRACK Diagnostic
        CALL    BIOS_SET_TRACK          ;Call the Command
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
WRITE_SEC2:
        LD      HL,INPUT_SECTOR_MSG
        CALL    PMSG
        CALL    GETHEX                  ;Get 00,01,02,03..... for selected table
        JP      C,LOOP                  ;Abort if ESC
        OR      A,A
        JR      NZ,WSEC_VALID
        LD      HL,NO_SEC_0_MSG
        CALL    PMSG
        JR      WRITE_SEC2
WSEC_VALID:
        LD      (@SEC),A        ;Store for SEEK_TRACK Diagnostic
        CALL    BIOS_SET_SECTOR         ;Call the Command
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
WSEC_VALID1:
        BIT     SIDE_BIT,(IX+HW_BYTE)   ;Is it a DS drive
        JR      Z,WRITE_SEC3
        LD      HL,INPUT_SIDE_MSG
        CALL    PMSG
        CALL    GETCMD                  ;Get A or B
        CP      A,ESC
        JP      Z,LOOP                  ;Abort if ESC
```

```
        CP      A,'A'
        JR      NZ,WNOT_A_SIDE
        XOR     A,A
        LD      (@SIDE),A
        CALL    BIOS_SET_SIDE
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        JR      WRITE_SEC3
WNOT_A_SIDE:
        CP      A,'B'
        JR      NZ,WNOT_B_SIDE
        LD      A,1
        LD      (@SIDE),A
        CALL    BIOS_SET_SIDE
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        JR      WRITE_SEC3
WNOT_B_SIDE:
        LD      HL,INVALID_SIDE_MSG
        CALL    PMSG
        JR      WSEC_VALID1

WRITE_SEC3:
        LD      HL,SECTOR_BUFFER ;Location where data will be obtained from
        CALL    BIOS_WRITE_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      Z,LOOP

        LD      HL,SECTOR_DATA_WR
        CALL    PMSG
        LD      HL,SECTOR_BUFFER ;Location where data will be placed
        LD      A,0FFH
        LD      (HALT_CHECK_FLAG),A       ;Set so we have a keyboard check for CRT display
        CALL    DISPLAY_SECTOR
        JP      LOOP



;-------- RANDOM Sector Read/Write Test (Repeat until ESC) ------------------------
; Random R/W a sector on disk. A random sector is read, data is inverted, written back, compared,
; then origional data is written back.

R_SEC_RDWR:
        LD      HL,R_SEC_RDWR_MSG
        CALL    PMSG

        CALL    BIOS_SET_HOME             ;Start at track 0
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        LD      A,0                       ;Not 1, because of jump to RSEC_RDWR2 below
        LD      (@SEC),A
        XOR     A,A
        LD      (@TRK),A
        JP      RSEC_RDWR2
```

```
RSEC_RDWR1:
        LD      HL,R_SEEK_TO_MSG ;Will seek to track XXH sector yyH.
        CALL    PMSG
        LD      A,(@TRK)
        CALL    PACC
        LD      HL,R_SECTOR_MSG
        CALL    PMSG
        LD      A,(@SEC)
        CALL    PACC
        LD      HL,H_MSG
        CALL    PMSG
        CALL    CRLF

        LD      A,(@TRK)          ;Track sent from[A]
        CALL    BIOS_SET_TRACK            ;Call the Command to set track number
        CALL    NZ,SHOW_ERRORS
        LD      A,(@SEC)          ;Sector sent from[A]
        CALL    BIOS_SET_SECTOR           ;Call the Command to set track number
        CALL    NZ,SHOW_ERRORS

;       CALL    BIOS_SEEK_TRACK           ;To bypass re-seek module in read sector code
;       CALL    NZ,SHOW_ERRORS            ;For fixing bugs only

        LD      HL,SECTOR_BUFFER ;Location where first data read will be placed
        CALL    BIOS_READ_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        CALL    FLIP_SECTOR_BUFFER        ;Complement the contents of the sector buffer

        LD      HL,SECTOR_BUFFER ;Write complement data back to same sector
        CALL    BIOS_WRITE_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      HL,SECTOR_BUFFER2;Read data a second time to this location
        CALL    BIOS_READ_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        CALL    COMPARE_SECTOR_BUFFER     ;Check the read back data matches
        JR      Z,PUT_BACK                ;If so put fack origional data

        LD      HL,RW_MISMATCH_MSG
        CALL    PMSG

PUT_BACK:                                 ;Restore to the origional contents
        CALL    FLIP_SECTOR_BUFFER        ;Complement the contents of the sector buffer again

        LD      HL,SECTOR_BUFFER ;Write back the origional data
        CALL    BIOS_WRITE_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      HL,SECTOR_DATA_RD
        CALL    PMSG
```

```
        LD      HL,SECTOR_BUFFER ;Location where data is
        LD      A,0H
        LD      (HALT_CHECK_FLAG),A       ;Set so we have NO keyboard check for CRT display
        CALL    DISPLAY_SECTOR

        CALL    CONSTAT                   ;Check for abort at Consol
        JR      Z,RSEC_RDWR2
        CALL    CI
        CP      A,ESC
        JP      Z,LOOP

RSEC_RDWR2:
        CALL    RANDOM_TRK_SEC            ;Get next random number for Track AND Sector
        JP      RSEC_RDWR1

;
FLIP_SECTOR_BUFFER                        ;Complememt the contents of a sector buffer in RAM
        LD      HL,SECTOR_BUFFER
        LD      E,(IX+SEC_SIZE_BYTES)     ;128,256,512 or 1024 byte sector size
        LD      D,(IX+SEC_SIZE_BYTES+1)
FLIP_1: LD      A,(HL)
        LD      A,31H                     ;<<<<<<<<<<<<<
        LD      (HL),A
        INC     HL
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,FLIP_1
        RET

COMPARE_SECTOR_BUFFER                     ;Check the contents of two sector buffres are the same
        LD      HL,SECTOR_BUFFER
        LD      BC,SECTOR_BUFFER2
        LD      E,(IX+SEC_SIZE_BYTES)     ;128,256,512 or 1024 byte sector size
        LD      D,(IX+SEC_SIZE_BYTES+1)
COMPARE:LD      A,(BC)
        CP      A,(HL)
        JR      NZ,MIS_MATCH
        INC     BC
        INC     HL
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,COMPARE
        XOR     A,A                       ;Return Z if the same
        RET
MIS_MATCH:
        XOR     A,A
        DEC     A                         ;Return NZ if different
        RET

;

;-------- SEQUENTAL Sector Read/Write Test (Repeat until ESC) ------------------------
; Sequental R/W a sector on disk. Starting on Track 0, Sec 1, side A a sector is read, data is
; inverted, written back, compared, then origional data is written back.
```

```
S_SEC_RDWR:
        LD      HL,S_SEC_RDWR_MSG
        CALL    PMSG

        CALL    BIOS_SET_HOME           ;Start at track 0
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        LD      A,0                     ;Not 1, because of jump to RSEC_RDWR2 below
        LD      (@SEC),A
        XOR     A,A
        LD      (@TRK),A
        JP      SSEC_RDWR2

SSEC_RDWR1:
        LD      HL,R_SEEK_TO_MSG ;Will seek to track XXH sector yyH.
        CALL    PMSG
        LD      A,(@TRK)
        CALL    PACC
        LD      HL,R_SECTOR_MSG
        CALL    PMSG
        LD      A,(@SEC)
        CALL    PACC
        LD      HL,H_MSG
        CALL    PMSG
        CALL    CRLF

        LD      A,(@TRK)          ;Track sent from[A]
        CALL    BIOS_SET_TRACK          ;Call the Command to set track number
        CALL    NZ,SHOW_ERRORS
        LD      A,(@SEC)          ;Sector sent from[A]
        CALL    BIOS_SET_SECTOR         ;Call the Command to set track number
        CALL    NZ,SHOW_ERRORS

;       CALL    BIOS_SEEK_TRACK         ;To bypass re-seek module in read sector code
;       CALL    NZ,SHOW_ERRORS          ;For fixing bugs only

        LD      HL,SECTOR_BUFFER ;Location where first data read will be placed
        CALL    BIOS_READ_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        CALL    FLIP_SECTOR_BUFFER      ;Complement the contents of the sector buffer

        LD      HL,SECTOR_BUFFER ;Write complement data back to same sector
        CALL    BIOS_WRITE_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      HL,SECTOR_BUFFER2;Read data a second time to this location
        CALL    BIOS_READ_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        CALL    COMPARE_SECTOR_BUFFER   ;Check the read back data matches
        JR      Z,SPUT_BACK             ;If so put back the origional data
```

```
        LD      HL,RW_MISMATCH_MSG
        CALL    PMSG

SPUT_BACK:                              ;Restore to the origional contents
        CALL    FLIP_SECTOR_BUFFER      ;Complement the contents of the sector buffer again

        LD      HL,SECTOR_BUFFER ;Write back the origional data
        CALL    BIOS_WRITE_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      HL,SECTOR_DATA_RD
        CALL    PMSG
        LD      HL,SECTOR_BUFFER ;Location where data is
        LD      A,0H
        LD      (HALT_CHECK_FLAG),A     ;Set so we have NO keyboard check for CRT display
        CALL    DISPLAY_SECTOR

        CALL    CONSTAT                 ;Check for abort at Consol
        JR      Z,SSEC_RDWR2
        CALL    CI
        CP      A,ESC
        JP      Z,LOOP

SSEC_RDWR2:
        CALL    NEXT_TRK_SEC            ;Get next number for Track/Sector.
        JP      SSEC_RDWR1



;
;-------------------------- Write Boot Sector -----------------------------
; This routine will write a boot sector to disk. It will ask for a location
; of the boot data in RAM first.  Note depending on the size of the sectors
; on the disk it will write 128,256,512 or 1K bytes. Be sure you have first set
; correct drive table number.

WR_BOOT:LD      HL,WR_BOOT_MSG
        CALL    PMSG
        CALL    BIOS_SET_HOME
        CALL    GETHEX
        LD      H,A
        CALL    GETHEX
        LD      L,A                     ;[HL] now has location of boot data
        PUSH    HL                      ;Save 2X for below
        PUSH    HL
        CALL    CRLF

        XOR     A,A
        CALL    BIOS_SET_TRACK
        CALL    NZ,SHOW_ERRORS
        INC     A
        CALL    BIOS_SET_SECTOR
        CALL    NZ,SHOW_ERRORS
```

```
        POP     HL                      ;[HL] has location of boot data
        CALL    BIOS_WRITE_SECTOR
        CALL    NZ,SHOW_ERRORS

        LD      HL,SECTOR_DATA_WR
        CALL    PMSG
        CALL    CRLF
        POP     HL                      ;[HL] has location of boot data
        CALL    DISPLAY_SECTOR
        JP      LOOP

;---------------- Read & Display a Complete Track (as written on disk) ------------------------------
; This routine will read the raw encoded data of a complete track from the current disk.  It will
; ask for a track # first. Be sure you have first set correct drive table number.

READ_TRACK:
        LD      HL,RD_TRACK_MSG
        CALL    PMSG
        CALL    GETHEX
        LD      (@TRK),A
        CALL    CRLF

        CALL    BIOS_SET_HOME
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP
        LD      A,(@TRK)
        CALL    BIOS_SET_TRACK
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        CALL    BIOS_SEEK_TRACK         ;<<<<<<<< Call the Seek V Command
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      HL,SECTOR_BUFFER ;Location where data will be placed
        CALL    BIOS_READ_TRACK         ;<<<<<<<< Read actual track

        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      HL,RD1_TRACK_MSG
        CALL    PMSG

        LD      HL,SECTOR_BUFFER ;Location where data will be placed
        LD      A,H
        CALL    PACC
        LD      A,L
        CALL    PACC
        LD      HL,H_MSG
        CALL    PMSG
        CALL    CRLF

        LD      HL,SECTOR_BUFFER ;Location where data is
        CALL    DISPLAY_TRACK

        LD      HL,END_TRACK_MSG
```

```
        CALL    PMSG
        CALL    CRLF
        JP      LOOP

;------------------- FORMAT the current disk -----------------------------

FORMAT_DISK:
        LD      HL,FORMATTING1_MSG      ;"Will format current disk ("
        CALL    PMSG
        CALL    BIOS_GET_DRIVE          ;be sure we have the right disk (returned in [D])
        LD      A,D
        ADD     A,'A'                   ;A,B,C,D
        LD      C,A
        CALL    CO
        LD      HL,FORMATTING2_MSG      ;":) to this format:-"
        CALL    PMSG
        CALL    CRLF
        CALL    SHOW_DRIVE_NAME         ;show format info
        LD      HL,SURE_MSG             ;"Are you sure"
        CALL    PMSG
        CALL    GETCMD
        CP      ESC
        JP      Z,LOOP
        CP      A,'Y'
        JP      Z,FORMAT_1
        LD      HL,FORM_ABORT_MSG;"Will abort format"
        CALL    PMSG
        CALL    CRLF
        JP      LOOP

FORMAT_1:

        LD      HL,START_TRACK_MSG      ;"Format starting on Track (Usually 00H or 02H):-"
        CALL    PMSG
        CALL    GETHEX                  ;usually 00 or 02  in [A]
        LD      (CURRENT_FORMAT_TRACK),A

        LD      HL,DISK_FORMAT_MSG
        CALL    PMSG
        CALL    SHOW_DRIVE_NAME         ;show current info
        CALL    CRLF

        CALL    BIOS_SET_HOME           ;Start at track 0
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

NEXT_TRACK:
        LD      C,CMD_FORMAT_TRACK      ;Format a complete track
        CALL    S100OUT

        LD      A,(CURRENT_FORMAT_TRACK) ;get requested track
        LD      C,A
        CALL    S100OUT                 ;Send track number

        LD      C,CONFIRM_FORMAT        ;Now send SPECIAL OK to FORMAT Disk flag
        CALL    S100OUT
```

```
        LD      HL,CURRENT_TRACK_MSG
        CALL    PMSG
        LD      A,(CURRENT_FORMAT_TRACK) ;Show the requested track
        CALL    PACC

                                         ;<<< Now wait until track is formatted >>>
WAIT_F: CALL    S100STAT         ;Wait until ZFDC Board is ready
        JP      NZ,TRACK_DONE            ;NZ, something there!
        CALL    CONSTAT                  ;Is there an ESC from user at the console
        JR      Z,WAIT_F         ;Nothing then wait some more
        CALL    CI
        CP      A,ESC                    ;Was an ESC character eneterd
        JR      Z,FORMAT_ABORT1
        JR      WAIT_F

TRACK_DONE:
        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
        CALL    NZ,SHOW_ERRORS
        JP      NZ,FORMAT_ABORT          ;If error abort

        LD      A,(CURRENT_FORMAT_TRACK) ;Else point to the next track
        INC     A
        LD      (CURRENT_FORMAT_TRACK),A ;Save new requested track
        CP      A,(IX+NTRKS)              ;Are we done yet
        JP      NZ,NEXT_TRACK

FORMAT_FINISHED:
        LD      HL,FORMAT_FINISHED_MSG
        CALL    PMSG
        CALL    BIOS_SET_HOME            ;Start at track 0
        CALL    NZ,SHOW_ERRORS
        JP      POST_FORMAT

FORMAT_ABORT1:
        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
        CALL    NZ,SHOW_ERRORS
FORMAT_ABORT:
        LD      HL,FORMAT_ABORTED_MSG
        CALL    PMSG
        CALL    BIOS_SET_HOME            ;Start at track 0
        CALL    NZ,SHOW_ERRORS
        JP      LOOP

POST_FORMAT:                             ;For MSDOS Disks the disk needs to be modified/Initilized
        LD      A,(IX+SPECIAL_FLAG)
        OR      A,A
        JP      Z,LOOP                   ;If Z then no after formatting mods required

        CP      A,1                      ;1.44M MSDOS Disks
        JP      Z,MSDOS144
        CP      A,2
        JP      Z,MSDOS12
        CP      A,3                      ;Old 360K MSDOS 2.0 (9 sectors/track) disk
```

```
        JP      Z,MSDOS360
        CP      A,4                     ;Old 360BK MSDOS 1.1 (8 sectors/track) disk
        JP      Z,MSDOS360B

        LD      HL,UNKNOWN_MOD_MSG
        CALL    PMSG
        XOR     A,A
        DEC     A
        JP      LOOP

MSDOS144:                               ;Initilize sectors on 3.5" 1.44M MSDOS/Windows disk
        XOR     A,A                     ;Fill in first FAT area
        CALL    BIOS_SET_TRACK
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      A,2                     ;FAT at sector 2 with F0,FF,FF
        CALL    BIOS_SET_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        CALL    ZERO_BUFFER             ;First fill sector area with 0's

        LD      A,0F0H                  ;<-- Note
        LD      HL,SECTOR_BUFFER ;[HL] will have location of sector data
        LD      [HL],A
        INC     HL
        LD      A,0FFH
        LD      [HL],A
        INC     HL
        LD      [HL],A

        LD      HL,SECTOR_BUFFER ;[HL] has location of FAT data, write it
        CALL    BIOS_WRITE_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      A,0BH                   ;Next sector B with F0,FF,FF for second FAT
        CALL    BIOS_SET_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      HL,SECTOR_BUFFER ;[HL] has location of 2nd FAT data, write it
        CALL    BIOS_WRITE_SECTOR;Same as first above
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      A,1H                    ;Setup the first (Boot) sector
        CALL    BIOS_SET_SECTOR         ;MSDOS/Windows expects key bytes in the first 28 bytes
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      DE,SECTOR_BUFFER ;Next move the special 28 bytes into the area
        LD      HL,MSDOS_14BOOT_DATA    ;Note because the above sectors only used 3 bytes the
        LD      BC,28                   ;rest of the sector is still full of 0's
        LDIR
```

```
        LD      HL,SECTOR_BUFFER ;[HL] has location of 2nd FAT data, write it
        CALL    BIOS_WRITE_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP                 ;We are done here
        LD      HL,MSDOS14_MOD_MSG
        CALL    PMSG
        JP      LOOP

MSDOS12:                        ;Initilize sectors on 5" 1.2M MSDOS/Windows disk
        XOR     A,A                     ;Fill in first FAT area
        CALL    BIOS_SET_TRACK
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      A,2                     ;First FAT at sector 2 with F9,FF,FF
        CALL    BIOS_SET_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        CALL    ZERO_BUFFER             ;Fill sector area with 0's

        LD      A,0F9H                  ;<--Note
        LD      HL,SECTOR_BUFFER ;[HL] will have location of sector data
        LD      [HL],A
        INC     HL
        LD      A,0FFH
        LD      [HL],A
        INC     HL
        LD      [HL],A

        LD      HL,SECTOR_BUFFER ;[HL] has location of FAT data, write it
        CALL    BIOS_WRITE_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      A,09H                   ;Also sector 9 with F9,FF,FF for second FAT
        CALL    BIOS_SET_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      HL,SECTOR_BUFFER ;[HL] has location of 2nd FAT data, write it
        CALL    BIOS_WRITE_SECTOR;Same as first above
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      A,1H                    ;Setup the first (Boot) sector
        CALL    BIOS_SET_SECTOR         ;MSDOS/Windows expects key bytes in the first 28 bytes
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      DE,SECTOR_BUFFER ;Next move the special 28 bytes into the area
        LD      HL,MSDOS_12BOOT_DATA    ;Note because the above sectors only used 3 bytes the
        LD      BC,28                   ;rest of the sector is still full of 0's
        LDIR
```

```
        LD      HL,SECTOR_BUFFER ;[HL] has location of 2nd FAT data, write it
        CALL    BIOS_WRITE_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP                 ;We are done here
        LD      HL,MSDOS12_MOD_MSG
        CALL    PMSG
        JP      LOOP

MSDOS360:                               ;Initilize sectors on 5" 360K (9 sec/track) MSDOS/Windows disk
        XOR     A,A                     ;Fill in first FAT area
        CALL    BIOS_SET_TRACK
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      A,2                     ;First FAT at sector 2 with FD,FF,FF
        CALL    BIOS_SET_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        CALL    ZERO_BUFFER             ;Fill sector area with 0's

        LD      A,0FDH                  ;<--Note
        LD      HL,SECTOR_BUFFER ;[HL] will have location of sector data
        LD      [HL],A
        INC     HL
        LD      A,0FFH
        LD      [HL],A
        INC     HL
        LD      [HL],A

        LD      HL,SECTOR_BUFFER ;[HL] has location of FAT data, write it
        CALL    BIOS_WRITE_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      A,04H                   ;Also sector 4 with FD,FF,FF for second FAT
        CALL    BIOS_SET_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      HL,SECTOR_BUFFER ;[HL] has location of 2nd FAT data, write it
        CALL    BIOS_WRITE_SECTOR;Same as first above
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      A,1H                    ;Setup the first (Boot) sector
        CALL    BIOS_SET_SECTOR         ;MSDOS/Windows expects key bytes in the first 28 bytes
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      DE,SECTOR_BUFFER ;Next move the special 28 bytes into the area
        LD      HL,MSDOS_360BOOT_DATA   ;Note because the above sectors only used 3 bytes the
        LD      BC,28                   ;rest of the sector is still full of 0's
        LDIR

        LD      HL,SECTOR_BUFFER ;[HL] has location of 2nd FAT data, write it
```

```
        CALL    BIOS_WRITE_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP                 ;We are done here
        LD      HL,MSDOS360_MOD_MSG
        CALL    PMSG
        JP      LOOP

MSDOS360B:                              ;Initilize sectors on 5" 360K (8 sec/track) MSDOS/Windows disk
        XOR     A,A                     ;Fill in first FAT area
        CALL    BIOS_SET_TRACK          ;<<< NOTE I HAVE NOT TESTED THIS FORMAT WITH DOS 1.1 >>>
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      A,2                     ;First FAT at sector 2 with FD,FF,FF
        CALL    BIOS_SET_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        CALL    ZERO_BUFFER             ;Fill sector area with 0's

        LD      A,0FFH                  ;<--Note
        LD      HL,SECTOR_BUFFER ;[HL] will have location of sector data
        LD      [HL],A
        INC     HL
        LD      A,0FFH
        LD      [HL],A
        INC     HL
        LD      [HL],A

        LD      HL,SECTOR_BUFFER ;[HL] has location of FAT data, write it
        CALL    BIOS_WRITE_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      A,04H                   ;Also sector 4 with FD,FF,FF for second FAT
        CALL    BIOS_SET_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      HL,SECTOR_BUFFER ;[HL] has location of 2nd FAT data, write it
        CALL    BIOS_WRITE_SECTOR;Same as first above
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      A,1H                    ;Setup the first (Boot) sector
        CALL    BIOS_SET_SECTOR         ;MSDOS/Windows expects key bytes in the first 28 bytes
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP

        LD      DE,SECTOR_BUFFER ;Next move the special 28 bytes into the area
        LD      HL,MSDOS_360BBOOT_DATA  ;Note because the above sectors only used 3 bytes the
        LD      BC,28                   ;rest of the sector is still full of 0's
        LDIR

        LD      HL,SECTOR_BUFFER ;[HL] has location of 2nd FAT data, write it
        CALL    BIOS_WRITE_SECTOR
```

```
        CALL    NZ,SHOW_ERRORS
        JP      NZ,LOOP                 ;We are done here
        LD      HL,MSDOS360B_MOD_MSG
        CALL    PMSG
        JP      LOOP


ZERO_BUFFER:                            ;Fill sector buffer area with 0's
        LD      HL,SECTOR_BUFFER ;[HL] will have location of sector data
        LD      BC,512
FILL_MORE:                              ;Fill area with 0's first
        LD      A,0
        LD      [HL],A
        INC     HL
        DEC     BC
        LD      A,C
        OR      A,B
        JR      NZ,FILL_MORE
        RET



;--------------- Get WD2793 Sector register value ---------------------------------

GET_WD_SECTOR:
        LD      HL,GET_WD_SEC_MSG;WD2798 Sector Register =
        CALL    PMSG
        CALL    BIOS_GET_WD_SECTOR      ;Return with Sector in [D]
        JP      Z,GET_SECTOR1
        CALL    SHOW_ERRORS
        JP      LOOP
GET_SECTOR1:
        LD      A,D
        LD      (@SEC),A        ;Store Sector at @SEC
        LD      A,D
        CALL    PACC
        LD      HL,H_MSG        ;H.
        CALL    PMSG
        JP      LOOP
submit

;--------------- Get WD2793 Track register value ---------------------------------

GET_WD_TRACK:
        LD      HL,GET_WD_TRK_MSG;WD2798 Track Register =
        CALL    PMSG
        CALL    BIOS_GET_WD_TRACK;Return with Track in [D]
        JP      Z,GET_TRACK1
        CALL    SHOW_ERRORS
        JP      LOOP
GET_TRACK1:
        LD      A,D
        LD      (@TRK),A        ;Store track at @TRK
        LD      A,D
        CALL    PACC
        LD      HL,H_END_MSG            ;H.
```

```
        CALL    PMSG
        JP      LOOP


;--------------- Print string explaining last error code # sent from ZFDC board ---------------------------------

GET_ERROR_STRING
        LD      HL,GET_ERROR_NUMBER_MSG   ;"Enter Error Number"
        CALL    PMSG

        CALL    GETHEX                    ;Get Hex number
        PUSH    AF                        ;Save it
        LD      HL,H_END_MSG
        CALL    PMSG
        LD      HL,RETURN_NUMBER_MSG      ;"String Returned = "
        CALL    PMSG
        POP     AF                        ;Get it back

        CALL    BIOS_ERROR_STRING;Print String on CRT
        JP      LOOP

;--------------- Get WD2793 Status register value --------------------------------

GET_WD_STATUS:
        LD      HL,GET_WD_SEC_MSG;WD2798 Status Register =
        CALL    PMSG
        CALL    BIOS_GET_WD_SECTOR        ;Return with Sector in [D]
        JP      NZ,STATUS_ERR
        LD      A,D
        CALL    PACC

        LD      HL,GET_WD_TRK_MSG;WD2798 Status Register =
        CALL    PMSG
        CALL    BIOS_GET_WD_TRACK;Return with Track in [D]
        JP      NZ,STATUS_ERR
        LD      A,D
        CALL    PACC

        LD      HL,GET_WD_STAT_MSG        ;WD2798 Status Register =
        CALL    PMSG
        CALL    BIOS_GET_WD_STATUS        ;Return with Status in [D]
        JP      NZ,STATUS_ERR
        LD      A,D
        CALL    BITS
        JP      LOOP

STATUS_ERR:
        CALL    SHOW_ERRORS
        JP      LOOP


;--------- Set ZFDC Board so it displays commands recieved and ACK' sent
;           back in the TRACK & SECTOR Displays

DEBUG_MODE
        LD      A,(DEBUG_FLAG)
```

```
        OR      A,A
        JR      Z,DEBUG_ON
        XOR     A,A
        LD      (DEBUG_FLAG),A          ;Set to Z to turn off
        LD      HL,DEBUG_OFF_MSG
        CALL    PMSG
        CALL    BIOS_DEBUG_OFF
        CALL    NZ,SHOW_ERRORS
        JP      LOOP
DEBUG_ON:
        DEC     A
        LD      (DEBUG_FLAG),A          ;Set to 0FFZ to turn on
        LD      HL,DEBUG_ON_MSG
        CALL    PMSG
        CALL    BIOS_DEBUG_ON
        CALL    NZ,SHOW_ERRORS
        JP      LOOP

;-------- Set ZFDC Board so it displays TRACK & SECTOR Information.

DETAIL_SEC_DATA:
        LD      A,(DISPLAY_FLAG)
        OR      A,A
        JR      Z,DISPLAY_OFF
        XOR     A,A
        LD      (DISPLAY_FLAG),A
        LD      HL,DISPLAY_ON_MSG
        CALL    PMSG
        JP      LOOP
DISPLAY_OFF:
        DEC     A
        LD      (DISPLAY_FLAG),A
        LD      HL,DISPLAY_OFF_MSG
        CALL    PMSG
        JP      LOOP



;Dump ZFDC Board memory variables to S-100 system of most RAM variables and flag values (starting at 8000H)

RAM_DUMP:                               ;CMD = 19H
        LD      HL,RAM_DUMP_A_MSG
        CALL    PMSG
        CALL    CRLF

        LD      C,CMD_RAM_DUMP          ;Send RAM dump command to ZFDC board
        CALL    S100OUT
        CALL    WAIT_FOR_ACK            ;Wait for a NO_ERRORS flag to come back
        JP      NZ,LOOP

        LD      HL,DRIVE_A_PORT_MSG     ;Do Drive A:
        CALL    PMSG
        CALL    GET_DRIVE_DUMP          ;Will return with ESC if we want to abort
        CP      A,ESC
        JP      Z,STOP_DRIVE_DUMP
        LD      C,NO_ERRORS_FLAG ;Else sent a NO_ERRORS Flag
```

```
        CALL    S100OUT
        CALL    CRLF

        LD      HL,DRIVE_B_PORT_MSG       ;Do Drive B:
        CALL    PMSG
        CALL    GET_DRIVE_DUMP
        CP      A,ESC
        JP      Z,STOP_DRIVE_DUMP
        LD      C,NO_ERRORS_FLAG ;Else send a NO_ERRORS Flag
        CALL    S100OUT
        CALL    CRLF

        LD      HL,DRIVE_C_PORT_MSG       ;Do Drive C:
        CALL    PMSG
        CALL    GET_DRIVE_DUMP
        CP      A,ESC
        LD      C,NO_ERRORS_FLAG ;Else send a NO_ERRORS Flag
        CALL    S100OUT
        JP      Z,STOP_DRIVE_DUMP
        CALL    CRLF

        LD      HL,DRIVE_D_PORT_MSG       ;Do Drive D:
        CALL    PMSG
        CALL    GET_DRIVE_DUMP
        CP      A,ESC
        JP      Z,STOP_DRIVE_DUMP
        LD      C,NO_ERRORS_FLAG ;Else send a NO_ERRORS Flag
        CALL    S100OUT
        CALL    CRLF

        LD      HL,Z80_SP_MSG             ;Next display the stack pointer
        CALL    PMSG
        CALL    S100IN
        CALL    PACC

        CALL    S100IN
        CALL    PACC

        LD      HL,CURRENT_DR_MSG;Next display @CURRENT_DRIVE_SELECT
        CALL    PMSG
        CALL    S100IN
        CALL    PACC

        CALL    WAIT_FOR_ACK              ;Ignore errors (if any)
        CALL    CRLF
        JP      LOOP


STOP_DRIVE_DUMP:
        LD      C,CMD_ABORT               ;Send RAM dump command to ZFDC board
        CALL    S100OUT
        JP      LOOP

GET_DRIVE_DUMP:
        LD      HL,HW_PORT_MSG
        CALL    PMSG
```

```
CALL    S100IN                  ;Note potential to lockup here (but unlightly)
LD      (DR_STORE),A            ;Store bits in for later below
CALL    BITS                    ;Show port bit pattern

LD      HL,NSCTRS_MSG           ;Sectors/Track for disk
CALL    PMSG
CALL    S100IN
LD      D,A                     ;<---- Store sector/track for later (0 if an unformatted disk)
DEC     A                       ;Because table data is sec/track+1
CALL    PACC

LD      HL,NTRKS_MSG            ;Tracks/Side
CALL    PMSG
CALL    S100IN
CALL    PACC

LD      HL,HEADR_MSG            ;For Formatting
CALL    PMSG
CALL    S100IN
CALL    PACC

LD      HL,GAP1_MSG             ;       "
CALL    PMSG
CALL    S100IN
CALL    PACC

LD      HL,GAP2_MSG             ;       "
CALL    PMSG
CALL    S100IN
CALL    PACC

LD      HL,GAP3_MSG             ;       "
CALL    PMSG
CALL    S100IN
CALL    PACC

LD      HL,GAP4_MSG             ;       "
CALL    PMSG
CALL    S100IN
CALL    PACC

LD      HL,GAP4R_MSG            ;       "
CALL    PMSG
CALL    S100IN
CALL    PACC

LD      HL,SEC_SIZE_FLAG_MSG    ;0=128 Byte sectors, 1 = 256, 2 = 512, 4=1024 Byte sectors
CALL    PMSG
CALL    S100IN
CALL    PACC

LD      HL,GAP_FILL_CHAR_MSG    ;Byte used in disk formating
CALL    PMSG
CALL    S100IN
CALL    PACC
```

```
LD      HL,DATA_FILL_CHAR_MSG     ;     "       "       "
CALL    PMSG
CALL    S100IN
CALL    PACC

LD      HL,SPECIAL_FLAG_MSG       ;Flag byte for cases where after formatting disk need to be initilized. Normally 0, CPM86_FLAG = 1
CALL    PMSG
CALL    S100IN
CALL    PACC

LD      HL,SKEW_MSG               ;High address of sector skew table
CALL    PMSG
CALL    S100IN
CALL    PACC

CALL    S100IN                    ;Low address
CALL    PACC

LD      HL,FORMAT_NUM_MSG;Each format will have a unique number in the table list below.
CALL    PMSG
CALL    S100IN
CALL    PACC

LD      HL,SYS_TRKS_MSG           ;How many tracks for system (usually 2 for 8-inch disks)
CALL    PMSG
CALL    S100IN
CALL    PACC

LD      HL,SEC_SIZE_BYTES_MSG     ;Two Bytes. (128,256,512 or 1024)
CALL    PMSG
CALL    S100IN                    ;High address
CALL    PACC

CALL    S100IN                    ;Low Address
CALL    PACC

LD      HL,TRACK_SIZE_MSG;Two Bytes. Track size (in bytes) of that disks format
CALL    PMSG
CALL    S100IN                    ;High address
CALL    PACC

CALL    S100IN                    ;Low Address
CALL    PACC

LD      HL,CPM_TRACK_MSG ;CPM requested Track for this disk
CALL    PMSG
CALL    S100IN
CALL    PACC

LD      HL,CPM_SECTOR_MSG;CPM requested Sector for this disk
CALL    PMSG
CALL    S100IN
LD      E,A                       ;store sector data for later (0 if not used by CPM)
CALL    PACC

LD      A,D                       ;Check if the drive has an assigned format table (Sec = 0)
```

```
        OR      A,A                     ;If zero then no format yet assigned
        JR      NZ,SHOW_DRIVE_INFO
        LD      HL,NO_SIDES_MSG
        CALL    PMSG
        LD      HL,NO_FORMAT_MSG ;No format for this disk so far
        CALL    PMSG
        JR      DONE_PORT

SHOW_DRIVE_INFO:
        LD      A,(DR_STORE)            ;Get Drive paramaters from above
        BIT     SIDE_BIT,A
        JR      Z,BS_BIT        ;Z for Side B
        LD      HL,AS_BIT_MSG
        CALL    PMSG
        JR      LOOK_AT_BITS
BS_BIT: LD      HL,BS_BIT_MSG
        CALL    PMSG

LOOK_AT_BITS:
        LD      A,(DR_STORE)            ;Get Drive paramaters from above
        BIT     DENSITY_BIT,A           ;Now analyze the high nibble bits (SD or DD)
        JR      Z,DD_BIT
        LD      HL,SD_BIT_MSG
        CALL    PMSG
        JR      RAM_CHECK_SIZE
DD_BIT: LD      HL,DD_BIT_MSG
        CALL    PMSG

RAM_CHECK_SIZE:
        LD      A,(DR_STORE)            ;Get Drive paramaters from above
        BIT     SIZE_BIT,A              ;5" or 8"
        JR      Z,S5_BIT
        LD      HL,S8_BIT_MSG
        CALL    PMSG
        JR      RAM_CHECK_SIDE
S5_BIT: LD      HL,S5_BIT_MSG
        CALL    PMSG

RAM_CHECK_SIDE:
        CALL    S100IN                  ;Get SS/DS flag for this disk

        BIT     SIDE_BIT,A              ;Is it SS or DS disk
        JR      Z,CPM_SS        ;00000100 for Single sided disk
        LD      HL,CPM_DS_MSG
        CALL    PMSG
        JR      DONE_PORT
CPM_SS: LD      HL,CPM_SS_MSG
        CALL    PMSG

DONE_PORT:
        LD      HL,NEXT_DISK_MSG
        CALL    PMSG
        CALL    GETCMD                  ;Check if more info is required
        RET
```

```
;-------------------------------------------------
TBD:             NOP
                 NOP              ;fall into NOT_DONE

NOT_DONE:
         LD      HL,NOT_DONE_MSG
         CALL    PMSG
         JP      LOOP



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


;Display the Sector ID read from disk to RAM location at (TRACK_ID_BUFFERR)

DISPLAY_TRACK_ID                  ;Display the 6 Bytes on the CRT
         LD      HL,TRK_ID_MSG
         CALL    PMSG
         LD      B,6
         LD      HL,TRACK_ID_BUFFER
SHOW1:   LD      A,(HL)
         CALL    PACC
         INC     HL
         DJNZ    SHOW1
         RET


;Display the Sector read from disk to RAM location at (SECTOR_BUFFER)
DISPLAY_TRACK:
         PUSH    AF
         PUSH    BC
         PUSH    DE
         PUSH    HL
DISP:    CALL    CRLF
         LD      HL,SECTOR_BUFFER ;GET PARAMETERS IN [HL],[DE]
         LD      DE,SECTOR_BUFFER + MAX_TRACK_SIZE
         LD      A,L                       ;ROUND OFF ADDRESSES TO XX00H
         AND     0F0H
         LD      L,A
         LD      A,E                       ;FINAL ADDRESS LOWER HALF
         LD      A,16                      ;Display line count
         LD      (LINE_COUNT),A
         AND     0F0H
         ADD     A,10H                     ;FINISH TO END 0F LINE
TSF172:  CALL    LFAD                      ;CRLF, Print [HL] value
TSF175:  LD      C,SPACE                   ;One space across
         CALL    CO
         LD      A,(HL)
         CALL    PACC
         CALL    HILOX
         LD      A,L
```

```
        AND     0FH
        JR      NZ,TSF175
        LD      C,TAB                   ;INSERT A TAB BETWEEN DATA
        CALL    CO
        LD      B,4H                    ;ALSO 4 SPACES
TTA11:  LD      C,SPACE
        CALL    CO
        DJNZ    TTA11
        LD      B,16                    ;NOW PRINT ASCII (16 CHARACTERS)
        PUSH    DE                      ;TEMPORLY SAVE [DE]
        LD      DE,0010H
        SBC     HL,DE
        POP     DE
TT11:   LD      A,(HL)
        AND     7FH
        CP      ' '                     ;FILTER OUT CONTROL CHARACTERS'
        JR      NC,TT33
TT22:   LD      A,'.'
TT33:   CP      07CH
        JR      NC,TT22
        LD      C,A                     ;SET UP TO SEND
        CALL    CO
        INC     HL
        DJNZ    TT11                    ;REPEAT FOR WHOLE LINE
        LD      A,(LINE_COUNT)
        DEC     A
        LD      (LINE_COUNT),A          ;Check if we have done a page of lines
        JR      NZ,TSF172
        LD      A,16
        LD      (LINE_COUNT),A
        CALL    CRLF
        PUSH    HL
        LD      HL,HIT_SP_MSG           ;"Hit ESC to Abort Msg"
        CALL    PMSG
        CALL    CI
        CP      A,ESC
        POP     HL
        JR      Z,TRK_DONE
        CALL    CRLF
        JR      TSF172
;
TRK_DONE:
        POP     HL                      ;Abort display
        POP     DE
        POP     BC
        POP     AF
        RET

;Display the Sector read from disk to RAM location at (SECTOR_BUFFER)
DISPLAY_SECTOR:
        PUSH    AF
        PUSH    BC
        PUSH    DE
        PUSH    HL

        PUSH    HL
```

```
        CALL    SHOW_TSS_LOC            ;Track = xxH  Sector = xxH, Side=A/B
        POP     HL                      ;Move DMA address into HL

        LD      A,(DISPLAY_FLAG) ;Is Detailed sector data display ON/OFF
        OR      A,A
        JP      NZ,SEC_DONE

        PUSH    HL
        LD      E,(IX+SEC_SIZE_BYTES)   ;128,256,512 or 1024 byte sector size
        LD      D,(IX+SEC_SIZE_BYTES+1)
        ADD     HL,DE
        EX      DE,HL                   ;[DE] is now [HL]+ sector size
        POP     HL
        JR      SDISP1

SDISP:  CALL    CRLF
SDISP1: LD      A,L                     ;ROUND OFF ADDRESSES TO XX00H
        AND     0F0H
        LD      L,A
        LD      A,16                    ;Display line count
        LD      (LINE_COUNT),A
        LD      A,E                     ;FINAL ADDRESS LOWER HALF
        AND     0F0H
        ADD     A,10H                   ;FINISH TO END 0F LINE
SF172:  CALL    LFAD                    ;CRLF, Print [HL] value
SF175:  LD      C,SPACE                 ;One space across
        CALL    CO
        LD      A,(HL)
        CALL    PACC
        CALL    HILOX                   ;If [HL]=[DE] we pop stack and return
        LD      A,L
        AND     0FH
        JR      NZ,SF175
        LD      C,TAB                   ;INSERT A TAB BETWEEN DATA
        CALL    CO
        LD      B,4H                    ;ALSO 4 SPACES
STA11:  LD      C,SPACE
        CALL    CO
        DJNZ    STA11
        LD      B,16                    ;NOW PRINT ASCII (16 CHARACTERS)
        PUSH    DE                      ;TEMPORLY SAVE [DE]
        LD      DE,0010H
        SBC     HL,DE
        POP     DE
ST11:   LD      A,(HL)
        AND     7FH
        CP      ' '                     ;FILTER OUT CONTROL CHARACTERS'
        JR      NC,ST33
ST22:   LD      A,'.'
ST33:   CP      07CH
        JR      NC,ST22
        LD      C,A                     ;SET UP TO SEND
        CALL    CO
        INC     HL
        DJNZ    ST11                    ;REPEAT FOR WHOLE LINE
        LD      A,(LINE_COUNT)
```

```
        DEC     A
        LD      (LINE_COUNT),A          ;Check if we have done a page of lines
        JR      NZ,SF172
        LD      A,16
        LD      (LINE_COUNT),A
        CALL    CRLF
        LD      A,(HALT_CHECK_FLAG)     ;Do we have a keyboard check for CRT display
        OR      A,A
        JR      Z,SF172
        PUSH    HL
        LD      HL,HIT_SP_MSG           ;"Hit ESC to Abort Msg"
        CALL    PMSG
        CALL    CI
        CP      A,ESC
        POP     HL
        JR      Z,SEC_DONE
        CALL    CRLF
        JR      SF172
SEC_DONE:
        POP     HL
        POP     DE
        POP     BC
        POP     AF
        RET


;
;PRINT [HL] AND A SPACE
LFAD:   CALL    CRLF
        PUSH    HL
        PUSH    BC
        CALL    LADR
        LD      C,SPACE
        CALL    CO
        POP     BC
        POP     HL
        RET

;PRINT [HL] ON CONSOL
LADR:   LD      A,H
        CALL    PACC
        LD      A,L
        CALL    PACC
        RET


;RANGE TEST ROUTINE CARRY SET = RANGE EXCEEDED (Used by Sector and Track display, only)
HILOX:  CALL    HILO
        RET     NC
;--------
        LD      B,10
        LD      C,BS                    ;Cleanup bug where one extra data byte is printed!
SCLEAN: CALL    CO                      ;Need it initially, to get ascii on last line printed!
        DJNZ    SCLEAN
        LD      B,10
        LD      C,' '                   ;Then print 8 spaces
```

```
SCLEAN1:CALL     CO
        DJNZ     SCLEAN1
        LD       C,CR                    ;and finel CR
        CALL     CO
;-------
        POP      HL                      ;Balance up Stack
        POP      HL                      ;All done
        POP      DE
        POP      BC
        POP      AF
        RET
HILO:   INC      HL                      ;RANGE CHECK SET CARRY IF [DE]=[HL]
        LD       A,H
        OR       L
        SCF
        RET      Z
        LD       A,E
        SUB      L
        LD       A,D
        SBC      A,H
        RET



WAIT_FOR_ACK:
        PUSH     BC
        PUSH     DE
        LD       BC,0
        LD       E,STATUS_DELAY          ;Timeout, (about 2 seconds)
WAIT_1: 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 input mode
        JR       Z,WAIT_2         ;if low then ZFDC is still in input mode
        CALL     S100STAT         ;Wait until ZFDC Board sends something
        JP       NZ,GET_HAND_SHAKE;Return Z flag set if OK, Status in [E]
WAIT_2: DJNZ     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
        DEC      A
        POP      DE                      ;Balance up stack
        POP      BC
        RET                              ;Return NZ flag set if timeout AND 0FFH in [A]

GET_HAND_SHAKE:
        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
```

```
WAIT_FOR_JOB_DONE:                      ;This is a special wait (a long time) for things like disk formatting
        PUSH    BC                      ;the operator can abort the job being done by typing ESC key
        PUSH    DE
        LD      B,0
WAIT_3: DJNZ    WAIT_3                  ;Let hardware settle down from possible previous command
WAIT_4: 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 input mode
        JR      Z,WAIT_4         ;if low then ZFDC is in input mode
        CALL    S100STAT         ;Wait until ZFDC Board is ready
        JP      NZ,GET_HAND_SHAKE;NZ, something there!
        CALL    CONSTAT                 ;Is there an ESC at the consol
        JR      Z,WAIT_4
        CALL    CI
        CP      A,ESC
        JR      NZ,WAIT_4
        POP     DE                      ;Balance up stack
        POP     BC
        XOR     A,A
        DEC     A
        LD      A,ABORT_FLAG            ;Return 0FEH, This error message flags operator terminated command
        RET                             ;Return NZ flag as well

;----------- Generalized Error string display routine. The Error # is in [A] -------------------------

SHOW_ERRORS:                            ;Display error information
        PUSH    AF                      ;Need to retain Z flag info
        CP      A,0FFH                  ;WAIT_FOR_ACK Timeout, (about 2 seconds) recieved
        JR      Z,TIMEOUT_ERR
        CP      A,CMD_RANGE_ERR
        JR      NC,RANGE_ERROR
        LD      HL,ERROR_TBL
        ADD     A,A                     ;X2
        ADD     A,L
        LD      L,A
        LD      A,(HL)
        INC     HL
        LD      H,(HL)
        LD      L,A                     ;[HL] Now contains pointer to error string
        CALL    PMSG                    ;Print Error message string
        CALL    CRLF
        POP     AF
        RET

TIMEOUT_ERR:
        LD      HL,TIMEOUT_ERROR_MSG
        CALL    PMSG
        POP     AF
        RET

RANGE_ERROR:                            ;If out or range Error message number
        PUSH    AF
        LD      HL,RANGE_ERROR_MSG
        CALL    PMSG
        POP     AF
        CALL    PACC
        LD      HL,H_MSG
```

```
        CALL    PMSG
        CALL    CRLF
        POP     AF
        RET


SHOW_DRIVE_NAME:
        PUSH    IX                      ;Save origional pointer
SHOW_DR1:LD     A,(IX+TITLE)
        OR      A,A                     ;Repeat up to end of string
        JP      Z,SHOW_DR2
        LD      C,A
        CALL    CO
        INC     IX
        JP      SHOW_DR1
SHOW_DR2:
        POP     IX
        RET


; General routine to show the current location of the active disk head.
; Will display track, sector and side if two sided drive/disk
; Format is 'At TRACK xxH, SCETOR yyH, SIDE A/B.'
; Note: NO CR/LF at start of string
;
SHOW_TSS_LOC:
        PUSH    HL
        LD      HL,AT_TRK               ;Announce current track position
        CALL    PMSG
        LD      A,(@TRK)
        CALL    PACC
        LD      HL,AT_SEC               ;and Sector position
        CALL    PMSG
        LD      A,(@SEC)
        CALL    PACC
SHOW_SIDES:                             ;Announce head if 2 sided disk
        BIT     SIDE_BIT,(IX+HW_BYTE)   ;Is it a 1 or 2 sided disk
        JR      Z,RW_SKIP_SIDES         ;If 0 is a ss disk so skip side display
        LD      A,(@SIDE)               ;Get hardware selection
        OR      A,A                     ;Port Hardware for side selection (0=A, 1=B)
        JR      NZ,RW_B_SIDE
        LD      HL,HEAD0_MSG            ;Announce side 0 (or A)
        CALL    PMSG
        POP     HL
        RET
RW_B_SIDE:
        LD      HL,HEAD1_MSG            ;Announce side 1 (or B)
        CALL    PMSG
        POP     HL
        RET
RW_SKIP_SIDES:
        LD      HL,H_MSG        ;If SS disk, no need for a HEAD# message
        CALL    PMSG
        POP     HL
        RET
```

```
;Main Routing to get a command character from Keyboard. Character returned in [A]

GETCMD:  CALL     CI                      ;GET A CHARACTER, convert to UC, ECHO it
         CALL     UCASE
         CP       A,ESC
         RET      Z                       ;Don't echo an ESC
         PUSH     AF                      ;Save it
         PUSH     BC
         LD       C,A
         CALL     CO                      ;Echo it
         POP      BC
         POP      AF                      ;Get it back. Return with Character in [A]
         RET

;                                 ;Convert LC to UC
UCASE:   CP       A,'a'           ;must be >= lowercase a
         RET      C               ; else go back...
         CP       A,'z'+1         ;must be <= lowercase z
         RET      NC              ; else go back...
         SUB      A,'a'-'A'       ;subtract lowercase bias
         RET

; ASCII TO BINARY CONVERSION ROUTINE
ASBIN:   SUB      30H
         CP       0AH
         RET      M
         SUB      07H
         RET


; Return with 2 HEX digits in [A]. If abort, Carry flag set + ESC in [A]

GETHEX:
         PUSH     BC
         CALL     GETCMD                  ;Get a character from keyboard & ECHO
         CP       A,ESC
         JR       Z,HEX_ABORT
         CP       '/'                     ;check 0-9, A-F
         JR       C,HEX_ABORT
         CP       'F'+1
         JR       NC,HEX_ABORT
         CALL     ASBIN                   ;Convert to binary
         SLA      A
         SLA      A
         SLA      A
         SLA      A                       ;Shift to high nibble
         LD       B,A                     ;Store it
         CALL     GETCMD                  ;Get 2nd character from keyboard & ECHO
         CP       A,ESC
         JR       Z,HEX_ABORT
         CP       '/'                     ;check 0-9, A-F
         JR       C,HEX_ABORT
         CP       'F'+1
         JR       NC,HEX_ABORT
         CALL     ASBIN                   ;Convert to binary
```

```
        OR      A,B                     ;add in the first digit
        OR      A,A                     ;To return NC
        POP     BC
        RET
HEX_ABORT:
        SCF                             ;Set Carry flag
        LD      A,ESC
        POP     BC
        RET



;Print the acccumulator value on CRT in HEX-ASCII. Return [A],[BC] unchanged

PACC:   PUSH    AF
        PUSH    BC
        PUSH    AF
        RRCA
        RRCA
        RRCA
        RRCA
        CALL    ZCONV
        POP     AF
        CALL    ZCONV
        POP     BC
        POP     AF
        RET

ZCONV:  AND     A,0FH                   ;HEX to ASCII
        ADD     90H
        DAA
        ADC     40H
        DAA
        LD      C,A
        CALL    CO
        RET



;DISPLAY BIT PATTERN IN [A]
;
BITS:   PUSH    AF
        PUSH    BC
        PUSH    DE
        LD      E,A
        LD      B,8
BQ2:    SLA     E
        LD      A,18H
        ADC     A
        LD      C,A
        CALL    CO
        DJNZ    BQ2
        POP     DE
```

```
        POP     BC
        POP     AF
        RET

CRLF:   PUSH    AF                      ;Send CR/LF to CRT Return [A] & [BC] Unchanged
        PUSH    BC
        LD      C,CR
        CALL    CO
        LD      C,LF
        CALL    CO
        POP     BC
        POP     AF
        RET


PMSG:   PUSH    BC                      ;Save [BC]
PMSG2:  LD      A,(HL)                  ;PRINT MESSAGE STRING in [HL] up to 0
        OR      A
        JP      Z,PMSG1
        LD      C,A
        CALL    CO
        INC     HL
        JP      PMSG2
PMSG1:  POP     BC
        RET


; CHECK IF AN ABORT (ESC) CHARACTER IS PRESENT AT THE CONSOL
; HOLD EVERYTHING IF SPACEBAR IS PRESSED. Return NZ if ESC
;
CHECKABORT:                     ;see if an abort is required
        CALL    CONSTAT
        RET     Z
CHECK1: CALL    CI
        CP      A,ESC           ;ESC to abort
        JR      Z,CABORT
        CP      A,' '           ;If spacebar then freeze CRT display
        JR      Z,CHECK1
        XOR     A,A
        RET
CABORT: XOR     A,A             ;Was ESC, so return with NZ flag
        DEC     A               ;return NZ, & 0FFH in A if ESC there
        RET
;

; Simple random number generator. Return number in [A]
RANDOM: LD      A,R             ;Seed will be differentd each time
        LD      B,A
        ADD     A,A
        ADD     A,A
        ADD     A,B
        RR      A
        ADD     A,7
        RET

;Get a random Track & Sector number for current track/drive table put in @trk & @sec
```

```
RANDOM_TRK_SEC:
        CALL    RANDOM                  ;Get next random number for Track AND Sector
        BIT     SIZE_BIT,(IX+HW_BYTE)
        JR      NZ,RANDOM1
        AND     1FH                     ;Tracks max for 5"
RANDOM1:AND     3FH                     ;77 Tracks on an 8" drive
        LD      B,(IX+NTRKS)            ;Trim to size
        CP      A,B
        JR      NC,RANDOM_TRK_SEC
        LD      (@TRK),A        ;Store new track number
RAND_SEC:
        CALL    RANDOM
        AND     1FH                     ;Will never be more than this
        LD      B,(IX+NSCTRS)           ;Trim to size
        DEC     B                       ;Becaurse it's sectors+1
        CP      A,B
        JR      NC,RAND_SEC
        PUSH    AF
        AND     A,1H
        LD      (@SIDE),A               ;Close enough for a random side!
        POP     AF
        LD      (@SEC),A        ;Store new sector number
        OR      A,A
        RET     NZ
        LD      A,1
        LD      (@SEC),A
        RET



;Get next Track & Sector number for current track/drive table put in @trk & @sec
NEXT_TRK_SEC:
        LD      A,(@SEC)
        INC     A
        LD      (@SEC),A
        LD      B,(IX+NSCTRS)           ;Trim to size
        CP      A,B                     ;Are we at the end
        RET     C
        LD      A,1                     ;Remember no sector 0
        LD      (@SEC),A
        BIT     SIDE_BIT,(IX+HW_BYTE)   ;Is is a double sided disk
        JR      Z,ONE_SIDE_ONLY
        LD      A,(@SIDE)
        OR      A,A                     ;If A side switch to B
        JR      NZ,BACK_TO_A
        LD      A,1
        LD      (@SIDE),A               ;Set to side B
        RET
BACK_TO_A:
        XOR     A,A
        LD      (@SIDE),A
ONE_SIDE_ONLY:
        LD      A,(@TRK)
        INC     A
        LD      (@TRK),A
        CP      A,(IX+NTRKS)            ;Are we at the end
```

```
        RET     C
        XOR     A,A                     ;Reset to track 0
        LD      (@TRK),A
        RET


; Software time delay ~0.5 Sec @ 6MHZ
DELAY_1S:
        PUSH    AF
        PUSH    BC
        LD      A,5H
        LD      BC,0                    ;Delay
WAIT_D: DJNZ    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
        POP     BC
        POP     AF
        RET


;-----------------------------------------------------------------------------------------------------

SIGNON:         DB      CR,LF
                DB      'ZFDC Board Diagnostic & Format Program. '
                DB      '(V2.8 by John Monahan 04/30/2011)',CR,LF,0

MAIN_MENU:      DB      CR,LF,LF
                DB      '00 FORMAT DISK    01 Set Drive Format 02 Select Drive    03 Set HOME',CR,LF
                DB      '04 Set TRACK      05 Set SECTOR       06 Set SIDE        07 Step IN Test',CR,LF
                DB      '08 Get TRK ID     09 Random TRK ID    0A Seek(NV)Test    0B Seek(V) to TRK',CR,LF
                DB      '0C Random Seek(V) 0D READ a Sector    0E Step RD Sector  0F Random RD Sec  ',CR,LF
                DB      '10 WRITE a Sector 11 Step Sec R/W     12 Random Sec R/W  13 Write Boot Sec',CR,LF
                DB      '14 Get Format No. 15 Read TRACK       16 Error To String 17 SEC display ON/OFF',CR,LF
                DB      '18 PIO Test       19 To Monitor       1A Show Signon     1B Reset ZFDC Board',CR,LF
                DB      '1C WD2793 Status  1D Data Dump        1E Debug ON/OFF    1F Multi-Sec RD',CR,LF
                DB      '20 Multi-Sect WR  ESC To CPM (any time)',CR,LF
                DB,LF,0

DISK_TBL_MSG:   DB      'Disk Format Table = ',0
DISK_TBL1_MSG:  DB      'New Disk Format Table = ',0
CMD_ERROR_MSG   DB      CR,LF,'Out of range command',CR,LF,0

MONITOR_MSG:    DB      CR,LF,'Sending MONITOR Command to ZFDC Board',CR,LF,0
SIGNON_MSG:     DB      CR,LF,'Sending SIGNON Command to FDC Board',CR,LF,0
RESET_MSG:      DB      CR,LF,'Sending RESET Command to FDC Board',CR,LF,0
TIMEOUT_MSG:    DB      CR,LF,BELL,'Timeout Error. ZFDC Board failed to send back an ACK handshake messege.',CR,LF,0
ERR_CODE_MSG:   DB      CR,LF,'Error Code = ',CR,LF,0
DRIVE_MSG:      DB      CR,LF,'Select ZFDC Drive # (0,1,2 or 3):- ',0
TABLE_MSG_8:    DB      CR,LF,LF
                DB      '01 = 8" (Also 5" & 3.5"), SDSS, 26 X 128 Byte Sectors, (IBM 3740 Format).',CR,LF
                DB      '02 = 8" (Also 5" & 3.5"), DDSS, 50 X 128 Byte Sectors, (SD_Systems Format).',CR,LF
                DB      '03 = 8" (Also 5" & 3.5"), DDDS, 26 X 256 Byte Sectors, (IBM System 34 Format).',CR,LF
                DB      '04 = 8" (Also 5" & 3.5"), DDDS, 15 X 512 Byte Sectors.',CR,LF
```

```
                DB      '05 = 8" (Also 5" & 3.5"), DDSS, 8 X 1024 Byte Sectors.',CR,LF
                DB      '06 = 8" (Also 5" & 3.5"), DDDS, 8 X 1024 Byte Sectors.',CR,LF,LF
                DB      '15 = 3.5" (Only), for MSDOS, 1.44M DDDS, 18 X 512 Byte Sec., 80 Tracks.',CR,LF
                DB      '16 = 3.5" (Only), for CPM, 1.44M DDDS, 18 X 512 Byte Sec., 80 Tracks.',CR,LF
                DB      '17 = 5" (Only), for MSDOS, 1.2M DDDS, 18 X 512 Byte Sec., 80 Tracks.',CR,LF
                DB      '18 = 5" (Only), for CPM, 1.2M DDDS, 18 X 512 Byte Sec., 80 Tracks.',0

TABLE_MSG_5:    DB      CR,LF,LF
                DB      '07 = 5" SDSS, 17 X 128 Byte Sectors, (SD-Systems Format).',CR,LF
                DB      '08 = 5" DDSS, 28 X 128 Byte Sectors, (SD-Systems Format).',CR,LF
                DB      '09 = 5" DDDS, 8 X 512 Byte Sectors, (IBM PC CPM-86 format).',CR,LF
                DB      '0A = 5" DDDS, 9 X 512 Byte Sectors, (DEC VT180 format).',CR,LF
                DB      '0B = 5" DDDS, 16 X 256 Byte Sectors, (TOSHIBA T-100 format).',CR,LF
                DB      '0C = 5" SDDS, 18 X 128 Byte Sectors, (CROMEMCO CDOS Format).',CR,LF
                DB      '0D = 5" DDDS, 10 X 512 Byte Sectors, (CROMEMCO CDOS/CPM Format).',CR,LF
                DB      '0E = 5" SDDS, 10 X 512 Byte Sectors,  (EPSON QX-10 Format).',CR,LF
                DB      '0F = 5" DDDS, 5 X 1024 Byte Sectors, (MORROW MD3 format).',CR,LF
                DB      '10 = 5" DDDS, 8 X 512 Byte Sectors, (ZENITH Z-100 format).',CR,LF
                DB      '11 = 5" DDDS, 10 X 512 Byte Sectors, (SUPERBRAIN QD format).',CR,LF
                DB      '12 = 5" DDDS, 8 X 512 Byte Sectors,  (IBMPC MSDOS 1.1 format).',CR,LF
                DB      '13 = 5" DDDS, 9 X 512 Byte Sectors, (IBMPC MSDOS 2.x format).',CR,LF
                DB      '14 = 5" DDSS, 10 X 512 Byte Sectors, (TRS-80 MOD-III format).',0

                DB      CR,LF,'Enter Drive Format Table menu number (xxH):- ',0
BAD_TABLE_MSG:  DB      CR,LF,'Invalid Format Table Number!',CR,LF,0
DISK_TABLE_MSG: DB      CR,LF,'Current Format Table Number = ',0
TRACK_MSG:      DB      CR,LF,'Enter Track Number (xxH):- ',0
SIDE_MSG:       DB      CR,LF,'Select A = Side A, B = Side B:- ',0
SEC_MSG: DB     CR,LF,'Enter Sector Number (xxH):- ',0
HOME_MSG:       DB      CR,LF,'Set current selected drive head to track 0',CR,LF,0
STEP_IN_MSG:    DB      CR,LF,'Sending Head Step IN Command to current drive (Repetedly)',CR,LF,0
STEP_OUT_MSG:   DB      CR,LF,'Sending Head Step OUT Command to current drive',CR,LF,0
CMD_SEEK_NV_MSG: DB     CR,LF,'Sending Seek Track Command (No Verify) to current drive (Repetedly)',CR,LF,0
CMD_SEEK_TRACK_MSG: DB  CR,LF,'Sending Seek to current track (With Verify) on current drive.',CR,LF,0
TRACK_ID_MSG:   DB      CR,LF,'Sending Get Track ID Command to current drive (Repetedly)',0
R_TRACK_ID_MSG: DB      CR,LF,'Sending RANDOM Get Track ID Command to current drive (Repetedly)',0
READ_SECTOR_MSG:DB      CR,LF,'Read Sector Command. (Will read SEC Buffer from Current sector on ZFDC).',0
WRITE_SECTOR_MSG: DB    CR,LF,'Write Sector Command. (Will write SEC Buffer to Current sector on ZFDC).',0

MULTI_RD_SECTOR_MSG: DB CR,LF,'Read Multi-Sector Command. (Will read multiple sectors from ZFDC).'
                DB      CR,LF,'Sector data will be read to loaction 5000H in RAM upwards.',CR,LF,0
MULTI_WR_SECTOR_MSG: DB CR,LF,'Write Multi-sector Command. (Will write multiple sectors to ZFDC).'
                DB      CR,LF,'Sector data will be written from loaction 5000H in RAM upwards.',CR,LF,0

NUM_OF_RD_SEC:  DB      CR,LF,'Enter number of sectors to read. (Note: No check for TPA overflow is done):-',0
RD_SEC_SIZE_MSG: DB     CR,LF,'Returned READ sector size = ',0
SEEK_TEST_MSG   DB      CR,LF,'Continous Seek Test.',CR,LF,0
H_END_MSG       DB      'H.',0
GET_WD_TRK_MSG  DB      CR,LF,'WD2798 Track Register = ',0
GET_WD_SEC_MSG  DB      CR,LF,'WD2798 Sector Register = ',0
GET_WD_STAT_MSG DB      CR,LF,'WD2798 Status Register = ',0

SEL_DRIVE_A_MSG: DB     CR,LF,'Selecting ZFDC Drive 0',0
SEL_DRIVE_B_MSG: DB     CR,LF,'Selecting ZFDC Drive 1',0
SEL_DRIVE_C_MSG: DB     CR,LF,'Selecting ZFDC Drive 2',0
SEL_DRIVE_D_MSG: DB     CR,LF,'Selecting ZFDC Drive 3',0
```

```
INVALID_DRIVE_MSG: DB     CR,LF,'Invalid Drive.',0
SEL_TABLE_MSG:    DB      CR,LF,'New Drive Table Number = ',0
                  DB      CR,LF,'The ZFDC Board is now selecting the new Disk Paramater Table.',CR,LF,0
GETTING_TABLE:    DB      CR,LF,'The ZFDC Board will return the current Disk Paramater Table number.',CR,LF,0
WILL_SET_TRK_MSG: DB      'H.',CR,LF
                  DB      'The ZFDC Board will update CP/M requested Track.',CR,LF,0
WILL_SET_SIDE_MSG: DB     '.',CR,LF,
                  DB      'The ZFDC Board will update CP/M requested Side.',CR,LF,0
WILL_SET_SEC_MSG: DB      'H.',CR,LF,
                  DB      'The ZFDC Board will update CP/M requested Sector.',CR,LF,0

AT_TRK_MSG:       DB      CR,LF,'At Track:- ',0
AT_TRK:           DB              ' TRACK ',0
AT_SEC:           DB              'H, SECTOR ',0
HEAD0_MSG         DB              'H, SIDE A',0
HEAD1_MSG         DB              'H, SIDE B',0
H_MSG             DB              'H. ',0
ATHEAD0           DB              ' SIDE A ',0
ATHEAD1           DB              ' SIDE B ',0
RD_TEST_MSG       DB      CR,LF,'Continous Sector Read Test.',CR,LF,0
WR_TEST_MSG       DB      CR,LF,'Continous Sector Write Test.',CR,LF,0
PIO_MSG           DB      CR,LF,'PIO Loop test. Type any character -- should echo back.',CR,LF,0
ECHO_MSG DB       CR,LF,'Software Loop test. Type any character -- should echo back.',CR,LF,0
AT_TRK_0_MSG      DB      CR,LF,'Home to Track 0',CR,LF,0
STEP_TO_MSG       DB      CR,LF,'Will step IN one track to Track: ',0
ERROR1_BITS_MSG   DB      'WD2793 Status = ',0
SEEK_TO_MSG       DB      CR,LF,'Will seek IN one track to Track: ',0
R_SEEK_TO_MSG     DB      CR,LF,'Will seek to Track: ',0
R_SECTOR_MSG      DB      'H.  Sector: ',0
SHOW_SEC_MSG      DB      'Track ',0
SHOW_SEC_MSG1     DB      'H, Sector ',0
TRK_ID_MSG        DB      CR,LF,LF,'Track ID = ',0
RANGE_ERROR_MSG   DB      CR,LF,'Out of range Error message number. Got: ',0
R_SEEK_TRACK_MSG DB       CR,LF,'Continous Random Track seek test. ',0
NOT_DONE_MSG      DB      CR,LF,'Code not written yet!',0
R_SEC_RD_MSG      DB      CR,LF,'Random Sector Read test. ',0
S_SEC_RD_MSG      DB      CR,LF,'Sequental Sector Read test. ',0
R_SEC_RDWR_MSG    DB      CR,LF,'Random Sector Read/Write test. ',0
S_SEC_RDWR_MSG    DB      CR,LF,'Sequental Sector Read/Write test. ',0
DEBUG_ON_MSG      DB      CR,LF,'ZFDC Board set to Debug mode. CMDs & ACKs in Track & Sector Displays. ',0
DEBUG_OFF_MSG     DB      CR,LF,'ZFDC Board set to normal display mode. ',0
MULTI_SEC_ROK     DB      CR,LF,LF,'Sectors read via the CMD_RD_MULTI_SECTOR command were read OK.',0
MULTI_SEC_WOK     DB      CR,LF,LF,'Sectors written via the CMD_WR_MULTI_SECTOR command were written OK.',0
SECTOR_DATA_RD    DB      CR,LF,'Data read from ',0
SECTOR_DATA_WR    DB      CR,LF,'Data written to ',0
WR_BOOT_MSG       DB      CR,LF,'Will write Boot Sector to current disk on Track 0 Sector 1. '
                  DB      CR,LF,'Please enter HEX location in RAM of Boot (xxxxH): ',0
DRIVE_SIZE_MSG    DB      CR,LF,'Drive Type: 8" = A,  5"(1.2M) = B,  5"(360K) = C,  3.5" = D:- ',0
INVALID_CMD_MSG:DB        CR,LF,'Invalid Command!',0
FORMATTING1_MSG: DB       CR,LF,LF,'Will Format the current disk (',0
FORMATTING2_MSG: DB       ':) to this format:-',0
SURE_MSG DB       CR,LF,'Are you sure? (Y/N) ',0
SEL_TBL_NUM_MSG:DB        CR,LF,'Select table number from the list above (xxH):- ',0
NEW_TABLE_F_MSG:DB        CR,LF,'Will now proceed to format disk.'
                  DB      CR,LF,'New Format will be: ',0
NEW_TABLE_MSG:    DB      CR,LF,'New Disk table will be:- ',0
```

```
FORM_ABORT_MSG:  DB       CR,LF,'Aborting disk formatting! ',0
SENT_FORMAT_MSG:DB        CR,LF,'Will now send Format Track Commands to ZFDC Board. ',0
MENU_CHOICE_MSG:DB              'Menu selection (xxH) = ',0
FORMAT_DONE_MSG:DB        CR,LF,'Disk format completed successfuly.',0
RD_TRACK_MSG:    DB       CR,LF,'Please enter the TRACK number you wish to read (xxH): ',0
RD1_TRACK_MSG:   DB       CR,LF,'Track Read Correctly:- ',0
HIT_SP_MSG:      DB       CR,LF,'Hit Space Bar for more data. ESC to abort.',0
END_TRACK_MSG:   DB       CR,LF,'End of track data dump.',0
RW_MISMATCH_MSG:DB        CR,LF,'Sector data Read/Write mis-match found!',0
BAD_TABLE_STATUS:DB       CR,LF,'The ZFDC Board may have an incorrect Table Number.',0
INPUT_TRACK_MSG:DB        CR,LF,'Input Track number (xxH):-',0
INPUT_SECTOR_MSG:DB       CR,LF,'Input Sector number (xxH):-',0
INPUT_SIDE_MSG:DBCR,LF,'Input Side (A,B):-',0
NO_SEC_0_MSG:    DB       CR,LF,'Sorry, sectors are numbered 1,2,3...',0
INVALID_SIDE_MSG:DB       CR,LF,'Invalid side.',0
DISPLAY_ON_MSG:  DB       CR,Lf,'Display of Sector Data ON.',0
DISPLAY_OFF_MSG:DB        CR,LF,'Display of Sector Data OFF.',0
FORM_TRK_MSG:    DB       CR,'Formatting (Side A) Track: ',0
FORM_TRK2_MSG:   DB       CR,'Formatting (Side A & B) Track: ',0
EOL_MSG: DB       CR,'                                        ',0
ABORT_FORM_MSG:  DB       CR,LF,'Disk/Track formatting aborted. Note, disk may be un-readable!',0

RAM_DUMP_A_MSG   DB       CR,LF,'Sending command to ZFDC board to obtain RAM data dump!',0
DRIVE_A_PORT_MSG DB       CR,LF,'DRIVE 0: RAM variables:-',CR,LF,0
DRIVE_B_PORT_MSG DB       CR,LF,'DRIVE 1: RAM variables:-',CR,LF,0
DRIVE_C_PORT_MSG DB       CR,LF,'DRIVE 2: RAM variables:-',CR,LF,0
DRIVE_D_PORT_MSG DB       CR,LF,'DRIVE 3: RAM variables:-',CR,LF,0

HW_PORT_MSG      DB       'Hardware Data Port = ',0
NSCTRS_MSG       DB       CR,LF,'Sectors/Track (+1) = ',0
NTRKS_MSG        DB       '   Tracks/Side = ',0
HEADR_MSG        DB       CR,LF,'Header = ',0
GAP1_MSG DB       '   GAP1 = ',0
GAP2_MSG DB       '   GAP2 = ',0
GAP3_MSG DB       '   GAP3 = ',0
GAP4_MSG DB       '     GAP4 = ',0
GAP4R_MSG        DB       '   GAP4R = ',0
SEC_SIZE_FLAG_MSG DB      CR,LF,'SEC SIZE FLAG = ',0
GAP_FILL_CHAR_MSG DB      '        GAP FILL CHAR = ',0
DATA_FILL_CHAR_MSG DB     '        DATA FILL CHAR = ',0
SPECIAL_FLAG_MSG DB       CR,LF,'SPECIAL FLAG = ',0
SKEW_MSG DB       '         SEC SKEW ADDRESS = ',0
FORMAT_NUM_MSG   DB       '   FORMAT NUM = ',0
SYS_TRKS_MSG     DB       CR,LF,'SYSTEM TRKS = ',0
SEC_SIZE_BYTES_MSG DB     '          SEC SIZE BYTES = ',0
TRACK_SIZE_MSG   DB       '     TRACK SIZE = ',0
CPM_TRACK_MSG:   DB       CR,LF,'CPM/DOS Req Track = ',0
CPM_SECTOR_MSG:  DB       '    CPM/DOS Req Sector = ',0
CPM_DS_MSG:      DB       ' Double Sided Disk',0
CPM_SS_MSG:      DB       ' Single Sided Disk',0
NO_SIDES_MSG:    DB       '   No Sides Assigned',0
NEXT_DISK_MSG:   DB       CR,LF,LF,'Hit Spacebar for more data, ESC to abort ',0
Z80_SP_MSG:      DB       CR,LF,'Z80 Stack Pointer = ',0
NO_FORMAT_MSG:   DB       CR,LF,'No Format has been assigned to this disk so far.',0
SD_BIT_MSG:      DB       CR,LF,'Single Density',0
DD_BIT_MSG:      DB       CR,LF,'Double Density',0
```

```
S8_BIT_MSG:       DB       '  8" Drive ',0
S5_BIT_MSG:       DB       '  5" Drive ',0
AS_BIT_MSG:       DB       '   Side A. ',0
BS_BIT_MSG:       DB       '   Side B. ',0
CURRENT_DR_MSG:           DB       CR,LF,'CURRENT DRIVE SELECT = ',0
NO_INITILIZE_MSG:DB       CR,LF,BELL,'Could not Initilzie the ZFDC Board',CR,LF,0
START_TRACK_MSG: DB       CR,LF,'Format starting on Track (Usually 00H or 02H):',0
DISK_FORMAT_MSG: DB       CR,LF,LF,'Will Format disk as follows:-',CR,LF,0
CURRENT_TRACK_MSG:        DB       CR,'Formatting track: ',0
FORMAT_FINISHED_MSG:      DB       CR,LF,'Format of disk Complete.',CR,LF,0
FORMAT_ABORTED_MSG:       DB       CR,LF,BELL,'Format of disk ABORTED!',CR,LF,0
GET_ERROR_NUMBER_MSG:     DB       CR,LF,'Enter Error Code Number (00H-3BH):',0
RETURN_NUMBER_MSG:        DB       CR,LF,LF,'Error Message String Returned:-',CR,0
UNKNOWN_MOD_MSG: DB       CR,LF,BELL,'An unknown post format modification was requested!',CR,LF,0
MSDOS14_MOD_MSG: DB       CR,LF,'Disk has been initilized correctly for a (3.5", 1.44M IBM-PC) MSDOS disk.',CR,LF,0
MSDOS12_MOD_MSG: DB       CR,LF,'Disk has been initilized correctly for a (5", 1.2M IBM-PC) MSDOS disk.',CR,LF,0
MSDOS360_MOD_MSG:DB       CR,LF,'Disk has been initilized correctly for a (5", 360K IBM-PC) MSDOS disk.',CR,LF,0
MSDOS360B_MOD_MSGDB       CR,LF,'Disk has been initilized correctly for a (5", 360K IBM-PC) MSDOS (V1.1) disk.',CR,LF,


;        NOTE TABLE MUST BE WITHIN 0-FFH BOUNDRY

ERROR_TBL         EQU      ($ & 0FF00H) + 100H
         ORG      ERROR_TBL

         DW        PT_NES_FLAG              ;00H,    No Errors flag for previous cmd, sent back to S-100 BIOS
         DW       PT_BUSY_ERR              ;01H,    WD2793 Timeout Error Before CMD was started
         DW       PT_HUNG_ERR              ;02H,    General WD2793 Timeout Error After CMD was sent
         DW       PT_TABLE_ERR             ;03H,    Disk parameter table error
         DW       PT_DRIVE_ERR             ;04H,    Drive not 0-3
         DW       PT_TRACK_RANGE_ERR     ;05H,      Drive track not valid for this disk
         DW       PT_SECTOR_RANGE_ERR    ;06H,      Drive sector not valid for this disk
         DW       PT_SIDE_ERR              ;07H,    No B side on this disk
         DW       PT_SIDE_ERR1             ;08H,    Invalid Side Paramater
         DW       PT_SECTOR_SIZE_ERR     ;09H,      Size of sector > 1024 Bytes

         DW       PT_RESTORE_HUNG  ;0AH,    WD2793 Timeout Error after RESTORE Command
         DW       PT_RESTORE_ERR           ;0BH,    Restore to track 0 Error

         DW       PT_STEPIN_HUNG            ;0CH,    WD2793 Timeout Error after STEP-IN Command
         DW       PT_STEPIN_ERR            ;0DH,    Head Step In Error, DRIVE NOT READY ERROR
         DW       PT_STEPOUT_HUNG  ;0EH,    WD2793 Timeout Error after STEP-OUT Command
         DW       PT_STEPOUT_ERR            ;0FH,    Head Step Out Error, NOT READY ERROR

         DW       PT_SEEK_NV_HUNG  ;10H,    WD2793 Timeout Error after SEEK-NV Command
         DW       PT_SEEK_NV_ERR1  ;11H,    Seek with No Verify Error, NOT READY ERROR
         DW       PT_SEEK_NV_ERR2  ;12H,    Seek with No Verify with SEEK error bit set

         DW       PT_SEEK_TRK_HUNG ;13H,    WD2793 Timeout Error after SEEK with Verify Command
         DW       PT_SEEK_TRK_ERR1 ;14H,    Seek to track in [B'] with Verify error, DRIVE NOT READY ERROR bit set
         DW       PT_SEEK_TRK_ERR2 ;15H,    Seek to track in [B'] with Verify error with SEEK ERROR bit set
         DW       PT_SEEK_REST_HUNG;16H,    WD2793 Timeout Error after RESTORE within SEEK with Verify Command
         DW       PT_SEEK_REST_ERR ;17H,    Restore to track 0, DRIVE NOT READY ERROR within SEEK with Verify Command

         DW       PT_ID_ERR_HUNG           ;18H,    WD2793 Timeout Error after READ TRACK ID Command
         DW       PT_ID_ERR1               ;19H,    Track ID Error, DRIVE NOT READY ERROR
```

```
        DW      PT_ID_ERR2              ;1AH,    Track ID Error, RNF ERROR
        DW      PT_ID_ERR3              ;1BH,    Track ID Error, LOST DATA ERROR
        DW      PT_ID_ERR4              ;1CH,    Track ID Error, CRC ERROR

        DW      PT_RS_HUNG              ;1DH,    WD2793 Timeout Error after Read Sector Command was sent
        DW      PT_RS_ERR1              ;1EH,    Sector read error, DRIVE NOT READY ERROR
        DW      PT_RS_ERR2              ;1FH,    Sector read error, RNF ERROR
        DW      PT_RS_ERR3              ;20H,    Sector read error, LOST DATA ERROR
        DW      PT_RS_ERR4              ;21H,    Sector read error, CRC ERROR
        DW      PT_RS_SK_TRK_HUNG;22H,   WD2793 Timeout Error after SEEK within READ SECTOR Command
        DW      PT_RS_RES_HUNG          ;23H,    WD2793 Timeout Error after RESTORE command within READ SECTOR Command
        DW      PT_RS_RES_ERR           ;24H,    Restore to track 0, DRIVE NOT READY ERROR within READ SECTOR Command
        DW      PT_RS_SKTRK_ERR1 ;25H,   Seek to track error, DRIVE NOT READY ERROR bit set within READ SECTOR Command
        DW      PT_RS_SKTRK_ERR2 ;26H,   Seek to track error with SEEK ERROR bit set within READ SECTOR Command

        DW      PT_WS_HUNG              ;27H,    WD2793 Timeout Error after Read Sector Command was sent
        DW      PT_WS_ERR1              ;28H,    Sector write error, DRIVE NOT READY ERROR
        DW      PT_WS_ERR2              ;29H,    Sector write error, RNF ERROR
        DW      PT_WS_ERR3              ;2AH,    Sector write error, LOST DATA ERROR
        DW      PT_WS_ERR4              ;2BH,    Sector write error, CRC ERROR
        DW      PT_WS_SK_TRK_HUNG;2CH,   WD2793 Timeout Error after SEEK within WRITE SECTOR Command
        DW      PT_WS_RES_HUNG          ;2DH,    WD2793 Timeout Error after RESTORE command within WRITE SECTOR Command
        DW      PT_WS_RES_ERR           ;2EH,    Restore to track 0, DRIVE NOT READY ERROR within WRITE SECTOR Command
        DW      PT_WS_SKTRK_ERR1 ;2FH,   Seek to track error, DRIVE NOT READY ERROR bit set within WRITE SECTOR Command
        DW      PT_WS_SKTRK_ERR2 ;30H,   Seek to track error with SEEK ERROR bit set within WRITE SECTOR Command
        DW      PT_DISK_WP_ERR          ;31H,    Sector write error, Disk is write protected

        DW      PT_CONFIRM_FORMAT;32H,   Confirm disk format cmd request
        DW      PT_FORMAT_HUNG          ;33H,    WD2793 Timeout Error after Track Format Command was sent
        DW      PT_FORMAT1_ERR          ;34H,    Disk format request error
        DW      PT_FORMAT2_ERR          ;35H,    Track format error (Side A)
        DW      PT_FORMAT3_ERR          ;36H,    Track format error (Side B)
        DW      PT_FORMAT4_ERR          ;37H,    Restore error after formatting disk

        DW      PT_RT_ERR_HUNG          ;38H,    Disk Read Track hung error
         DW      PT_RT_ERR              ;39H     Disk Read track error

         DW      PT_DRIVE_INACTIVE      ;3AH    Drive is inactive
         DW      PT_DRIVE_DOOR          ;3BH    Drive door open

         DW      PT_ABORT_FLAG          ;3CH    Special error flag to signify the user aborted a command
        DW      PT_CMD_RANGE_ERR ;3DH,   CMD out or range


PT_NES_FLAG             DB      CR,LF,      'Error = 00H',CR,LF,'No Errors flag sent back to S-100 BIOS.',0
PT_BUSY_ERR             DB      CR,LF,BELL, 'Error = 01H',CR,LF,'WD2793 Timeout Error.',0
PT_HUNG_ERR             DB      CR,LF,BELL, 'Error = 02H',CR,LF,'General WD2793 Timeout Error.',0
PT_TABLE_ERR            DB      CR,LF,BELL, 'CMD=04H,  Error = 03H',CR,LF,'Disk parameter table error.',0
PT_DRIVE_ERR            DB      CR,LF,BELL, 'CMD=05H,  Error = 04H',CR,LF,'Drive not 0-3.',0
PT_TRACK_RANGE_ERR      DB      CR,LF,BELL, 'CMD=07H,  Error = 05H',CR,LF,'Drive track not valid for this disk.',0
PT_SECTOR_RANGE_ERR     DB      CR,LF,BELL, 'CMD=06H,  Error = 06H',CR,LF,'Drive sector not valid for this disk.',0
PT_SIDE_ERR             DB      CR,LF,BELL, 'CMD=08H,  Error = 07H',CR,LF,'No B side on this disk.',0
PT_SIDE_ERR1            DB      CR,LF,BELL, 'CMD=08H,  Error = 08H',CR,LF,'Invalid Side Parameter.',0
PT_SECTOR_SIZE_ERR      DB      CR,LF,BELL, 'CMD=09H,  Error = 09H',CR,LF,'Size of sector > 1024 Bytes.',0

PT_RESTORE_HUNG         DB      CR,LF,BELL, 'CMD=0AH,  Error = 0AH',CR,LF,'WD2793 Timeout Error after RESTORE Command.',0
```

```
PT_RESTORE_ERR          DB       CR,LF,BELL, 'CMD=0AH,  Error = 0BH',CR,LF,'Restore to track 0 Error.',0
PT_STEPIN_HUNG           DB       CR,LF,BELL, 'CMD=0BH,  Error = 0CH',CR,LF,'WD2793 Timeout Error after STEP-IN Command.',0
PT_STEPIN_ERR            DB       CR,LF,BELL,  'CMD=0BH  Error = 0DH',CR,LF,'Head Step-In Error, DRIVE NOT READY. ',0

PT_STEPOUT_HUNG          DB       CR,LF,BELL, 'CMD=0CH,  Error = 0EH',CR,LF,'WD2793 Timeout Error after STEP-OUT Command.',0
PT_STEPOUT_ERR           DB       CR,LF,BELL, 'CMD=0CH,  Error = 0FH',CR,LF,'Head Step Out Error, NOT READY ERROR.',0

PT_SEEK_NV_HUNG          DB       CR,LF,BELL, 'CMD=0DH,  Error = 10H',CR,LF,'WD2793 Timeout Error after SEEK-NV Command.',0
PT_SEEK_NV_ERR1          DB       CR,LF,BELL, 'CMD=0DH,  Error = 11H',CR,LF,'Seek (with No Verify) Error, '
                         DB       'DRIVE NOT READY ERROR.',0
PT_SEEK_NV_ERR2          DB      CR,LF,BELL, 'CMD=0DH,  Error = 12H',CR,LF,'Seek (with No Verify), SEEK error bit set.',0

PT_SEEK_TRK_HUNG DB       CR,LF,BELL, 'CMD=0EH,  Error = 13H',CR,LF,'WD2793 Timeout Error after SEEK '
                         DB       'with Verify Command.',0
PT_SEEK_TRK_ERR1 DB      CR,LF,BELL, 'CMD=0EH,  Error = 14H',CR,LF,'Seek to track (with Verify) error, '
                         DB       'NOT READY bit set.',0
PT_SEEK_TRK_ERR2 DB       CR,LF,BELL, 'CMD=0EH,  Error = 15H',CR,LF,'Seek to track (with Verify) error '
                         DB       'with SEEK ERROR bit set.',0
PT_SEEK_REST_HUNGDB       CR,LF,BELL, 'CMD=0EH,  Error = 16H',CR,LF,'WD2793 Timeout Error after RESTORE'
                         DB       'within a SEEK (With Verify) Command.',0
PT_SEEK_REST_ERR DB      CR,LF,BELL, 'CMD=0EH,  Error = 17H',CR,LF,'Restore to track 0, DRIVE NOT '
                         DB       'READY ERROR within a SEEK (With Verify) Command.',0

PT_ID_ERR_HUNG           DB      CR,LF,BELL, 'CMD=0FH,  Error = 18H',CR,LF,'WD2793 Timeout Error after '
                         DB       'READ TRACK ID Command.',0
PT_ID_ERR1               DB       CR,LF,BELL, 'CMD=0FH,  Error = 19H',CR,LF,'Track ID Error, DRIVE NOT READY ERROR.',0
PT_ID_ERR2               DB       CR,LF,BELL, 'CMD=0FH,  Error = 1AH',CR,LF,'Track ID Error, RNF ERROR.',0
PT_ID_ERR3               DB       CR,LF,BELL, 'CMD=0FH,  Error = 1BH',CR,LF,'Track ID Error, LOST DATA ERROR.',0

PT_ID_ERR4               DB       CR,LF,BELL, 'CMD=0FH,  Error = 1CH',CR,LF,'Track ID Error, CRC ERROR.',0

PT_RS_HUNG               DB       CR,LF,BELL, 'CMD=10H,  Error = 1DH',CR,LF,'WD2793 Timeout Error after '
                         DB       'READ SECTOR Command.',0
PT_RS_ERR1               DB       CR,LF,BELL, 'CMD=10H,  Error = 1EH',CR,LF,'READ SECTOR error, DRIVE NOT READY ERROR.',0
PT_RS_ERR2               DB       CR,LF,BELL, 'CMD=10H,  Error = 1FH',CR,LF,'READ SECTOR error, RNF ERROR.',0
PT_RS_ERR3               DB       CR,LF,BELL, 'CMD=10H,  Error = 20H',CR,LF,'READ SECTOR error, LOST DATA ERROR.',0
PT_RS_ERR4               DB       CR,LF,BELL, 'CMD=10H,  Error = 21H',CR,LF,'READ SECTOR error, CRC ERROR.',0
PT_RS_SK_TRK_HUNGDB       CR,LF,BELL, 'CMD=10H,  Error = 22H',CR,LF,'WD2793 Timeout Error after SEEK ',0
                         DB       'within a READ SECTOR Command.',0
PT_RS_RES_HUNG           DB       CR,LF,BELL, 'CMD=10H,  Error = 23H',CR,LF,'WD2793 Timeout Error after a RESTORE command '
                         DB       'within a READ SECTOR Command.',0
PT_RS_RES_ERR            DB       CR,LF,BELL, 'CMD=10H,  Error = 24H',CR,LF,'Restore to Track 0, DRIVE NOT '
                         DB       'READY ERROR within a READ SECTOR Command.',0
PT_RS_SKTRK_ERR1 DB      CR,LF,BELL, 'CMD=10H,  Error = 25H',CR,LF,'Seek to Track error, DRIVE NOT '
                         DB       'READY ERROR bit set within a READ SECTOR Command.',0
PT_RS_SKTRK_ERR2 DB      CR,LF,BELL, 'CMD=10H,  Error = 26H',CR,LF,'Seek to Track error with SEEK ERROR '
                         DB       'bit set within a READ SECTOR Command.',0

PT_WS_HUNG               DB       CR,LF,BELL, 'CMD=11H,  Error = 27H',CR,LF,'WD2793 Timeout Error '
                         DB       'after WRITE SECTOR Command.',0
PT_WS_ERR1               DB       CR,LF,BELL, 'CMD=11H,  Error = 28H',CR,LF,'WRITE SECTOR error, DRIVE NOT READY ERROR.',0
PT_WS_ERR2               DB       CR,LF,BELL, 'CMD=11H,  Error = 29H',CR,LF,'WRITE SECTOR error, RNF ERROR.',0
PT_WS_ERR3               DB       CR,LF,BELL, 'CMD=11H,  Error = 2AH',CR,LF,'WRITE SECTOR error, LOST DATA ERROR.',0
PT_WS_ERR4               DB       CR,LF,BELL, 'CMD=11H,  Error = 2BH',CR,LF,'WRITE SECTOR error, CRC ERROR.',0
PT_WS_SK_TRK_HUNGDB       CR,LF,BELL, 'CMD=11H,  Error = 2CH',CR,LF,'WD2793 Timeout Error after SEEK '
                         DB       'within WRITE SECTOR Command.',0
```

```
PT_WS_RES_HUNG            DB      CR,LF,BELL, 'CMD=11H,  Error = 2DH',CR,LF,'WD2793 Timeout Error after '
                          DB       'RESTOR command within a WRITE SECTOR Command.',0
PT_WS_RES_ERR             DB      CR,LF,BELL, 'CMD=11H,  Error = 2EH',CR,LF,'Restore to track 0, DRIVE NOT '
                          DB       'READY ERROR within a WRITE SECTOR Command.',0
PT_WS_SKTRK_ERR1 DB      CR,LF,BELL, 'CMD=11H,  Error = 2FH',CR,LF,'Seek to track error, DRIVE NOT '
                          DB       'READY ERROR bit set within a WRITE SECTOR Command.',0
PT_WS_SKTRK_ERR2 DB      CR,LF,BELL, 'CMD=11H,  Error = 30H',CR,LF,'Seek to track error with SEEK ERROR '
                          DB       'bit set within a WRITE SECTOR Command.',0
PT_DISK_WP_ERR            DB      CR,LF,BELL, 'CMD=11H,  Error = 31H',CR,LF,'WRITE SECTOR error, '
                          DB       'Disk is write protected.',0

PT_CONFIRM_FORMATDB      CR,LF,BELL, 'CMD=16H,  Error = 32H',CR,LF,'Confirm disk format command request.',0
PT_FORMAT_HUNG            DB      CR,LF,BELL, 'CMD=16H,  Error = 33H',CR,LF,'WD2793 Timeout Error after Track '
                          DB       'Format Command.',0
PT_FORMAT1_ERR            DB      CR,LF,BELL, 'CMD=16H,  Error = 34H',CR,LF,'Disk format request error.',0
PT_FORMAT2_ERR            DB      CR,LF,BELL, 'CMD=16H,  Error = 35H',CR,LF,'Track format error (Side A).',0
PT_FORMAT3_ERR            DB      CR,LF,BELL, 'CMD=16H,  Error = 36H',CR,LF,'Track format error (Side B).',0
PT_FORMAT4_ERR            DB      CR,LF,BELL, 'CMD=16H,  Error = 37H',CR,LF,'Restore error after formatting disk.',0

PT_RT_ERR_HUNG            DB       CR,LF,BELL, 'CMD=15H,  Error = 39H,',CR,LF,'Disk Read Track error,DRIVE NOT READY.',0
PT_RT_ERR                 DB       CR,LF,BELL, 'CMD=15H',CR,LF,'Error = 39H,',CR,LF,'Disk Read Track (unknown) error.',0

PT_DRIVE_INACTIVE        DB        CR,LF,BELL, 'CMD=3AH',CR,LF,'No detected disk in drive',0
PT_DRIVE_DOOR            DB        CR,LF,BELL, 'CMD=3BH',CR,LF,'Drive door open?',0

PT_ABORT_FLAG             DB       CR,LF,BELL, 'User aborted current command.',0
PT_CMD_RANGE_ERR DB       CR,LF,BELL, 'Error = 39H,  CMD out or range.',0

TIMEOUT_ERROR_MSG:        DB       CR,LF,BELL, 'Timeout Error.',CR,LF
                          DB       '(ZFDC Board did not reply back in time to previous command).',0



; LOOKUP TABLES OF DISK PARAMETER  LISTS &  POINTERS
DPL_POINTERS:             EQU      $
        DW      UNFORMATTED
        DW      STD8IBM
        DW      STDDDT
        DW      DDT256
        DW      DDT512
        DW      DDT1K
        DW      DDT1K2
        DW      MINSDT
        DW      MINDDT
        DW      MINCPM
        DW      DEC
        DW      TOSHIBA
        DW      CDOS
        DW      CDOSDD
        DW      EPSON
        DW      MORROW
        DW      ZENITH
        DW      SUPER
        DW      MSDOS
        DW      MSDOS2
        DW      TRS80
```

```
        DW      IBM144          ;IBM-PC 1.44M 3.5" disk for MSDOS Formatting (Fill character 0's)
        DW      CPM144          ;IBM-PC 1.44M 3.5" disk for CPM Formatting (Fill character E5's)
        DW      IBM120          ;IBM-PC 1.2M   5" disk for MSDOS Formatting (Fill character 0's)
        DW      CPM120          ;IBM-PC 1.2M   5" disk for CPM Formatting (Fill character E5's)

DPL_POINTERS_END EQU    $

DPL_COUNT       EQU     (DPL_POINTERS_END - DPL_POINTERS)/2

;-----------------------------------------------------------------------------------------------
;Note in V2.8 and later, support has been added support for IBM 1.2M/5" and 1.44M/3.5" disk formats.
;As far as the WD2793;is concerned these behave as 8" disks. It's just the disk capacity is larger.
;(Only the old 360K type SD & DD, 300RPM, 5" disks need a clock speed adjustment).
;So whenever I referr to 8" disks, I am also including these 1.2M and 1.4M disks.
;
;       LOOKUP TABLES OF DISK PARAMETERS

;       8" SINGLE DENSITY DRIVE VARIABLES (UNFORMATTED Disk)
UNFORMATTED:
        DB      00011000B       ;Disk HW_BYTE (SDSS)
        DB      1               ;SECTORS PER TRACK (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      0               ;TRACKS PER SIDE
        DB      0               ;HEADER GAP (SD-Systems has 100-27, IBM is 40!)
        DB      0               ;GAP 1 (0's)
        DB      0               ;GAP 2 (FF's)
        DB      0               ;GAP 3 (FF's)
        DB      0               ;GAP 4 (FF's)
        DB      0               ;GAPR (Flag for multiple repeats of GAP4)
        DB      0               ;128 Bytes/sec
        DB      0FFH            ;GAP Format fill character
        DB      0E5H            ;Data area fill character
        DB      0H              ;No special post format
        DW      SKEW_UF         ;Location of this disks sector skew table
        DB      00H             ;Each format will have a unique number. For disk to disk copy
        DB      0               ;Tracks set aside for operating system (eg CPM 2)
        DW      128             ;128 Bytes/sec
        DW      0H              ;Size in bytes of 1 formatted track
        DB      'Unformatted Disk',0
SKEW_UF:
        DB      0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H
        DB      0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H,0H



;       8" SINGLE DENSITY DRIVE VARIABLES (IBM 3740 Format)
STD8IBM:DB      00011000B       ;Disk HW_BYTE (SDSS)
        DB      26+1            ;SECTORS PER TRACK (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      77              ;TRACKS PER SIDE
        DB      40              ;HEADER GAP (SD-Systems has 100-27, IBM is 40!)
        DB      6               ;GAP 1 (0's)
        DB      11              ;GAP 2 (FF's)
        DB      27              ;GAP 3 (FF's)
        DB      247             ;GAP 4 (FF's)
        DB      1               ;GAPR (Flag for multiple repeats of GAP4)
        DB      0               ;128 Bytes/sec
        DB      0FFH            ;GAP Format fill character
```

```
        DB      0E5H            ;Data area fill character
        DB      0H              ;No special post format
        DW      SKEW_IBM ;Location of this disks sector skew table
        DB      01H             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      128             ;128 Bytes/sec
        DW      13F0H           ;Size in bytes of 1 formatted track
        DB      '8" SDSS, 26 X 128 Byte Sectors (IBM 3740 Format)',0
SKEW_IBM:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH
        DB      10H,11H,12H,13H,14H,15H,16H,17H,18H,19H,1AH


;       8" DOUBLE DENSITY (128 BYTE SECTORS)  SD Systems Format
STDDDT: DB      00001000B       ;Disk HW_BYTE  (DDSS)
        DB      50+1            ;SECTORS PER TRACK (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      77              ;TRACKS PER SIDE
        DB      80              ;HEADER GAP (SD-Systems has 100-16, IBM is 80!)
        DB      8               ;GAP 1 (4E's)
        DB      22              ;GAP 2 (4E's)
        DB      16              ;GAP 3 (4E's)
        DB      190  ;(was 199) ;GAP 4 (4E's) (X3 = 597)
        DB      3               ;GAPR (Flag for multiple repeats of GAP4)
        DB      0               ;128 Bytes/sec
        DB      4EH             ;GAP Format fill character
        DB      0E5H            ;Data area fill character
        DB      0H              ;No special post formating modifications of disk req
        DW      SKEW_SDT ;Location of this disks sector skew table
        DB      02H             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      128             ;128 Bytes/sec
        DW      2740H           ;Size in bytes of 1 formatted track
        DB      '8" DDSS, 50 X 128 Byte Sectors (SD_Systems Format)',0
SKEW_SDT:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH
        db      10H,11H,12H,13H,14H,15H,16H,17H,18H,19H,1AH,1BH,1CH,1DH,1EH,1FH
        db      20H,21H,22H,23H,24H,25H,26H,27H,28H,29H,2AH,2BH,2CH,2DH,2EH,2FH
        db      30H,31H,32H

;
;       8" DOUBLE DENSITY (256 BYTE SECTORS) (IBM System 34 Format)
DDT256: DB      00001100B       ;Disk HW_BYTE (DDDS)
        DB      26+1            ;NBR SECTORS PER TRACK (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      77              ;NBR TRACKS PER SIDE
        DB      80              ;HEADER GAP (SD-Systems has 100-54, IBM is 80!)
        DB      12              ;GAP 1 (00's)
        DB      22              ;GAP 2 (4E's)
        DB      54              ;GAP 3 (4E's)
        DB      199             ;GAP 4 (4E's) (X3 = 597)
        DB      3               ;GAPR (Flag for multiple repeats of GAP4)
        DB      1               ;256 Bytes/sec
        DB      4EH             ;GAP Format fill character
        DB      0E5H            ;Data area fill character
        DB      0H              ;No special post formating modifications of disk req
        DW      SKEW_256 ;Location of this disks sector skew table
        DB      03H             ;Each format will have a unique number. For disk to disk copy
```

```
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      256             ;256 Bytes/sec
        DW      2780H           ;Size in bytes of 1 formatted track
        DB      '8" DDDS, 26 X 256 Byte Sectors (IBM System 34 Format)',0
SKEW_256:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH
        db      10H,11H,12H,13H,14H,15H,16H,17H,18H,19H,1AH

;
;       8" DOUBLE DENSITY (512 BYTE SECTORS)
DDT512: DB      00001100B       ;Disk HW_BYTE (DDDS)
        DB      15+1            ;NBR SECTORS PER TRACK (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      77              ;NBR TRACKS PER SIDE
        DB      80              ;HEADER GAP (SD-Systems has 100-54, IBM is 80!)
        DB      12              ;GAP 1
        DB      22              ;GAP 2
        DB      84              ;GAP 3
        DB      200             ;GAP 4 (4E's) (X3 = 597)
        DB      2               ;GAPR (Flag for multiple repeats of GAP4)
        DB      2               ;512 Bytes/sec
        DB      4EH             ;GAP Format fill character
        DB      0E5H            ;Data area fill character
        DB      0H              ;No special post formating modifications of disk req
        DW      SKEW_512 ;Location of this disks sector skew table
        DB      04H             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      512             ;512 Bytes/sec
        DW      2780H           ;Size in bytes of 1 formatted track
        DB      '8" DDDS, 15 X 512 Byte Sectors.',0
SKEW_512:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH


;
;       8" DOUBLE DENSITY (1024 BYTE SECTORS - Single Sided)
DDT1K:  DB      00001000B       ;Disk HW_BYTE (DDSS) ;
        DB      8+1             ;NBR SECTORS PER TRACK (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      77              ;NBR TRACKS PER SIDE
        DB      80              ;INDEX HEADER GAP
        DB      12              ;NBR GAP 1
        DB      22              ;NBR GAP 2
        DB      54              ;NBR GAP 3
        DB      190  ;(was 199)  ;GAP 4
        DB      3               ;GAPR (Flag for multiple repeats of GAP4)
        DB      3               ;1024 Bytes/sec
        DB      4EH             ;GAP Format fill character
        DB      0E5H            ;Data area fill character
        DB      0H              ;No special post formating modifications of disk req
        DW      SKEW_1K         ;Location of this disks sector skew table
        DB      05H             ;Each format will have a unique number. For disk to disk copy
        DB      1               ;Tracks set aside for operating system (eg CPM 2)
        DW      1024            ;1024 Bytes/sec
        DW      2700H           ;Size in bytes of 1 formatted track
        DB      '8" DDSS, 8 X 1024 Byte Sectors.',0
SKEW_1K:
        DB      1H,2H,3H,4H,5H,6H,7H,8H
```

```
;
;       8" DOUBLE DENSITY (1024 BYTE SECTORS - Double Sided)
DDT1K2:  DB      00001100B        ;Disk HW_BYTE (DDDS) ;
         DB      8+1              ;NBR SECTORS PER TRACK (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
         DB      77               ;NBR TRACKS PER SIDE
         DB      80               ;INDEX HEADER GAP
         DB      12               ;NBR GAP 1
         DB      22               ;NBR GAP 2
         DB      54               ;NBR GAP 3
         DB      190  ;(was 199)  ;GAP 4
         DB      3                ;GAPR (Flag for multiple repeats of GAP4)
         DB      3                ;1024 Bytes/sec
         DB      4EH              ;GAP Format fill character
         DB      0E5H             ;Data area fill character
         DB      0H               ;No special post formating modifications of disk req
         DW      SKEW_1KDS        ;Location of this disks sector skew table
         DB      06H              ;Each format will have a unique number. For disk to disk copy
         DB      1                ;Tracks set aside for operating system (eg CPM 2)
         DW      1024             ;1024 Bytes/sec
         DW      2700H            ;Size in bytes of 1 formatted track
         DB      '8" DDDS, 8 X 1024 Byte Sectors.',0
SKEW_1KDS:
         DB      1H,2H,3H,4H,5H,6H,7H,8H

;
;---------5" DRIVES ----------------
;
; 5", 128 byte, SD SD-Systems Format
MINSDT:  DB      00010000B        ;Disk HW_BYTE (SDSS)
         DB      18+1             ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
         DB      35               ;tracks per side
         DB      20               ;index header gap
         DB      6                ;GAP 1
         DB      11               ;GAP 2
         DB      8                ;GAP 3
         DB      221              ;GAP 4 (FF's)
         DB      1                ;GAPR (Flag for multiple repeats of GAP4)
         DB      0                ;128 Bytes/sec
         DB      0FFH             ;GAP Format fill character
         DB      0E5H             ;Data area fill character
         DB      0H               ;No special post formating modifications of disk req
         DW      SKEW_MINSD       ;Location of this disks sector skew table
         DB      07H              ;Each format will have a unique number. For disk to disk copy
         DB      2                ;Tracks set aside for operating system (eg CPM 2)
         DW      128              ;128 Bytes/sec
         DW      0BFFH            ;Size in bytes of 1 formatted track (more than enough!)
         DB      '5" SDSS, 18 X 128 Byte Sectors, (SD-Systems Format).',0
SKEW_MINSD:
         DB      1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH,10H,11H,12H

;
; 5", 128 byte, DD SD-Systems Format
MINDDT:  DB      00000000B        ;Disk HW_BYTE (DDSS)
         DB      28+1             ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
         DB      35               ;tracks per side
         DB      100-16           ;index header gap
```

```
        DB      8               ;GAP 1
        DB      22              ;GAP 2
        DB      16              ;GAP 3
        DB      247             ;GAP 4
        DB      1               ;GAPR (Flag for multiple repeats of GAP4)
        DB      0               ;128 Bytes/sec
        DB      4EH             ;GAP Format fill character
        DB      0E5H            ;Data area fill character
        DB      0H              ;No special post formating modifications of disk req
        DW      SKEW_MINDD      ;Location of this disks sector skew table
        DB      08H             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      128             ;128 Bytes/sec
        DW      17FFH           ;Size in bytes of 1 formatted track (more than enough!)
        DB      '5" DDSS, 28 X 128 Byte Sectors, (SD-Systems Format).',0
SKEW_MINDD:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH
        db      10H,11H,12H,13H,14H,15H,16H,17H,18H,19H,1AH,1BH,1CH

;
; 5", 512 byte, DDDS, 8 sector IBM PC CPM-86 format
MINCPM: DB      00000100B       ;Disk HW_BYTE (DDDS)
        DB      8+1             ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      40              ;tracks per side
        DB      80              ;index header gap
        DB      12              ;GAP 1
        DB      22              ;GAP 2
        DB      80              ;GAP 3
        DB      207             ;GAP 4 (4E's) (1038)
        DB      5               ;GAPR (Flag for multiple repeats of GAP4)
        DB      2               ;512 Bytes/sec
        DB      04EH            ;GAP Format fill character
        DB      0E5H            ;Data area fill character (for CPM86)
        DB      CPM86_FLAG      ;Special post formating modifications of disk req
        DW      SKEW_CPM86      ;Location of this disks sector skew table
        DB      09H             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      512             ;512 Bytes/sec
        DW      17FFH           ;Size in bytes of 1 formatted track (more than enough!)
        DB      '5" DDDS, 8 X 512 Byte Sectors (IBM PC CPM-86 format).',0
SKEW_CPM86:
        DB      1H,2H,3H,4H,5H,6H,7H,8H
;
;
; 5", 512 byte, DDDS, 9 sector DEC VT180 format
DEC:    DB      00000100B       ;Disk HW_BYTE (DDDS)
        DB      9+1             ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      40              ;tracks per side
        DB      80              ;index header gap
        DB      12              ;GAP 1
        DB      22              ;GAP 2
        DB      26              ;GAP 3
        DB      218             ;GAP 4 (4E's) (872)
        DB      4               ;GAPR (Flag for multiple repeats of GAP4)
        DB      2               ;512 Bytes/sec
        DB      04EH            ;GAP Format fill character
```

```
        DB      0E5H            ;Data area fill character (for CPM)
        DB      0               ;No special post formating modifications of disk req
        DW      SKEW_DEC ;Location of this disks sector skew table
        DB      0AH             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      512             ;512 Bytes/sec
        DW      17FFH           ;Size in bytes of 1 formatted track (more than enough!)
        DB      '5" DDDS, 9 X 512 Byte Sectors. (DEC VT180 format).',0
SKEW_DEC:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H
;
;
; 5", 256 byte, DDDS, 16 sector TOSHIBA T-100 format
TOSHIBA:DB      00000100B       ;Disk HW_BYTE (DDDS)
        DB      16+1            ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      35              ;tracks per side
        DB      80              ;index header gap
        DB      12              ;GAP 1
        DB      22              ;GAP 2
        DB      50              ;GAP 3
        DB      183             ;GAP 4 (4E's) (366)
        DB      2               ;GAPR (Flag for multiple repeats of GAP4)
        DB      1               ;256 Bytes/sec
        DB      04EH            ;GAP Format fill character
        DB      0E5H            ;Data area fill character (for CPM)
        DB      0               ;No special post formating modifications of disk req
        DW      SKEW_TOSH       ;Location of this disks sector skew table
        DB      0BH             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      256             ;512 Bytes/sec
        DW      17FFH           ;Size in bytes of 1 formatted track (more than enough!)
        DB      '5" DDDS, 16 X 256 Byte Sectors. (TOSHIBA T-100 format).',0
SKEW_TOSH:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH,10H
;
;
; 5", 128 byte, CROMEMCO CDOS (SINGLE density) Format
CDOS:   DB      00010100B       ;Disk HW_BYTE (SDDS)
        DB      18+1            ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      40              ;tracks per side
        DB      20-8            ;index header gap
        DB      6               ;GAP 1
        DB      11              ;GAP 2
        DB      8               ;GAP 3
        DB      185             ;GAP 4 (FF's)
        DB      1               ;GAPR (Flag for multiple repeats of GAP4)
        DB      0               ;128 Bytes/sec
        DB      0FFH            ;GAP Format fill character
        DB      0E5H            ;Data area fill character
        DB      0H              ;No special post formating modifications of disk req
        DW      SKEW_CDOS       ;Location of this disks sector skew table
        DB      0CH             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      128             ;128 Bytes/sec
        DW      0BFFH           ;Size in bytes of 1 formatted track (more than enough!)
        DB      '5" SDDS, 18 X 128 Byte Sectors, (CROMEMCO CDOS Format).',0
```

```
SKEW_CDOS:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH,10H,11H,12H
;
; 5", 512 byte, CROMEMCO CDOS w/INTL TERM. CP/M Format
CDOSDD: DB      00000100B       ;Disk HW_BYTE (DDDS)
        DB      10+1            ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      40              ;tracks per side
        DB      80              ;index header gap
        DB      12              ;GAP 1
        DB      22              ;GAP 2
        DB      30              ;GAP 3
        DB      214             ;GAP 4 (FF's)
        DB      1               ;GAPR (Flag for multiple repeats of GAP4)
        DB      2               ;512 Bytes/sec
        DB      0FFH            ;GAP Format fill character
        DB      0E5H            ;Data area fill character
        DB      0H              ;No special post formating modifications of disk req
        DW      SKEW_CDOS2      ;Location of this disks sector skew table
        DB      0DH             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      512             ;512 Bytes/sec
        DW      17FFH           ;Size in bytes of 1 formatted track (more than enough!)
        DB      '5" DDDS, 10 X 512 Byte Sectors, (CROMEMCO CDOS/CPM Format).',0
SKEW_CDOS2:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH

; 5", 512 byte, EPSON QX-10 Format
EPSON:  DB      00010100B       ;Disk HW_BYTE (SDDS)
        DB      10+1            ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      40              ;tracks per side
        DB      80              ;index header gap
        DB      12              ;GAP 1
        DB      22              ;GAP 2
        DB      30              ;GAP 3
        DB      214             ;GAP 4 (FF's)
        DB      1               ;GAPR (Flag for multiple repeats of GAP4)
        DB      2               ;512 Bytes/sec
        DB      0FFH            ;GAP Format fill character
        DB      0E5H            ;Data area fill character
        DB      0H              ;No special post formating modifications of disk req
        DW      SKEW_EPSON      ;Location of this disks sector skew table
        DB      0EH             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      512             ;512 Bytes/sec
        DW      0CFFH           ;Size in bytes of 1 formatted track (more than enough!)
        DB      '5" SDDS, 10 X 512 Byte Sectors.  (EPSON QX-10 Format).',0
SKEW_EPSON:
        DB      1H,3H,5H,7H,9H,2H,4H,6H,8H,0AH          ;<-- note skew table
;
;
; 5", 1K byte, DDDS, 5 sector MORROW MD3 format
MORROW: DB      00000100B       ;Disk HW_BYTE (DDDS)
        DB      5+1             ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      40              ;tracks per side
        DB      80              ;index header gap
        DB      12              ;GAP 1
```

```
        DB      22              ;GAP 2
        DB      50              ;GAP 3
        DB      192             ;GAP 4 (4E's) (574)
        DB      3               ;GAPR (Flag for multiple repeats of GAP4)
        DB      3               ;1024 Bytes/sec
        DB      04EH            ;GAP Format fill character
        DB      0E5H            ;Data area fill character (for CPM)
        DB      0               ;No special post formating modifications of disk req
        DW      SKEW_MORROW     ;Location of this disks sector skew table
        DB      0FH             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      1024            ;1024 Bytes/sec
        DW      17FFH           ;Size in bytes of 1 formatted track (more than enough!)
        DB      '5" DDDS, 5 X 1024 Byte Sectors. (MORROW MD3 format).',0
SKEW_MORROW:
        DB      1H,2H,3H,4H,5H
;
;
; 5", 512 byte, DDDS, 5 sector ZENITH Z-100 format
ZENITH: DB      00000100B       ;Disk HW_BYTE (DDDS)
        DB      8+1             ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      40              ;tracks per side
        DB      80              ;index header gap
        DB      12              ;GAP 1
        DB      22              ;GAP 2
        DB      26              ;GAP 3
        DB      242             ;GAP 4 (4E's) (1454)
        DB      6               ;GAPR (Flag for multiple repeats of GAP4)
        DB      2               ;512 Bytes/sec
        DB      04EH            ;GAP Format fill character
        DB      0E5H            ;Data area fill character (for CPM)
        DB      0               ;No special post formating modifications of disk req
        DW      SKEW_ZENITH     ;Location of this disks sector skew table
        DB      10H             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      512             ;512 Bytes/sec
        DW      17FFH           ;Size in bytes of 1 formatted track (more than enough!)
        DB      '5" DDDS, 8 X 512 Byte Sectors. (ZENITH Z-100 format).',0
SKEW_ZENITH:
        DB      1H,2H,3H,4H,5H,6H,7H,8H
;
;
; 5", 512 byte, DDDS, 10 sector SUPERBRAIN QD format
SUPER:  DB      00000100B       ;Disk HW_BYTE (DDDS)
        DB      10+1            ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      35              ;tracks per side
        DB      80              ;index header gap
        DB      12              ;GAP 1
        DB      22              ;GAP 2
        DB      16              ;GAP 3
        DB      177             ;GAP 4 (4E's) (354)
        DB      2               ;GAPR (Flag for multiple repeats of GAP4)
        DB      2               ;512 Bytes/sec
        DB      04EH            ;GAP Format fill character
        DB      0E5H            ;Data area fill character (for CPM)
        DB      0               ;No special post formating modifications of disk req
```

```
        DW      SKEW_SUPER      ;Location of this disks sector skew table
        DB      11H             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      512             ;512 Bytes/sec
        DW      17FFH           ;Size in bytes of 1 formatted track (more than enough!)
        DB      '5" DDDS, 10 X 512 Byte Sectors, (SUPERBRAIN QD format).',0
SKEW_SUPER:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH
;
;
; 5", IBM PC, MSDOS 1.1, 512 byte, DDDS, 8 sector format
MSDOS:  DB      00000100B       ;Disk HW_BYTE (DDDS)
        DB      8+1             ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      40              ;tracks per side
        DB      80              ;index header gap
        DB      12              ;GAP 1
        DB      22              ;GAP 2
        DB      80              ;GAP 3
        DB      193             ;GAP 4 (4E's)
        DB      2               ;GAPR (Flag for multiple repeats of GAP4)
        DB      2               ;512 Bytes/sec
        DB      04EH            ;GAP Format fill character
        DB      00H             ;<--- Data area fill character
        DB      04H             ;<--- Special formating modifications of disk req (+++ NOT DONE YET)
        DW      SKEW_DOS1       ;Location of this disks sector skew table
        DB      12H             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      512             ;512 Bytes/sec
        DW      17FFH           ;Size in bytes of 1 formatted track (more than enough!)
        DB      '5" DDDS, 8 X 512 Byte Sectors,  (IBMPC MSDOS 1.1 format).',0
SKEW_DOS1:
        DB      1H,2H,3H,4H,5H,6H,7H,8H
;
;
; 5", IBM PC, MSDOS 2.x, 512 byte, DDDS, 9 sector format
MSDOS2: DB      00000100B       ;Disk HW_BYTE (DDDS)
        DB      9+1             ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      40              ;tracks per side
        DB      80              ;index header gap
        DB      12              ;GAP 1
        DB      22              ;GAP 2
        DB      80              ;GAP 3
        DB      193             ;GAP 4 (4E's)
        DB      2               ;GAPR (Flag for multiple repeats of GAP4)
        DB      2               ;512 Bytes/sec
        DB      04EH            ;GAP Format fill character
        DB      00H             ;<---- Data area fill character
        DB      03H             ;<--- Special formating modifications of disk req
        DW      SKEW_DOS2       ;Location of this disks sector skew table
        DB      13H             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      512             ;512 Bytes/sec
        DW      17FFH           ;Size in bytes of 1 formatted track (more than enough!)
        DB      '5" DDDS, 9 X 512 Byte Sectors, (IBMPC MSDOS 2.x format).',0
SKEW_DOS2:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H
```

```
;
;
; 5", TRS-80 MOD-III, 512 byte, DDDS, 10 sector format
TRS80:   DB       00000000B        ;Disk HW_BYTE (DDSS)
         DB       10+1             ;sectors per track (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
         DB       40               ;tracks per side
         DB       80               ;index header gap
         DB       12               ;GAP 1
         DB       22               ;GAP 2
         DB       26               ;GAP 3
         DB       137              ;GAP 4 (4E's)
         DB       2                ;GAPR (Flag for multiple repeats of GAP4)
         DB       2                ;512 Bytes/sec
         DB       04EH             ;GAP Format fill character
         DB       0E5H             ;Data area fill character
         DB       0H               ;Special formating modifications of disk req (+++ NOT DONE YET)
         DW       SKEW_TRS ;Location of this disks sector skew table
         DB       14H              ;Each format will have a unique number. For disk to disk copy
         DB       2                ;Tracks set aside for operating system (eg CPM 2)
         DW       512              ;512 Bytes/sec
         DW       17FFH            ;Size in bytes of 1 formatted track (more than enough!)
         DB       '5", DDSS, 10 X 512 Byte Sectors, (TRS-80 MOD-III format).',0
SKEW_TRS:
         DB       1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH



;-------------- SPECIAL CASES FOR 3.5" DISKS and 1.2M 5" DISKS --------------------------
;        3.5" DOUBLE DENSITY (1.4M IBM-PC Format, this is my best guess so far)

IBM144:  DB       00001100B        ;Disk HW_BYTE (DDDS)
         DB       18+1             ;NBR SECTORS PER TRACK (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
         DB       80               ;NBR TRACKS PER SIDE
         DB       80               ;HEADER GAP (SD-Systems has 100-54, IBM is 80!)
         DB       12               ;GAP 1
         DB       22               ;GAP 2
         DB       84               ;GAP 3
         DB       200              ;GAP 4 (4E's) (X3 = 597)
         DB       1                ;GAPR (Flag for multiple repeats of GAP4)
         DB       2                ;512 Bytes/sec
         DB       4EH              ;GAP Format fill character
         DB       00H              ;<--- Data area fill character
         DB       01H              ;<----Note special post formating modifications of disk req
         DW       SKEW_144 ;Location of this disks sector skew table
         DB       15H              ;Each format will have a unique number. For disk to disk copy
         DB       2                ;Tracks set aside for operating system (eg CPM 2)
         DW       512              ;512 Bytes/sec
         DW       2E90H            ;Size in bytes of 1 formatted track
         DB       '1.4M (For MSDOS) DDDS, 18 X 512 Byte Sectors, 80 Tracks.',0
SKEW_144:
         DB       1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH
         DB       10H,11H,12H

 CPM144: DB       00001100B        ;Disk HW_BYTE (DDDS)
         DB       18+1             ;NBR SECTORS PER TRACK (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
         DB       80               ;NBR TRACKS PER SIDE
```

```
        DB      80              ;HEADER GAP (SD-Systems has 100-54, IBM is 80!)
        DB      12              ;GAP 1
        DB      22              ;GAP 2
        DB      84              ;GAP 3
        DB      200             ;GAP 4 (4E's) (X3 = 597)
        DB      1               ;GAPR (Flag for multiple repeats of GAP4)
        DB      2               ;512 Bytes/sec
        DB      4EH             ;GAP Format fill character
        DB      0E5H            ;<--- Data area fill character
        DB      0H              ;No special post formating modifications of disk req
        DW      SKEW_CPM144     ;Location of this disks sector skew table
        DB      16H             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      512             ;512 Bytes/sec
        DW      2E90H           ;Size in bytes of 1 formatted track
        DB      '1.4M (For CPM) DDDS, 18 X 512 Byte Sectors, 80 Tracks.',0
SKEW_CPM144:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH
        DB      10H,11H,12H


 ;      5" HIGH DENSITY (1.2M IBM-PC Format, this is my best guess so far)
IBM120: DB      00001100B       ;Disk HW_BYTE (DDDS)
        DB      15+1            ;NBR SECTORS PER TRACK (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      80              ;NBR TRACKS PER SIDE
        DB      80              ;HEADER GAP (SD-Systems has 100-54, IBM is 80!)
        DB      12              ;GAP 1
        DB      22              ;GAP 2
        DB      84              ;GAP 3
        DB      200             ;GAP 4 (4E's) (X3 = 597)
        DB      1               ;GAPR (Flag for multiple repeats of GAP4)
        DB      2               ;512 Bytes/sec
        DB      4EH             ;GAP Format fill character
        DB      00H             ;<---Data area fill character
        DB      02H             ;<--- Special post formating modifications of disk req
        DW      SKEW_120 ;Location of this disks sector skew table
        DB      17H             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      512             ;512 Bytes/sec
        DW      2780H           ;Size in bytes of 1 formatted track
        DB      '1.2M (For MSDOS) DDDS, 5", 18 X 512 Byte Sectors, 80 Tracks.',0
SKEW_120:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH


 ;      5" HIGH DENSITY (1.2M IBM-PC Format, this is my best guess so far)
CPM120: DB      00001100B       ;Disk HW_BYTE (DDDS)
        DB      15+1            ;NBR SECTORS PER TRACK (+1, because sectors are numbered 1,2,3 (not 0,1,2,3)
        DB      80              ;NBR TRACKS PER SIDE
        DB      80              ;HEADER GAP (SD-Systems has 100-54, IBM is 80!)
        DB      12              ;GAP 1
        DB      22              ;GAP 2
        DB      84              ;GAP 3
        DB      200             ;GAP 4 (4E's) (X3 = 597)
        DB      1               ;GAPR (Flag for multiple repeats of GAP4)
        DB      2               ;512 Bytes/sec
```

```
        DB      4EH             ;GAP Format fill character
        DB      0E5H            ;<---Data area fill character
        DB      0H              ;No special post formating modifications of disk req
        DW      SKEW_CPM120     ;Location of this disks sector skew table
        DB      18H             ;Each format will have a unique number. For disk to disk copy
        DB      2               ;Tracks set aside for operating system (eg CPM 2)
        DW      512             ;512 Bytes/sec
        DW      2780H           ;Size in bytes of 1 formatted track
        DB      '1.2M (For CPM) DDDS, 5", 18 X 512 Byte Sectors, 80 Tracks.',0
SKEW_CPM120:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H,0AH,0BH,0CH,0DH,0EH,0FH

;-------------------------------------------------------------------------------

MSDOS_14BOOT_DATADB     0EBH,3CH,90H    ;Unused. Cold boot jump code.
                        DB      'MSDOS5.0'      ;OEM name  & version
                        DW      0200H           ;bytes/sector
                        DB      01H             ;Sectors/cluster
                        DW      0001H           ;Reserved Sectors (for Dir, FAT etc)
                        DB      02H             ;Number of FAT copies
                        DW      00E0H           ;Max number of root directory entries
                        DW      0B040H          ;Total number of sectors in logical image
                        DB      0F0H            ;Media descriptor byte
                        DW      0009H           ;Number of sectors in FAT
                        DW      0012H           ;Number of sectors pet track
                        DW      0002H           ;Number of heads
                        DW      0000H           ;Number of hidden sectors

MSDOS_12BOOT_DATADB     0EBH,3CH,90H    ;Unused. Cold boot jump code.
                        DB      'MSDOS5.0'      ;OEM name  & version
                        DW      0200H           ;bytes/sector
                        DB      01H             ;Sectors/cluster
                        DW      0001H           ;Reserved Sectors (for Dir, FAT etc)
                        DB      02H             ;Number of FAT copies
                        DW      00E0H           ;Max number of root directory entries
                        DW      0960H           ;Total number of sectors in logical image
                        DB      0F9H            ;Media descriptor byte
                        DW      0007H           ;Number of sectors in FAT
                        DW      000FH           ;Number of sectors pet track
                        DW      0002H           ;Number of heads
                        DW      0000H           ;Number of hidden sectors

MSDOS_360BOOT_DATA      DB      0EBH,3CH,90H    ;Unused. Cold boot jump code.
                        DB      'MSDOS5.0'      ;OEM name  & version
                        DW      0200H           ;bytes/sector
                        DB      02H             ;Sectors/cluster
                        DW      0001H           ;Reserved Sectors (for Dir, FAT etc)
                        DB      02H             ;Number of FAT copies
                        DW      0070H           ;Max number of root directory entries
                        DW      02D0H           ;Total number of sectors in logical image
                        DB      0FDH            ;Media descriptor byte
                        DW      0002H           ;Number of sectors in FAT
                        DW      0009H           ;Number of sectors pet track
                        DW      0002H           ;Number of heads
                        DW      0000H           ;Number of hidden sectors
```

```
MSDOS_360BBOOT_DATA       DB       0EBH,3CH,90H     ;Unused. Cold boot jump code. (Note I have not tested with DOS V1.1)
                          DB       'MSDOS5.0'       ;OEM name  & version
                          DW       0200H            ;bytes/sector
                          DB       02H              ;Sectors/cluster
                          DW       0001H            ;Reserved Sectors (for Dir, FAT etc)
                          DB       02H              ;Number of FAT copies
                          DW       0070H            ;Max number of root directory entries
                          DW       02D0H            ;Total number of sectors in logical image
                          DB       0FFH             ;<--Media descriptor byte
                          DW       0002H            ;Number of sectors in FAT
                          DW       0008H            ;<-- Number of sectors pet track
                          DW       0002H            ;Number of heads
                          DW       0000H            ;Number of hidden sectors


; THE FOLLOWING RAM LOCATIONS ARE REQ
TRACK_ID_BUFFER: DS       6H       ;Buffer to store track ID
TABLE_NUMBER_OLD:DB       0H       ;Store of old Disk Parameter Table
ASCII_FLAG:               DB       0H       ;Flag for HEX or character display in sector dump
TABLE_NUMBER_NEW:DB       0H       ;Store of current Disk Parameter Table
LINE_COUNT                DB       12       ;For Track data dump
HALT_CHECK_FLAG           DB       0H       ;Flag to check if a keyboard halt display is required
DISPLAY_FLAG              DB       0H       ;if Z show sector data. If 0FFH turn off display
DEBUG_FLAG                DB       0H       ;to toggle ZFDC board debug mode
OLD_DRIVE                 DB       0H       ;Store for old drive number in case an abort
CURRENT_FORMAT_TRACK      DB       0H       ;Store for current track being formatted
DR_STORE          DB      0H       ;Store used in RAM data dump routine


SP_SAVE:          DW       SP_SAVE
                  DS       100H
STACK:       EQU  $
                                    ;The buffer will work out to be at ~5000H
S_BUFF       EQU  ($ & 0F000H) + 1000H      ;Put SECTOR_BUFFER on an 1K boundry for easy data display

                  ORG      S_BUFF
SECTOR_BUFFER:            DS       2100H    ;Buffer to store sector data
SECTOR_BUFFER2:           DS       2100H    ;Buffer to store second sector data
;END


;END
```