```
;THIS IS A PROGRAM TO FORMAT CPM DISKS USING THE S100COMPUTERS Z80/WD2793 (ZFDC) Controller board
;Note this is just a cut and past version of the origional ZFDCDIAG.Z80 Diagnostic program.
;See that program first if you want to understand/modify this one.

;      JOHN MONAHAN  (monahan@vitasoft.org)     12/26/2010
;      VERSION       0.1    Initial code
;
;      V0.1                 Started from ZFDCDIAG.Z80 Program
;      V1.0                 Added ability to copy system tracks from one disk to another
;      V1.1                 Copy system tracks not working properly
;      V1.2   3/7/2011      Write system tracks with Multi-sector R/W. Allow format of system, data or all of disk
;      V1.3   05/1/2011     Added support for IBM 1.2M/5" and 1.44M/3.5" disk formats + after format mods

;Please see the notes in ZFDC.Z80 & ZFDCDIAG.Z80 to understand the use of IX and IY

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

STATUS_DELAY  EQU    5            ;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
IBM_FORMAT_NUMBER EQU      1            ;ZFDC Disk Format number for an 8" SDSS IBM disk
SIZE_FLAG     EQU    0            ;0 for IBM 128 byte sectors


; BDOS EQUATES (VERSION 3)
;
RDCON         EQU    1            ;CP/M Read character
WRCON         EQU    2            ;CP/M Write character
COSTATUS      EQU    11           ;CP/M get consol status
CPM_SEL_DISK  EQU    14           ;CP/M Select a disk
CPM_RESET_DISK  EQU 37           ;Reset a specific disk
CPM_RESET_DISKS EQU 13           ;Reset all CPM disks
CPM_INVALID_DISK EQU       04          ;CP/M error code for an invalid drive selection

BDOS          EQU    5            ;All CPM BDOS calls to there

; 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 this program 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) here 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
; CPM Consol I/O Routines.   Crude, but simple I/O!
; You can go direct to hardware if you like. See ZFDCDIAG.Z80
; Other than communications with the ZFDC board there are no other hardware links in this code.

;      Main console status routine using CPM.
CONSTAT:PUSH  BC
```

```
        PUSH   DE
        PUSH   HL
        LD     C,COSTATUS
        CALL   BDOS                ;Get console status
        POP    HL
        POP    DE
        POP    BC
        OR     A,A
        RET                        ;Z if no character available

;       Main consol input routine using CPM. Character in [A] on return. All other registers unchanged
CI:     PUSH   BC
        PUSH   DE
        PUSH   HL
        LD     C,RDCON
        CALL   BDOS                ;Get a character from keyboard (& ECHO <----- note)
        POP    HL
        POP    DE
        POP    BC
        RET


;       Main consol output routine using CPM. Character in [C]. All registers unchanged except [AF]
CO:     PUSH   BC
        PUSH   DE
        PUSH   HL
        LD     E,C
        LD     C,WRCON
        CALL   BDOS
        POP    HL
        POP    DE
        POP    BC
        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

       CALL   BIOS_ZFDC_ALIVE           ;See if ZFDC Board has already been initilized correctly by CPM
       JP     Z,LOOP              ;If so then just go to it

       LD     HL,INITILIZING_MSG  ;tell user we had to initilize ZFDC board
       CALL   PMSG
                                  ;If not then we need to initilize it
       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,LOOP
BOARD_ERROR:
       LD     HL,NO_INITILIZE_MSG
       CALL   PMSG
       JP     0H


LOOP:  CALL   FORMAT_DISK         ;Format a disk
       JR     NZ,ANOTHER_DISK           ;If NZ returned there was an error
       LD     A,(@FORMAT)         ;If IBM 8" disk ,offer to copy system tracks
       CP     A,IBM_FORMAT_NUMBER ;(1)
       JR     NZ,FINISH
       LD     HL,ADD_SYSTEM_MSG   ;Add system tracks to the new disk ?
       CALL   PMSG
       LD     A,(@DRIVE)
       CALL   PACC
       LD     HL,ADD2_SYSTEM_MSG
       CALL   PMSG
       CALL   GETCMD
```

```
       CP     A,'Y'
       CALL   Z,SYSGEN            ;Note this routines assumes the CPM/DOS system is on ZFDC drive 0

ANOTHER_DISK:                     ;ALWAYS on first two disk tracks. May need to adjust for some (5") disks
       LD     HL,ANOTHER_DISK_MSG
       CALL   PMSG
       CALL   GETCMD
       CP     A,'Y'
       JR     NZ,FINISH
       CALL   CRLF
       JR     LOOP

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 & Reboot


;-----------------------------------------------------------------------------------------
;CORE ROUTINES TO COMMUNICATE WITH ZFDC BOARD

BIOS_ZFDC_ALIVE                         ;See if ZFDC Board is present and initilized
       LD     C,CMD_ZFDC_ALIVE
       CALL   S100OUT
       LD     C,0FH               ;Send a test Byte
       CALL   S100OUT
       CALL   WAIT_FOR_ACK        ;Wait and see what comes back
       CP     A,0F0H              ;See if an inverted copy came back
       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 drive of 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_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_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_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_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_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_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_GET_SEC_SIZE:                ;Get the currently selected drive's sector size
       LD     C,CMD_GET_SEC_SIZE
       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_SEC_COUNT:               ;Get the currently selected drive sectors/track+1
       LD     C,CMD_GET_SEC_COUNT
       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_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,(@SEC_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

       LD     HL,(@DMA)           ;Point to track buffer

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     A,(@SEC_COUNT)
```

```
       DEC    A
       LD     (@SEC_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]
       LD     (@DMA),HL           ;Update DMA address
       RET



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,(@SEC_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

       LD     HL,(@DMA)           ;Point to track buffer

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     A,(@SEC_COUNT)
       DEC    A
       LD     (@SEC_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]
       LD     (@DMA),HL           ;Update DMA address
       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




;----------- MAIN PROGRAM -----------------------------------------------------------------

;Select Drive in hardware and assign a disk format to it and format the disk in that format.

FORMAT_DISK:
       LD     HL,FORMAT_DRIVE_MSG
       CALL   PMSG
       CALL   GET_DISK            ;Return with disk select in [D], Z if invalid disk
       JR     Z,SET_DRIVE1        ;0, 1, 2 or 3 (only)

INVALID_DRIVE:
       LD     HL,INVALID_DRIVE_MSG
       CALL   PMSG
       XOR    A,A
       DEC    A
       RET

SET_DRIVE1:
       LD     A,D                 ;Selected disk returned in [D]
       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
       XOR    A,A
       DEC    A
       RET                         ;Ret NZ

TABLE_8:
       LD     HL,TABLE_MSG_8
       JR     GOT_TABLE
TABLE_5:
       LD     HL,TABLE_MSG_5
GOT_TABLE:
       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     HL,BAD_TABLE_MSG
       CALL   PMSG
       XOR    A,A
       DEC    A
       RET                         ;Ret NZ

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              ;Note this CMD also selects the drive
       JP     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
```

```
SEND_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,30H               ;0,1,2, = 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
       RET    Z
       CP     A,'Y'
       JP     Z,FORMAT_1
ABORT_FORMAT:
       LD     HL,FORM_ABORT_MSG   ;"Will abort format"
       CALL   PMSG
       CALL   CRLF
       XOR    A,A
       DEC    A
       RET                        ;Ret NZ

FORMAT_1:
       LD     HL,START_TRACK_MSG  ;"Format System, Data or All Tracks:-"
       CALL   PMSG
       CALL   GETCMD
       CP     ESC
       RET    Z
       CP     A,'S'
       JP     NZ,NOT_S
       XOR    A,A
       LD     (CURRENT_FORMAT_TRACK),A ;Tracks 0 & 1 only
       LD     A,2
       LD     (@END_TRACK),A
       JP     ALL_SETUP
NOT_S:
       CP     A,'D'
       JP     NZ,NOT_D
       LD     A,2
       LD     (CURRENT_FORMAT_TRACK),A ;Tracks 2 to end
       LD     A,(IX+NTRKS)
       LD     (@END_TRACK),A
       JP     ALL_SETUP
NOT_D:
```

```
       CP     A,'A'
       JP     NZ,ABORT_FORMAT
       XOR    A,A
       LD     (CURRENT_FORMAT_TRACK),A ;Tracks 0 to end
       LD     A,(IX+NTRKS)
       LD     (@END_TRACK),A

ALL_SETUP:
       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
       JR     NZ,FORMAT_ABORT

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
        LD      D,A
        LD      A,(@END_TRACK)
        CP      A,D                 ;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      Z,POST_FORMAT       ;See if any post formatting modifications are required

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
        XOR     A,A
        DEC     A
        RET                         ;Ret NZ

POSTMOD_ABORT:
        LD      HL,POSTMOD_ABORTED_MSG
        CALL    PMSG
        CALL    BIOS_SET_HOME       ;Start at track 0
        CALL    NZ,SHOW_ERRORS
        XOR     A,A
        DEC     A
        RET                         ;Ret NZ


POST_FORMAT:                        ;For MSDOS Disks the disk needs to be modified/Initilized
        LD      A,(IX+SPECIAL_FLAG)
        OR      A,A
        JP      Z,MOD_OK            ;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
        RET

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,POSTMOD_ABORT

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

        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,POSTMOD_ABORT

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

        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,POSTMOD_ABORT

        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,POSTMOD_ABORT
```

```
        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,POSTMOD_ABORT
        LD      HL,MSDOS14_MOD_MSG
        CALL    PMSG
MOD_OK:         XOR     A,A
        RET

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,POSTMOD_ABORT

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

        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,POSTMOD_ABORT

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

        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,POSTMOD_ABORT

        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,POSTMOD_ABORT

        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,POSTMOD_ABORT
        LD      HL,MSDOS12_MOD_MSG
        CALL    PMSG
        XOR     A,A
        RET

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,POSTMOD_ABORT

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

        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,POSTMOD_ABORT

        LD      A,04H               ;Also sector 4 with FD,FF,FF for second FAT
        CALL    BIOS_SET_SECTOR
```

```
        CALL    NZ,SHOW_ERRORS
        JP      NZ,POSTMOD_ABORT

        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,POSTMOD_ABORT

        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,POSTMOD_ABORT

        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,POSTMOD_ABORT
        LD      HL,MSDOS360_MOD_MSG
        CALL    PMSG
        XOR     A,A
        RET

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,POSTMOD_ABORT

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

        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,POSTMOD_ABORT

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

        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,POSTMOD_ABORT

        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,POSTMOD_ABORT

        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,POSTMOD_ABORT
        LD      HL,MSDOS360B_MOD_MSG
        CALL    PMSG
        XOR     A,A
        RET


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 from user a disk select (A, B, C or D)
```

```
GET_DISK:                       ;Get A,B,C or D for selected drive in [A]
       LD     HL,DRIVE_MSG
       CALL   PMSG
       CALL   GETCMD
       CP     A,'0'
       JP     NZ,NOT_DRIVE_A
       LD     HL,SEL_DRIVE_A_MSG
       CALL   PMSG
       LD     D,0
       XOR    A,A
       RET
NOT_DRIVE_A:
       CP     A,'1'
       JP     NZ,NOT_DRIVE_B
       LD     HL,SEL_DRIVE_B_MSG
       CALL   PMSG
       LD     D,1
       XOR    A,A
       RET
NOT_DRIVE_B:
       CP     A,'2'
       JP     NZ,NOT_DRIVE_C
       LD     HL,SEL_DRIVE_C_MSG
       CALL   PMSG
       LD     D,2
       XOR    A,A
       RET
NOT_DRIVE_C:
       CP     A,'3'
       JP     NZ,NOT_DRIVE_D
       LD     HL,SEL_DRIVE_D_MSG
       CALL   PMSG
       LD     D,3
       XOR    A,A
       RET
NOT_DRIVE_D
       XOR    A,A
       DEC    A
       RET

; {Print a one line description of the disk format for a drive. (Drive Paramater table in [IX])

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



; Copy System Tracks from another disk. This is currently written for 8" IBM SDSS disks
; Remember depending on sector sizes number the number of sectors copied and written may be different
; for other disk formats. Some day I will generalize the routine using IX for current format paramaters etc.


SYSGEN:       LD     HL,CPM_SOURCE_MSG   ;"For the source CPM system disk"
       CALL   PMSG

       CALL   GET_DISK            ;Return with disk select in [D], NZ if invalid disk
       JP     NZ,SYSGEN_ERR
       LD     A,D                 ;Get drive # in [A]
       LD     (@SOURCE_DRIVE),A   ;Store for below

       CALL   BIOS_SET_DRIVE             ;Select Source disk, in [A]
       CALL   NZ,SHOW_ERRORS
       JP     NZ,SYSGEN_ERR

       CALL   BIOS_SET_HOME       ;To Track 0
       CALL   NZ,SHOW_ERRORS
       JP     NZ,SYSGEN_ERR

       CALL   BIOS_GET_FORMAT            ;Check it actully has an IBM format
       LD     A,D
       CP     A,IBM_FORMAT_NUMBER ;(1)
       LD     (@SYS_FORMAT),A            ;store system format here (not used currently)
       JR     Z,FORMAT_OK         ;Did it return an unformatted drive

       LD     HL,NOT_IBM_MSG             ;Drive is not IBM format
       CALL   PMSG
       JP     SYSGEN_ERR

FORMAT_OK:
       LD     HL,COPYING_SYSTEM_MSG
       CALL   PMSG
       LD     A,(@SOURCE_DRIVE)
       CALL   PACC                ;Drive:# + CRLF
       LD     HL,CRLF_MSG
       CALL   PMSG

       CALL   BIOS_SET_HOME       ;To Track 0
       CALL   NZ,SHOW_ERRORS
       JP     NZ,SYSGEN_ERR
```

```
        XOR     A,A
        LD      (@TRACK),A          ;System is assumed to be on tracks 0 and (usually 1)
                                    ;see below
        CALL    BIOS_SET_TRACK
        CALL    NZ,SHOW_ERRORS
        JP      NZ,SYSGEN_ERR

        LD      A,1                 ;For multisec track read start on sector 1 each time
        CALL    BIOS_SET_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,SYSGEN_ERR

        LD      A,52                ;Will copy two whole track at a time
        LD      (@SEC_COUNT),A              ;Store for Multi sector read routine

        LD      HL,TRACK_BUFFER             ;Build image here
        LD      (@DMA),HL

        CALL    BIOS_MULTI_READ_SECTOR  ;<<<<<<<<<<< Read 52 sectors
        CALL    NZ,SHOW_ERRORS
        JP      NZ,SYSGEN_ERR

WRITE_SYSTEM:                       ;Now write it to destination disk
        LD      A,(@DRIVE)          ;Get back the just formatted disk in [A]
        CALL    BIOS_SET_DRIVE              ;Select Source disk, in [A]
        CALL    NZ,SHOW_ERRORS
        JP      NZ,SYSGEN_ERR

        CALL    BIOS_GET_FORMAT             ;Check it actully has an IBM format
        LD      A,D
        CP      A,IBM_FORMAT_NUMBER ;(1)
        JR      Z,WFORMAT_OK        ;Did it return an unformatted drive

        LD      HL,NOT_IBM_MSG              ;Drive is not IBM format
        CALL    PMSG
        JP      SYSGEN_ERR

WFORMAT_OK:
        LD      HL,WRITING_SYSTEM_MSG
        CALL    PMSG
        LD      A,(@DRIVE)
        CALL    PACC                ;Drive:# + CRLF
        LD      HL,CRLF_MSG
        CALL    PMSG

        CALL    BIOS_SET_HOME       ;To Track 0
        CALL    NZ,SHOW_ERRORS
        JP      NZ,SYSGEN_ERR
```

```
        XOR     A,A
        LD      (@TRACK),A          ;System is assumed to be on tracks 0 and 1

        CALL    BIOS_SET_TRACK
        CALL    NZ,SHOW_ERRORS
        JP      NZ,SYSGEN_ERR

        LD      A,1
        CALL    BIOS_SET_SECTOR
        CALL    NZ,SHOW_ERRORS
        JP      NZ,SYSGEN_ERR

        LD      A,52                ;Will copy two whole track at a time
        LD      (@SEC_COUNT),A              ;Store for Multi sector read routine

        LD      HL,TRACK_BUFFER             ;System image is here
        LD      (@DMA),HL

        CALL    BIOS_MULTI_WRITE_SECTOR     ;<<<<<<<<<<< Write 52 sectors
        CALL    NZ,SHOW_ERRORS
        JP      NZ,SYSGEN_ERR

        LD      HL,SYSGEN_DONE_MSG
        CALL    PMSG
        XOR     A,A
        RET                         ;Ret Z

SYSGEN_ERR:
        LD      HL,SYSGEN_ERR_MSG
        CALL    PMSG
        XOR     A,A
        DEC     A
        RET                         ;Ret NZ



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

; Main Routing to get a command character from Keyboard. Character returned in [A]
GETCMD:       CALL   CI                 ;GET A CHARACTER, convert to UC, ECHO it
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


WAIT_FOR_ACK:                     ;Wait for ZFDC Board to send back an acknowledge/error code
       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                   ;0FFH in [A] indicates timeout
       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



; 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
```

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

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


; Print a message @[HL] up to 0
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


; 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


;       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_HUNG   DB     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_HUNG   DB     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_HUNG   DB      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_FORMAT   DB      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



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

SIGNON:                    DB     CR,LF
                    DB     'ZFDC Board Format Program.'
                    DB     '(V1.2 by John Monahan 03/07/2012)',0
FORMAT_DRIVE_MSG    DB     CR,LF,'This Program will FORMAT a floppy disk attached to the ZDFD FDC board.'
                    DB     CR,LF,'Drives are numbered "relative" to the FDC (Not CPM A:, B:, C:, etc. numbering).',0
NO_INITILIZE_MSG:   DB     CR,LF,BELL,'Could not Initilzie the ZFDC Board',0
ANOTHER_DISK_MSG    DB     CR,LF,'Do you want to format another disk (Y/N):',0
CPM_SOURCE_MSG              DB     CR,LF,'For the source CPM system disk',0
DRIVE_MSG:          DB     CR,LF,'Please select the relative ZFDC drive # (0,1,2 or 3):- ',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
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

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

BAD_TABLE_MSG:             DB     CR,LF,'Invalid Format Table Number!',CR,LF,0
FORMATTING1_MSG:    DB     CR,LF,'Will Format the ZFDC relative disk #',0
FORMATTING2_MSG:    DB     ' to this format:-',0
SURE_MSG            DB     CR,LF,'Are you sure? (Y/N) ',0
FORM_ABORT_MSG:            DB     CR,LF,'Aborting disk formatting! ',0
START_TRACK_MSG:    DB     CR,LF,'Format System Tracks Only (S)'
                    DB     CR,LF,'Format Data Tracks Only   (D)'
                    DB     CR,LF,'Format All Tracks         (A) :- ',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. (Note may need to reboot CPM if a new disk format).',CR,LF,0
FORMAT_ABORTED_MSG: DB     CR,LF,BELL,'Format of disk ABORTED!',CR,LF,0
RANGE_ERROR_MSG            DB     CR,LF,'Out of range Error message number. Got: ',0
H_MSG               DB            'H. ',0
ADD_SYSTEM_MSG:            DB     CR,LF,'Copy CPM/DOS system tracks to drive ',0
ADD2_SYSTEM_MSG:    DB           ' from another drive? (Y/N) ',0
```

```
SYSGEN_MSG:          DB     CR,LF,'Copying CPM/DOS System tracks from drive: ',0
SYSGEN_MSG1:         DB     ' to drive: ',0
SYSGEN_ERR_MSG:              DB     CR,LF,'System image copying aborted.',CR,LF,0
SYSGEN_DONE_MSG:     DB     CR,LF,'System image copy correctly to the newly formatted disk.',CR,LF,0
COPYING_SYSTEM_MSG   DB     CR,LF,'Copying System tracks from relative ZFDC disk ',0
WRITING_SYSTEM_MSG   DB     CR,LF,'Writing System tracks on newly formatted ZFDC disk ',0
INITILIZING_MSG:     DB     CR,LF,'Initilizing ZFDC Board. '
                     DB     CR,LF,'Remember when finished, CPM may have to be rebooted.',CR,LF,0
NOT_IBM_MSG:         DB     CR,LF,'Sorry in this version only an IBM 8" SDSS source disk can be used!',0
CRLF_MSG:            DB     CR,LF,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_MSG    DB     CR,LF,'Disk has been initilized correctly for a (5", 360K IBM-PC) MSDOS (V1.1) disk.',CR,LF,
POSTMOD_ABORTED_MSG:         DB     CR,LF,'Aborted post-format disk modification.',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 V1.3 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_DATA   DB     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_DATA   DB     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

@DRIVE              DB     0H           ;CURRENT DISK DRIVE NUMBER (0,1,2,3)
@SOURCE_DRIVE       DB     0H           ;For Sysgen
@CPM_DRIVE_OFFSET   DB     0H           ;CPM/ZFDC Drive offset
@FORMAT                    DB     0H           ;CURRENT DISK FORMAT NUMBER (0,1,2,3...13H)
@SYS_FORMAT         DB     0H           ;Format for sysgen (in case system tracks are different)
CURRENT_FORMAT_TRACK       DB     0H           ;Store for current track being formatted
@SEC                DB     0H           ;For Sysgen
@TRACK              DB     0H           ;For Sysgen
@SEC_COUNT          DB     0H           ;For Sysgen
@DMA                DW     0H           ;For Sysgen
@END_TRACK          DB     0H           ;For Sysgen

SP_SAVE:            DW     SP_SAVE
                    DS     100H
STACK:     EQU      $
                    DB     'Start of System Tracks Store & Sector Buffer----->'
SECTOR_BUFFER:
TRACK_BUFFER:       DS     14000H       ;Will store system track image here for Sysgen
;END                                    ;Need enough to store two tracks for system copy
```