```
; Z80 Code to run the Z80 Driven WD2793 FDC board (ZFDC Board)
; This code will reside in the on-board 28C256 EPROM at 0H

;       John    Monahan         Started 8/27/2010

;       V0.1    8/27/2010       Initial workup
;       V0.2    8/28/2010       Plugin to CPM Monitor
;       V0.3    9/1/2010        Added WD2793 Code
;       V0.4    9/2/2010        Add disk parameter tables (Both Sec R/W code done)
;       V0.5    9/6/2010        Added the Alt Register set
;       V0.6    9/7/2010        Int's also use [A']
;       V0.7    9/10/2010       Disk track read done
;       V0.8    9/17/2010       Disk Format done
;
;       V1.0    10/3/2010       Jump to core Monitor only (Code rebuilt in stages)
;       V1.1    10/04/2010      Basic CMD's in place. Changed polarity of HW_BYTE bits in drive table & [D']
;       V1.12   10/04/2010      Correct BITS error.
;       V1.13   10/7/2010       Seeks done but returns bit errors
;       V1.14   10/8/2010       Stepin working without errors at 4MHz (Disable INTS)
;       V1.14   10/8/2010       Stepping in with Seek-Verify working (No need to disable INTS)
;       V1.15   10/12/2010      Redo error error routines
;       V1.16   10/12/2010      Get Track ID done using Status Port
;       V1.17   10/12/2010      Random Get Track ID working. Fixed out of range track
;       V1.18   10/12/2010      Random Sector Reads OK
;       V1.19   10/15/2010      Make all routines callable
;       V1.20   10/15/2010      Added re-seeks & re-reads to seek & Read Sector routines. Also new Error reporting
;       V1.21   10/16/2010      New status routines
;       V1.22   10/17/2010      1K sector reads working fine
;       V1.23   10/17/2010      Debug mode added to board display.
;       V1.24   10/17/2010      Fixed Hex Display Roll. Re-did HEX display (again)
;       V1.25   10/19/2010      Sector write done (for 8" disks)
;       V1.26   10/20/2010      Gone to 8MHz clock. Added Bit 7 check of WD port
;       V1.27   10/23/2010      All 8" disk formatting done (side B yet to do).
;       V1.8    10/23/2010      Before adding NMI controls
;       V1.9    10/26/2010      Added routines to handle INTRQ's and Watchdog timer
;       V2.0    10/28/2010      Changed Drive tables, used [DE] as count for Sec R/W, Abort capability for formatting
;       V2.01   11/03/2010      Removed Alt Rag set and went to memory locations for drive format paramaters
;       V2.02   11/04/2010      RAM data dump done
;       V2.03   11/05/2010      Combined select disk & disk format routine
;       V2.04   11/08/2010      Major cleanup. Added ZFDC_Busy handshake fot Get_Data
;       V2.05   11/11/2010      Added block I/O communication interrupt routines for sector read and writes to/from S-100 system
;
;       V2.0    12/07/2010      Configured for V2 prototype Board, Initilization test added
;       V2.1    12/25/2010      Added Error string reporting
;       V2.2    12/27/2010      Added Alive check CMD
;       V2.3    01/28/2011      Removed Timer, added WP Bit
;       V2.4    02/1/2011       Add ability to recover after a board reset without scrubbing memory variables
;       V2.5    02/17/2011      Add check there is a valid drive present CMD (CHECK_VALID_DRIVE)
```

```
;       V2.6    02/17/2011    Fixed problem not detecting WP disks
;       V2.7    02/26/2011    Added multi-sector R/W capability
;       V2.8    02/28/2011    Added faster multi-sector R/W capability (buffering whole tracks)
;       v2.9    04/30/2011    Added support for IBM 1.2M/5" and 1.44M/3.5" disk formats
;       V3.0    05/3/2011     Startup with drive #3 (D:) selected. Prevents possible power-up glitches if disk in A:,B: or C:
;       V3.1    07/18/2011    Changed 1K 8" sector format to 9 sectors/track. This is what I had in the past and what Godbout used.
;       V3.2    08/19/2011    Added multisector R/W capability for MS-DOS. (CPM & MSDOS do sec/track differently)
;       V3.4    08/20/2011    Corrected serious error where I was reading PIO2-Port B, use RAM store
;       V3.5    8/20/2011     Added DOS_SET_SECTOR command. DS disks did not work with CPM style CMD



;       Configuration...
;       EEPROM  from 0 to 7FFFH
;       RAM     from 8000H to FFFFH    Note memory locations are hard wired.

; The board works as follows:- The S-100 BIOS sends one byte commnands to this program. These
; commnads are examined (LOOP:) and a CALL to the appropiate routine follows. More bytes of data
; may (or may not) be expected to come or go to the S-100 system depending on the command.

; When a command is  completed it will usually send a "NO_ERRORS_FLAG" back to the S-100 system.  If there is an error,
; an error byte will be returned instead.

; Note: No effort was made to write compact efficient code. Routines are often repeated for
; simplicity and easy reading. Any optimization (been there, done that!), is for speed rather than code size.
; We have 32K or EPROM space, most will not be used!

; Note Also: It is very important that the BIOS "Disk Paramater Tables" in any S-100 BIOS are EXACTLY the
; same as those here on the ZFDC Board. There can be less disk formats (actully only one is required), but a disk format
; number (FORMAT_NUM) on the S-100 BIOS has to be EXACTLY the same disk format on this ZFDC board. This is because this
; table is used to determine sector size, sectors/track etc.  For this reason whenever a disk is selected the disk format
; must already have been sent (usually one time only).
;
; The two registers [IX] & [IY] are used throughout this "BIOS". [IX] always points to one of the many disk formats available
; in the BIOS. Each is given a unique format number. An unformatted disk has a format number of 0. A format number of 1
; corresponds to a standard 8" single density single sided IBM style disk. (See all tables at the end of this BIOS).

; [IY] always points to the current drive (A,B,C or D) which contains format paramaters for THAT particular drive.
; The Achilles heel of this "BIOS" is if you get [IX] or [IY] values wrong.
;
; In more detail:-
; [IX] always points to a "Disk paramater format table". Set initially to STD8IBM (format #1), an 8" single density
; IBM 26 X 128 byte sectors per track for an 8" drive. A new format is assigned by changing the [IX] pointer.
; VIP,the [IX] pointer offset assigned equates (see below), must not be changed (without changing all the tables).
;
; [IY] always points to a "Disk Drive Table". (For CP/M, this initially will be A:).
; A new format is assigned by copying over of the [IX] pointer values for that format to that drives [IY] drive table.
; Currently there are four [IY] tables for drives A:,B:,C: and D:.  All hardware communication
; and CPM/DOS correspondance (via track,sector,side etc) is through the disk [IY] table pointer+offsets.
```

```
;
; The only exception is for double sided drives to flip head R/W sides where a direct RESET-(Side B) or SET-(Side A)
; of bit 2 (SIDE_BIT,04H) of PIO2_PORT_B is used.
;
;Note in V2.8 and later, support has been added support for IBM 1.2M/5" and 1.44M/3.5" disk formats.
;As far as the WD2793;is concerned these behave as 8" disks. It's just the disk capacity is larger.
;(Only the old 360K type SD & DD, 300RPM, 5" disks need a clock speed adjustment).
;So whenever I referr to 8" disks, I am also including these 1.2M and 1.4M disks.
;


FALSE           EQU     0
TRUE            EQU     NOT FALSE

                                        ;Equates for display on SD Systems Video Board (Used in Monitor mode only)
SCROLL          EQU     01H             ;Set scrool direction UP.
LF              EQU     0AH
CR              EQU     0DH
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

CPM86_FLAG      EQU     01              ;Flag to indicate after 5" disk formating the CPM86 first sector needs to be modified
STATUS_DELAY    EQU     5               ;Time-out for waiting for WD2793 status port to go not busy. (~5 seconds @ 6MHz)

RAM_START       EQU      8000H          ;Location of the start of RAM area on the board
HIGH_RAM_BYTE   EQU         0H
ROM_START       EQU         0H
TOP_OF_RAM      EQU       0F8H          ;For multisector R/W's do not allow sector RAM buffer to go beyond 0F800H
MAX_TRACK_SIZE  EQU      3000H          ;Maximum number of bytes there will be on a track. Used for formatting a disk
                                        ;and reading a disk track

; Note:- The [AF'] alternative Z80 register is used with the two PIO interupt routines.  It is assumed to have exclusive use
; for this function. For efficiency and simplicity no stack is used during the two PIO interrupts (INPUT_INTS & OUTPUT_INTS)
; These interrupts can strike at any time.

; 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.

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 +1 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)



; EQUATES FOR [IY] REGISTER OFFSETS FOR TRACK, SIDE, SECTOR locations for each actual disk drive (A,B,C & D)

DRIVE_FORMAT_TABLE      EQU     0       ;The first 21 bytes of the above current disk format table will be copyed here
DRIVE_PORT              EQU     0       ;Will contain bit flags for:-
                                        ;Bits 0,1 are used for drive hardware selection (00=A, 01=B,10=C,11=D)
                                        ;Bit 2, 1 if side A is selected, 0 if Side B
                                        ;Bit 3, 1 if 8" disk, 0 = 5" disk
                                        ;Bit 4, 1 if Single density, 0 = Double density
                                        ;Bits 5-7 are used by hardware (the same for all disks)
                                                ;Bit 5 WD2793 INTRQ  1=ON, 0=OFF
                                                ;Bit 6 Write protect flag 0 = ON, 1 = OFF
                                                ;Bit 7 ZFDC Direction flag to S-100 system, 1 = ZFDC to S-100, 0 = S-100 to ZFDC

DRIVE_TRACK             EQU     30H     ;CPM/DOS requested track (may not be the same as in the WD2793 track register)
DRIVE_SECTOR            EQU     31H     ;CPM/DOS requested sector (may not be the same as in the WD2793 sector register)
DRIVE_SS_DS_FLAG        EQU     32H     ;0H if a single sided disk, 4H if a double sided disk.
DRIVE_CPM_DRIVE                 EQU     33H             ;Bits (0,1)uses to designate the drive select. (0,1,2,3)
DRIVE_NEW               EQU     34H     ;Need to check there is an actual disk there 0 = No disk. 0FFH if disk checked out.
DRIVE_SIDE              EQU     35H     ;CPM/DOS requested side

DIRECTION_BIT           EQU     7       ;Bits for the above flags 0 = BOARD DATA IN, 1 = BOARD DATA OUT
WP_BIT                  EQU     6       ;Bit to set Write protect LED (D17)
WD_INTRQ_BIT            EQU     5       ;High allows WD2793 INTRQ to trigger NMI's  (V2 Board)
DENSITY_BIT             EQU     4
SIZE_BIT                EQU     3
SIDE_BIT                EQU     2       ;04H
```

```
;       RAM ASSIGNED LOCATIONS FOR CRITICAL VARIBLAES USED THROUGHOUT THE "BIOS"
;We have plenty or RAM, so space things out so they can be easily found with the Monitor if needed.

STACK                   EQU     RAM_START+7FF0H                         ;<---- Stack near top or RAM

FBUFFER                 EQU     RAM_START+2000H                         ;Will build a Track image for formatting here (A000H)in RAM
                                                                        ;Can be up to MAX_TRACK_SIZE

SECTOR_BUFFER           EQU     RAM_START+500H                          ;Sector data will be loaded here (8500H). (Normally up to 1K)
                                                                        ;but it can be up to (### X sector size) for multi sector R/W's

@DRIVE_D_NEW            EQU     RAM_START+400H + DRIVE_NEW      ;Need to check there is an actual disk there 0 = No disk.
@DRIVE_D_CPM_DRIVE      EQU     RAM_START+400H + DRIVE_CPM_DRIVE  ;CPM drive number (3H)
@DRIVE_D_SS_DD_FLAG     EQU     RAM_START+400H + DRIVE_SS_DS_FLAG ;0H if a single sided disk, 4H if a double sided disk
@DRIVE_D_SECTOR                 EQU     RAM_START+400H + DRIVE_SECTOR   ;CPM requested sector
@DRIVE_D_TRACK          EQU     RAM_START+400H + DRIVE_TRACK    ;CPM requested track
                                                                        ;20 Bytes currently. (Leave room for table to be modified later)
@DRIVE_D_DRIVE_TABLE    EQU     RAM_START+400H                          ;The current format table for drive D: will start here

@DRIVE_C_NEW            EQU     RAM_START+300H + DRIVE_NEW        ;Need to check there is an actual disk there 0 = No disk.
@DRIVE_C_CPM_DRIVE      EQU     RAM_START+300H + DRIVE_CPM_DRIVE  ;CPM drive number (2H)
@DRIVE_C_SS_DD_FLAG     EQU     RAM_START+300H + DRIVE_SS_DS_FLAG ;0H if a single sided disk, 4H if a double sided disk
@DRIVE_C_SECTOR                 EQU     RAM_START+300H + DRIVE_SECTOR   ;CPM requested sector
@DRIVE_C_TRACK          EQU     RAM_START+300H + DRIVE_TRACK    ;CPM requested track
                                                                        ;20 Bytes currently. Leave room for table to be modified
@DRIVE_C_DRIVE_TABLE    EQU     RAM_START+300H                          ;The current format table for drive C: will start here

@DRIVE_B_NEW            EQU     RAM_START+200H + DRIVE_NEW      ;Need to check there is an actual disk there 0 = No disk.
@DRIVE_B_CPM_DRIVE      EQU     RAM_START+200H + DRIVE_CPM_DRIVE  ;CPM drive number (1H)
@DRIVE_B_SS_DD_FLAG     EQU     RAM_START+200H + DRIVE_SS_DS_FLAG ;0H if a single sided disk, 4H if a double sided disk
@DRIVE_B_SECTOR                 EQU     RAM_START+200H + DRIVE_SECTOR   ;CPM requested sector
@DRIVE_B_TRACK          EQU     RAM_START+200H + DRIVE_TRACK    ;CPM requested track
                                                                        ;20 Bytes currently. Leave room for table to be modified
@DRIVE_B_DRIVE_TABLE    EQU     RAM_START+200H                          ;The current format table for drive B: will start here

@DRIVE_A_NEW            EQU     RAM_START+100H + DRIVE_NEW      ;Need to check there is an actual disk there 0 = No disk.
@DRIVE_A_CPM_DRIVE      EQU     RAM_START+100H + DRIVE_CPM_DRIVE  ;CPM drive number (0H)
@DRIVE_A_SS_DD_FLAG     EQU     RAM_START+100H + DRIVE_SS_DS_FLAG ;0H if a single sided disk, 4H if a double sided disk
@DRIVE_A_SECTOR                 EQU     RAM_START+100H + DRIVE_SECTOR   ;CPM requested sector
@DRIVE_A_TRACK          EQU     RAM_START+100H + DRIVE_TRACK    ;CPM requested track
                                                                        ;20 Bytes currently. Leave room for table to be modified
@DRIVE_A_DRIVE_TABLE    EQU     RAM_START+100H                          ;The current format table for drive A: will start here

@PIO2B_BITS             EQU     RAM_START+6AH                   ;Because PIO2B is set to output mode we must store its current state here
@DMA_NEW                EQU     RAM_START+68H                   ;DW, For multi-sector R/W (Only)
@DMA                    EQU     RAM_START+64H                   ;DW, Normally points to SECTOR_BUFFER but changes for multi-sector R/W
```

```
@MULTI2_SEC_COUNT      EQU     RAM_START+62H                ;Count for multi-sector R/W's
@MULTI_SEC_COUNT       EQU     RAM_START+61H                ;Count for multi-sector R/W's
@MULTI_FLAG            EQU     RAM_START+60H                ;Falg to indicate this is a multisector read

@OLD_DRIVE_SELECT      EQU     RAM_START+51H                 ;Old selected drive. 0,1,2,3
@CURRENT_DRIVE_SELECT  EQU     RAM_START+50H                 ;Current selected drive. 0,1,2,3

S_DATA_MARK            EQU     RAM_START+40H                ;Pointer to start of Data area for track formatting
E_DATA_MARK            EQU     RAM_START+3CH                ;Pointer to end of Data area+1
E_SEC_MARK             EQU     RAM_START+38H                ;End Sector image +1
S_GAP4_MARK            EQU     RAM_START+34H                ;Start GAP4 area
E_GAP4_MARK            EQU     RAM_START+30H                ;End track +1 (Not used here. Can be used to determine a new minimum total track size)

F_TRK                  EQU     RAM_START+27H                ;for building format track image
F_SIDE                 EQU     RAM_START+26H                ;for building format track image

@RAM_FLAG2             EQU     RAM_START+24H                ;(22H -- 25H), If 00,00,00,00H here, upon a hardware reset, RAM is NOT re-initilized
@RAM_FLAG              EQU     RAM_START+22H                ;Used by CPM BIOS to recover from hardware hang-up's


@SEC_RT                EQU     RAM_START+21H                ;Sector R/W retry count
@SEEK_RT               EQU     RAM_START+20H                ;Seek retry count

@WAIT_FLAG             EQU     RAM_START+1AH
@TEMP_STACK            EQU     RAM_START+18H                ;Tempory stack pointer store (used by Monitor)
@INTR_ADDRESS          EQU     RAM_START+16H                ;WORD, Used by NMI interrrupt routine

@OUT_DATA_FLAG         EQU     RAM_START+14H                ;For SEND_DATA routine
@IN_DATA_FLAG          EQU     RAM_START+13H                ;For GET_DATA routine
@CURRENT_ACK           EQU     RAM_START+12H                ;Most recent ERROR # send back to S-100 system (shown in debug mode)
@CURRENT_CMD           EQU     RAM_START+11H                ;Current command being carried out (shown in debug mode)
@DEBUG_FLAG            EQU     RAM_START+10H                ;Flag to turn HEX displays to debug mode (0H if OFF, 0FFH if ON)

SECTOR_ID_BUFFER       EQU     RAM_START                    ;Sector ID data will be loaded here.


;       ZILOG PIO'S
PIO1_DATA_A    EQU     00000000B      ;PIO Data port to SEND data to S-100 bus
PIO1_DATA_B    EQU     00000010B      ;PIO Data port to GET data from S-100 bus
PIO1_CTL_A     EQU     00000001B      ;PIO CTRL port to Send data to S-100 bus
PIO1_CTL_B     EQU     00000011B      ;PIO CTRL port to Get data from S-100 bus

PIO2_DATA_A    EQU     00000100B      ;PIO Data port for TIL Displays
PIO2_DATA_B    EQU     00000110B      ;PIO Data port for Drive Select,Side,density etc. (Note: Output mode ONLY)
PIO2_CTL_A     EQU     00000101B      ;PIO CTRL port for TIL Displays
PIO2_CTL_B     EQU     00000111B      ;PIO CTRL port for Drive Select,Side,Density,WP and Busy etc.

;       Western Digital WD2793 CHIP PORTS
WD2793_BASE    EQU     00001000B      ;Base Port for WD2793 Chip (Pin 6 of U19A)
```

```
WD2793_STATUS  EQU     WD2793_BASE    ;STATUS PORT (Read)
WD2793_CMD     EQU     WD2793_BASE    ;COMMAND PORT (Write)
WD2793_TRACK   EQU     WD2793_BASE+1  ;TRACK PORT
WD2793_SECTOR  EQU     WD2793_BASE+2  ;SECTOR PORT
WD2793_DATA    EQU     WD2793_BASE+3  ;DATA PORT


;Western Digital Chip 2793 Commands:-
;The 2793 chip for most commands used bits 0&1 to set the head
;motor stepping rate. 00 being 3ms (@2MH clock, to chip),11 being 15 ms.
;My Tandon 8" drives take the fastest rate. Older drives may not.

RSCMD          EQU     00001000B      ;(0CH), RESTORE CMD   <--- (Some drives require a slower stepping rate r1,r0)
                                      ; Note the V bit (bit 3) is 0, sometimes this CMD will be used to format a disk
SKNCMD         EQU     00011000B      ;(18H), SEEK NO VERIFY CMD <--
SKCMD          EQU     00011100B      ;(1CH), SEEK WITH VERIFY CMD <---
RDACMD         EQU     11000100B      ;(C0H), READ TRACK/SECTOR ID CMD
STEPIN         EQU     01001000B      ;(A8H), Step-in NO verify on destination track, NO update TRK register
STEPOUT        EQU     01111000B      ;(E8H), Step-out NO verify on destination
RDCMD93        EQU     10000000B      ;(80H), READ SECTOR CMD       WD2793 chip
WRCMD93        EQU     10100000B      ;(A0H), WRITE SECTOR CMD
WRTCMD         EQU     11110100B      ;(F4H),Write a whole track command
RDTCMD         EQU     11100100B      ;(E4H),Read a whole track command


;ERROR Code masks for the WD2793 Status Register.

SECTOR_RETRY_MAX EQU   4              ;Number of times to try R/W a sector before returning an error
SEEK_RETRY_MAX EQU     2              ;Number of times to try seek a track before returning an error
HOME_ERR_MASK   EQU    80H            ;Error mask for Type I Home CMD
SIN_ERR_MASK   EQU     90H            ;Step head in one track command error bits
SOUT_ERR_MASK  EQU     90H            ;Step head out one track command error bits
SK_ERR_MASK     EQU    90H            ;Track Seek error bits
ID_ERR_MASK     EQU    9FH            ;Sector ID read error mask
RS_ERR_MASK     EQU    0BFH           ;Read sector data error mask
MRS_ERR_MASK    EQU    0AFH           ;Multi-sector Read data error mask (not currently used)
WS_ERR_MASK    EQU     0EFh           ;Write sector data error mask
MWS_ERR_MASK   EQU     0EFh           ;Multi-sector Write data error mask (not currently used)
RT_ERR_MASK     EQU    80H            ;Read Track error mask
WT_ERR_MASK     EQU    0E0H           ;Write Track error mask


;Commands to the ZFDC Board:-

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

```
CMD_SET_FORMAT EQU     4H              ;This will select a specified drive and assign a disk format table to that drive
CMD_SET_DRIVE  EQU     5H              ;This will select a specified drive (0,1,2,3)
CMD_GET_FORMAT EQU     6H              ;Return to S100 System the current Disk paramater format table number
CMD_SET_TRACK  EQU     7H              ;This will set head request to a specified track
CMD_SET_SIDE   EQU     8H              ;This will set side request to a specified side
CMD_SET_SECTOR EQU     9H              ;This will set sector request to a specified sector

CMD_SET_HOME   EQU     0AH             ;This will set head request to Track 0 of CURRENT drive
CMD_STEP_IN    EQU     0BH             ;Step head in one track of CURRENT drive
CMD_STEP_OUT   EQU     0CH             ;Step head out one track of CURRENT drive
CMD_SEEK_NV    EQU     0DH             ;Seek to track with NO verify of CURRENT drive

CMD_SEEK_TRACK EQU     0EH             ;Seek to track to (IY+DRIVE_TRACK) with the track verify bit set on CURRENT drive/format
CMD_GET_TRACK_ID  EQU  0FH             ;Read the CURRENT TRACK ID

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

CMD_GET_WD_TRACK  EQU  12H             ;Get the WD2793 Track register value
CMD_GET_WD_SECTOR EQU  13H             ;Get the WD2793 Sector register value
CMD_GET_WD_STATUS EQU  14H             ;Get the WD2793 Status register value

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

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

CMD_ABORT      EQU     20H             ;Generalized Abort of the current process command.
CMD_HANDSHAKE  EQU     21H             ;Handshake command only sent during board initilization/testing
CMD_GET_DRIVE  EQU     22H             ;Get the current selected drive number (0..3)
CMD_SET_TRACK_DS EQU   23H             ;Set Track (If a DS Disk, EVEN tracks on Side A, ODD tracks on Side B. Used by CPM)
CMD_GET_ERROR_STRING EQU 24H           ;Return a string explaining the an Error Code previously 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                ;CPM, Read data from multiple sectors starting at the CURRENT sector
CMD_WR_MULTI_SECTOR     EQU 2AH                ;CPM, Write data to multiple sectors starting at the CURRENT sector
CMD_DOS_RD_MULTI_SEC    EQU 2BH                ;MS-DOS, Read data from multiple sectors starting at the CURRENT sector.
CMD_DOS_WR_MULTI_SEC    EQU 2CH                ;MS-DOS, Write data to multiple sectors starting at the CURRENT sector.
CMD_GET_SIDE           EQU 2DH         ;Get the current selected side (0=A/1=B) of the current selected drive
CMD_DOS_SET_SECTOR     EQU 2EH         ;MS-DOS, Set current sector to be R/W

;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 valid 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
BUFFER_OVERFLOW EQU    3DH             ;Too many sectors were requested in a multi sector R/W request

CMD_RANGE_ERR  EQU     3DH             ;CMD out or range

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



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

        ORG     ROM_START              ;The EEPROM code will start here

        DI                             ;Just in case
        IM2                            ;Set Z80 to Interrupt mode 2
        XOR     A,A
        LD      I,A                    ;Set Interrupt Reg to 0H for PIO's below.

        LD      HL,(@RAM_FLAG)         ;If this location has 0000H in RAM, then NOT a cold start reset.
        LD      A,L                    ;Assumes that the (1/64K X 1/64K) chance 00,00,00,00H will not occour on power-on!
        OR      A,H
        JP      NZ,HARD_RESET          ;If NZ then do a complete hardware reset.
        LD      HL,(@RAM_FLAG2)               ;If this location has 0000H in RAM, then NOT a cold start reset.
        LD      A,L
        OR      A,H
        JP      NZ,HARD_RESET          ;If NZ then do a complete hardware reset.
        JP      BEGIN                  ;If 00,00,00,00H then skip RAM initilization
```

```
HARD_RESET:
        LD      HL,RAM_START            ;Clear ALL RAM.
RAM0:   XOR     A
        LD      (HL),A                  ;Zero RAM
        INC     HL
        LD      A,H
        CP      A,HIGH_RAM_BYTE                 ;Clear to Top of RAM (Note Stack not yet setup)
        JR      NZ,RAM0

        LD      HL,RAM_START            ;Check RAM.
RAM1:   LD      A,(HL)                  ;Is it 0
        OR      A                       ;Zero RAM
        JR      NZ,RAM_PROBLEM
        INC     HL
        LD      A,H
        CP      A,HIGH_RAM_BYTE                 ;Clear to Top of RAM (Note Stack not yet used)
        JR      NZ,RAM1

        LD      SP,STACK                ;Now setup a valid stack on the boards RAM (@ 0FFF0H)
        JP      BEGIN                   ;Jump Over INT Locations


        ORG     38H                     ;Z80 will also jump here if no RAM (FF's)
RAM_PROBLEM:
        LD      A,11010000B             ;To lower strobe for TIL 1 (Must be < 8 Bytes!)
        OUT     PIO2_DATA_B,A
        OUT     PIO2_DATA_A,A           ;Flag with an 'D' ion TIL display
        HALT                            ;In case bad memory

        ORG     40H                     ;<<< Int Location for PIO#1 S100 Port B INPUT
        DW      INPUT_INTS              ;Location of Input INT routine
        HALT


        ORG     66H                     ;<<< Location of NMI. Any Time the 74LS123 times OUT it will trigger a jump to here
NMI_INT:                                ;if "ARMED", also WD2793 IOREQ's do the same thing if ARMED
        LD      HL,(@INTR_ADDRESS)      ;Jump to the interrupt routine required for that timeout condition
        EX      (SP),HL
        RETN                            ;Jump to new location
        HALT


        ORG     80H                     ;<<< Int Location for PIO#1 S100 Port A OUTPUT
        DW      OUTPUT_INTS             ;Location of Output INT routine
        HALT


DEFAULT_NMI:                            ;For debugging, trap any unwanted NMI's here
        LD      A,0EEH
```

```
        CALL    DisplayHEX_01           ;Put EEE0 on the HEX display
        LD      A,0E0H
        CALL    DisplayHEX_23
        HALT


        ORG     100H
BEGIN:                                  ;First setup the two Zilog PIO's
        LD      A,4FH                   ;Mode 1 (Input) for PIO #1 Port B
        OUT     (PIO1_CTL_B),A
        LD      A,40H                   ;Setup Interrupt vector to 40H in RAM
        OUT     (PIO1_CTL_B),A
        LD      A,10000111B             ;Allow Interrupt Control of this port
        OUT     (PIO1_CTL_B),A
        LD      A,00000011B             ;Disable for now
        OUT     (PIO1_CTL_B),A

        LD      A,0FH                   ;Mode 1 (Output) for PIO #1 Port A
        OUT     (PIO1_CTL_A),A
        LD      A,80H                   ;Setup Interrupt vector to 80H in RAM
        OUT     (PIO1_CTL_A),A
        LD      A,10000111B             ;Allow Interrupt Control of this port
        OUT     (PIO1_CTL_A),A
        LD      A,00000011B             ;Disable for now
        OUT     (PIO1_CTL_A),A

                                        ;Now setup & test PIO #2
        LD      A,0FH                   ;Mode 0 (Output) for PIO #2 Port A
        OUT     (PIO2_CTL_A),A
        LD      A,03H                   ;Reset the INT enable FlipFlop (not required for this port)
        OUT     (PIO2_CTL_A),A

        LD      A,0FH                   ;Mode 0 (Output) for PIO #2 Port B
        OUT     (PIO2_CTL_B),A
        LD      A,03H                   ;Reset the INT enable FlipFlop (not required for this port)
        OUT     (PIO2_CTL_B),A
        LD      A,01011011B             ;Set default to NOT BUSY, No WP, No WD_INTRQ, Single density, 8" drive, Side A, Drive D
        OUT     PIO2_DATA_B,A
        LD      (@PIO2B_BITS),A                 ;Store current status because this port cannot be read

        CALL    OFF_WD2793_INTRQ        ;Prevent any WD2793 INT's

        XOR     A,A
        LD      (@DRIVE_A_CPM_DRIVE),A          ;CPM/Drive select bits for drive A:
        INC     A
        LD      (@DRIVE_B_CPM_DRIVE),A          ;CPM/Drive select bits for drive B:
        INC     A
        LD      (@DRIVE_C_CPM_DRIVE),A          ;CPM/Drive select bits for drive C:
        INC     A
```

```
        LD      (@DRIVE_D_CPM_DRIVE),A        ;CPM/Drive select bits for drive D:

INITILIZE:
        LD      A,10000011B           ;Enable Interrupts on PIO#1 (S100 I/O)
        OUT     (PIO1_CTL_A),A
        OUT     (PIO1_CTL_B),A
        EI                            ;Enable Interrupts

        LD      A,0D0H                ;Note: This will stop any current WD2793 chip process
        OUT     (WD2793_CMD),A
        XOR     A,A
        OUT     (WD2793_TRACK),A
        INC     A
        OUT     (WD2793_SECTOR),A


INITILIZE_RESET:                      ;A jump here will reset the board software
        DI                            ;Do not INT's
        XOR     A,A
        LD      (@IN_DATA_FLAG),A
        LD      (@DEBUG_FLAG),A               ;Start with HEX LED debug display mode off
        CALL    DIRECTION_IN          ;Set ZFDC Board to recieve data (default mode)
        CALL    DisplayHEX_01         ;Set to 00 for now
        CALL    DisplayHEX_23         ;Set to 00 for now

        LD      A,CMD_HANDSHAKE               ;Send to flag to let S-100 System know we are alive
        OUT     (PIO1_DATA_A),A               ;Put it in output port (Int is disabled)

        LD      B,0
WAIT_FOR_HANDSHAKE:                   ;Wait until S-100 system signs on.
        EI                            ;Allow INT's
        LD      A,(@IN_DATA_FLAG)     ;<<<<<<<<< LOOP IF NOTHING >>>>>>>>>
        OR      A,A
        JR      NZ,HANDSHAKE1         ;Loop until we get something (>>> Add watchdog later <<<<)
        LD      A,B
        INC     B
        CALL    DisplayHEX_01         ;Show we are waiting
        CALL    DELAY_200mS
        JR      WAIT_FOR_HANDSHAKE

HANDSHAKE1:
        CALL    GET_DATA              ;Show actual data byte recieved
;       CALL    DisplayHEX_23         ;Show data
;       CP      A,CMD_HANDSHAKE
        JR      NZ,WAIT_FOR_HANDSHAKE ;Not a CMD_HANDSHAKE, ther go back and wait some more
        LD      C,NO_ERRORS_FLAG      ;Yes, then send an ACK
        CALL    SEND_DATA             ;back to the S-100 System

START1: LD      IX,UNFORMATTED        ;Default drive table = Unformatted
```

```
        LD      IY,@DRIVE_A_DRIVE_TABLE       ;Note no drive has been made the current drive yet. (IY+DRIVE_PORT) = 0
        LD      B,1                    ;Set to IBM 8"
        LD      A,0
        CALL    CORE_SET_FORMAT               ;Select drive. Drive in [A], format in [B] (default IBM 8"), Update Drive TIL display

WARM_RESET:
        LD      A,0FFH                 ;Set so no current drive selected by CPM
        LD      (@CURRENT_DRIVE_SELECT),A

        CALL    DisplayTrack           ;Will have invalid TRK/SEC values to indicate we have not yet selected a drive
        CALL    DisplaySector          ;Will show 0FF00H

;<<<<<<<<<<<<<<<<<<<<<<<<<<< MAIN SYSTEM LOOP >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

; Each of the above routines when done will return back here to LOOP

LOOP:   CALL    GET_DATA               ;Get CMD in [A]
        LD      (@CURRENT_CMD),A       ;Save in case we wish to show command in debug mode

        LD      A,(@DEBUG_FLAG)
        OR      A,A
        CALL    NZ,DisplayCMD          ;If in debug mode, display current command first

        LD      A,(@CURRENT_CMD)       ;Save in case we wish to show command in debug mode
                                       ;Put most used commands first
        CP      A,CMD_SET_TRACK               ;7, This will set head request to a specified track. Track Byte (0-NTRKS)
        PUSH    AF
        CALL    Z,SET_TRACK
        POP     AF
        JP      Z,LOOP

        CP      A,CMD_SET_SECTOR       ;9, This will set request to a specified sector. For CPM & CPM86, Sector Byte (0-NSCTRS*2)
        PUSH    AF
        CALL    Z,SET_SECTOR
        POP     AF
        JP      Z,LOOP

        CP      A,CMD_DOS_SET_SECTOR   ;2EH, This will set request to a specified sector. For MS-DOS, Sector Byte (0-NSCTRS)
        PUSH    AF
        CALL    Z,DOS_SET_SECTOR
        POP     AF
        JP      Z,LOOP

        CP      A,CMD_READ_SECTOR      ;10, Read data from the current sector (on current track,side,drive).
        PUSH    AF
        CALL    Z,READ_SECTOR
        POP     AF
        JP      Z,LOOP
```

```
CP      A,CMD_WRITE_SECTOR     ;11, Write data to the current sector (on current track,side,drive)
PUSH    AF
CALL    Z,WRITE_SECTOR
POP     AF
JP      Z,LOOP

CP      A,CMD_SET_DRIVE               ;5, This will set current the Disk drive number
PUSH    AF
CALL    Z,SET_DRIVE
POP     AF
JP      Z,LOOP

CP      A,CMD_CHECK_DRIVE      ;28H, This will set current the Disk drive number
PUSH    AF
CALL    Z,CHECK_VALID_DRIVE
POP     AF
JP      Z,LOOP

CP      A,CMD_SET_HOME         ;0A, This will set head request to Track 0 of current drive
PUSH    AF
CALL    Z,SET_HOME             ;Note drive head(s) will be physically restored to track 0
POP     AF
JP      Z,LOOP

CP      A,CMD_SET_FORMAT       ;4, This will set request to a specified drive. Drive Byte (0-3)
PUSH    AF
CALL    Z,SET_FORMAT
POP     AF
JP      Z,LOOP

CP      A,CMD_RD_MULTI_SECTOR         ;29H, Multi-sector Read data from the current sector (on current track,side,drive).
PUSH    AF
CALL    Z,READ_MULTI_SECTOR
POP     AF
JP      Z,LOOP

CP      A,CMD_WR_MULTI_SECTOR         ;2AH, Multi-sector Write data from the current sector (on current track,side,drive).
PUSH    AF
CALL    Z,WRITE_MULTI_SECTOR
POP     AF
JP      Z,LOOP


CP      A,CMD_DOS_RD_MULTI_SEC        ;2BH, Multi-sector Read data from the current sector (on current track,side,drive).
PUSH    AF
CALL    Z,DOS_READ_MULTI_SECTOR
POP     AF
JP      Z,LOOP
```

```
CP      A,CMD_DOS_WR_MULTI_SEC        ;2CH, Multi-sector Write data from the current sector (on current track,side,drive).
PUSH    AF
CALL    Z,DOS_WRITE_MULTI_SECTOR
POP     AF
JP      Z,LOOP

CP      A,CMD_GET_ERROR_STRING        ;24H    Send string explaining last error code if requested
PUSH    AF
CALL    Z,SEND_ERRORS_STRING
POP     AF
JP      Z,LOOP

CP      A,CMD_SET_SIDE                ;8, This will set head request to a specified side. Side Byte (0=A, 1=B)
PUSH    AF
CALL    Z,SET_SIDE
POP     AF
JP      Z,LOOP

CP      A,CMD_GET_SIDE                ;2DH, Get the currently selected side of the CURRENT drive  (0=A, 1=B)
PUSH    AF
CALL    Z,GET_SIDE
POP     AF
JP      Z,LOOP


                            ;Less Frequently used commands....

CP      A,CMD_ZFDC_ALIVE      ;26H, This command is sent to check teh ZFDC board is present and active/initilized
PUSH    AF
CALL    Z,ZFDC_ALIVE
POP     AF
JP      Z,LOOP

CP      A,CMD_GET_SEC_SIZE    ;25H, This will return the sector size of the current disk
PUSH    AF
CALL    Z,GET_SEC_SIZE
POP     AF
JP      Z,LOOP

CP      A,CMD_GET_SEC_COUNT   ;26H, This will return the sector size of the current disk
PUSH    AF
CALL    Z,GET_SEC_COUNT
POP     AF
JP      Z,LOOP

CP      A,CMD_GET_FORMAT      ;6, This will return the Disk Format table number
PUSH    AF
CALL    Z,GET_FORMAT
POP     AF
```

```
JP      Z,LOOP

CP      A,CMD_SEEK_TRACK       ;0E, Seek to track with verify of current drive (seperate side select)
PUSH    AF
CALL    Z,SEEK_TRACK
POP     AF
JP      Z,LOOP

CP      A,CMD_SET_TRACK_DS     ;23H, Special set track for current drive (EVEN tracks Side A, ODD tracks side B)
PUSH    AF
CALL    Z,SET_TRACK_DS
POP     AF
JP      Z,LOOP

CP      A,CMD_STEP_IN          ;0B, Step head in one track of current drive
PUSH    AF
CALL    Z,STEP_IN
POP     AF
JP      Z,LOOP

CP      A,CMD_STEP_OUT         ;0C, Step head out one track of current drive
PUSH    AF
CALL    Z,STEP_OUT
POP     AF
JP      Z,LOOP

CP      A,CMD_SEEK_NV          ;0D, Seek to track with NO verify of current drive
PUSH    AF
CALL    Z,SEEK_NV
POP     AF
JP      Z,LOOP

CP      A,CMD_GET_TRACK_ID     ;0F, Read the current track ID (6 Bytes of data returned)
PUSH    AF
CALL    Z,GET_TRACK_ID
POP     AF
JP      Z,LOOP


CP      A,CMD_RESET_ZFDC       ;3, Reset the WD2793 chip AND board software & hardware
PUSH    AF
CALL    Z,RESET_ZFDC
POP     AF
JP      Z,LOOP

CP      A,CMD_GET_WD_TRACK     ;12, Return the WD2793 Track value
PUSH    AF
CALL    Z,GET_WD_TRACK
POP     AF
```

```
JP      Z,LOOP

CP      A,CMD_GET_WD_SECTOR    ;13, Return the WD2793 Sector value
PUSH    AF
CALL    Z,GET_WD_SECTOR
POP     AF
JP      Z,LOOP

CP      A,CMD_GET_WD_STATUS    ;14, Return the WD2793 Status value
PUSH    AF
CALL    Z,GET_WD_STATUS
POP     AF
JP      Z,LOOP

CP      A,CMD_GET_DRIVE               ;22H, This will return the Disk Format table number
PUSH    AF
CALL    Z,GET_DRIVE
POP     AF
JP      Z,LOOP

CP      A,CMD_TRACK_DUMP       ;15, Get a complete track
PUSH    AF
CALL    Z,READ_TRACK
POP     AF
JP      Z,LOOP

CP      A,CMD_FORMAT_TRACK     ;16, Format the current disk (using a disk paramater format table number)
PUSH    AF
CALL    Z,FORMAT_TRACK
POP     AF
JP      Z,LOOP

CP      A,CMD_PIO_TEST         ;0, Hardware loop test of PIO
PUSH    AF
CALL    Z,PIO_TEST
POP     AF
JP      Z,LOOP

CP      A,CMD_SHOW_SIGNON      ;2, This will Roll Hex digits in Track Display TIL's as a test
PUSH    AF
CALL    Z,SHOW_SIGNON
POP     AF
JP      Z,LOOP

CP      A,CMD_SET_DEBUG_ON     ;17H, Turn on HEX LED Debug display mode
PUSH    AF
CALL    Z,SET_DEBUG_ON
POP     AF
JP      Z,LOOP
```

```
        CP      A,CMD_SET_DEBUG_OFF    ;18H, Turnd off HEX LED Debug display mode
        PUSH    AF
        CALL    Z,SET_DEBUG_OFF
        POP     AF
        JP      Z,LOOP

        CP      A,CMD_RAM_DUMP         ;19H    Dump memory variables to S-100 system
        PUSH    AF
        CALL    Z,RAM_DUMP
        POP     AF
        JP      Z,LOOP

        CP      A,CMD_MONITOR          ;1, No extra data (Note in this case no NO_ERRORS  flag is sent back to CPM system)
        JP      Z,MONITOR              ;Note a Jump not a call

        LD      C,CMD_RANGE_ERR               ;Flag Error #
        CALL    SEND_DATA              ;Send an error flag back to S-100 System
        JP      LOOP



;---------------------------THE SYSTEM COMMAND ROUTINES --------------------------------------------------------------

;Simple test of PIO #1 (Ports A & B). Gets a character from S-100 system & returns it back.

PIO_TEST:
        XOR     A,A
        CALL    DisplayHEX_01          ;Set to 00 for now
        CALL    DisplayHEX_23          ;Set to 00 for now
PIO_TEST1:
        CALL    GET_DATA               ;Get one byte from S100 system (via interrupt mechanism)
        CP      A,ESC                  ;Abort if ESC
        JR      Z,PIO_TEST2
        PUSH    AF
        CALL    DisplayHEX_23          ;Show input character directly on sector TIL's
        POP     AF
        AND     7FH                    ;Strip high bit just in case errors (messes up VDB 8024 CRT display board)
        LD      C,A                    ;Need to send character in [C]
        CALL    SEND_DATA              ;Send it back to S-100 System (via interrupt mechanism)
        JR      PIO_TEST1              ;Repeat until ESC
PIO_TEST2:
        CALL    DisplayTrack
        CALL    DisplaySector
        RET


;-------------- Show simple signon message on Track Display LED's to show everything is OK.
```

```
SHOW_SIGNON:                        ;CMD = 2
        CALL    CORE_ROOL_HEX_DISPLAY
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE    ;Send ACK back to S-100

CORE_ROOL_HEX_DISPLAY:              ;A callable "self contained" routine also use elsewhere
        XOR     A,A
        INC     A
SHOW1:  CALL    DisplayHEX_23       ;This will bypass the reading of the Sector registed
        CALL    DELAY_100mS
        INC     A
        JR      NZ,SHOW1
        CALL    DisplayTrack        ;Put back current WD2793 Track & Sector
        CALL    DisplaySector
        XOR     A                   ;Return Z Flag set = No Errors
        RET


;---------------- FORCE COMPLETE BOARD RESET ON NEXT OUTPUT TO S-100 RESET PORT (Usually 13H)

RESET_ZFDC:                         ;CMD = 3
        LD      HL,0FFFFH
        LD      (@RAM_FLAG),HL      ;If this location has 00,00 in RAM, then not a cold start reset.
                                    ;With 0FFFFH there, then a hardware reset will initilize everything.
        LD      C,NO_ERRORS_FLAG
        CALL    SEND_DATA
        RET


CORE_RESET_2793_ROUTINE:            ;A callable "self contained" routine also use elsewhere
        PUSH    AF
        LD      A,0D0H              ;Note: This will stop any current WD2793 chip process
        OUT     (WD2793_CMD),A
        CALL    DELAY_30uS
        POP     AF
        RET


;--------- Check the ZFDC board is initilized and active. Returns to S-100 system an inverted copy of data byte sent

ZFDC_ALIVE:                         ;CMD = 26H
        CALL    GET_DATA            ;Normally we send 0FH and return F0H
        CPL                         ;Invert it and send it back
        LD      C,A                 ;Remember data always returned from [C]      (most common code error!)
        CALL    SEND_DATA
        RET

;------------ Get current drive paramater format table number (0,1,2...13H..)
```

```
GET_FORMAT:                         ;CMD = 6
        LD      C,(IY+FORMAT_NUM)
        CALL    SEND_DATA           ;Send current table number in [C]
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE    ;Send ACK back to S-100


;---------------- Get current drive number. Returns to S-100 system 0,1,2 or 3

GET_DRIVE:                          ;CMD = 22H
        LD      C,(IY+DRIVE_CPM_DRIVE) ;CPM bits for the drive. Remember always SEND_DATA in [C]
        CALL    SEND_DATA
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE    ;Send ACK back to S-100


;---------------- Get current drive selected side. Returns to S-100 system 0,1
;       [A],   0 = A, 1 = B For software flags
;       Bit 2, 1 = A, 0 = B For hardware bit

GET_SIDE:                           ;CMD = 2DH
        LD      A,(@PIO2B_BITS)             ;Get store hardware select bits (because PIO2-Port B cannot be read)
                                    ;(Note hardware bit is the opposite to the software flags returned)

        BIT     SIDE_BIT,A          ;Get Bit 2 (04H = Hardware Side A, 0=B)
        JR      Z,IS_SIDE_B
        LD      C,0                 ;Send back 0 for software flag A side currently selected
        JR      GET_SIDE1
IS_SIDE_B:
        LD      C,1                 ;Send back 1 = for software flag B side      currently selected
GET_SIDE1:
        CALL    SEND_DATA           ;Remember always SEND_DATA in [C]
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE    ;Send ACK back to S-100



;---------------- Get current disk's sector size. Returns to S-100 system 0,1,2 or 3... (X*128)

GET_SEC_SIZE:                       ;CMD = 25H
        LD      C,(IY+SEC_SIZE_FLAG)   ;Sec size for this drive. Remember always SEND_DATA in [C]
        CALL    SEND_DATA
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE    ;Send ACK back to S-100


;---------------- Get current disk's (sectors/track +1)

GET_SEC_COUNT:                      ;CMD = 25H
```

```
        LD      C,(IY+NSCTRS)           ;Sec/track+1 size for this drive. Remember always SEND_DATA in [C]
        CALL    SEND_DATA
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100


;---------------- Set current drive number. (0,1,2 or 3) Note: NO hardware/status check is done here

SET_DRIVE:                              ;CMD = 5
        CALL    GET_DATA                ;First get the drive select byte in [A]
        CP      A,4
        LD      C,DRIVE_ERR
        JP      NC,S100_ACK_ROUTINE     ;Drive number not 0-3

        CALL    CORE_SET_DRIVE          ;Drive in [A]

        LD      C,NO_ERRORS_FLAG        ;Will always return valid
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100


                                        ;Note NO hardware/status check is done here
CORE_SET_DRIVE:                                 ;[A] contains the drive select (A:=0, B:=1, C:=2 D:=3)
        LD      B,A                     ;Store, so we can add in below
        LD      A,(IY+DRIVE_CPM_DRIVE) ;Get the drive hardware select bits bits
        CP      A,B
        RET     Z                       ;Same drive ?

        LD      A,B
        LD      (@CURRENT_DRIVE_SELECT),A ;Save it for next time
        LD      HL,@DRIVE_A_DRIVE_TABLE
        LD      C,0
        ADD     HL,BC                   ;so:- 0,100H,200H,300H
        PUSH    HL
        POP     IY                      ;[IY] now points to that's drives RAM/Disk Table of variables

        LD      C,(IY+DRIVE_PORT)       ;Get the drives hardware density,size,sides,and select bits
        LD      A,(@PIO2B_BITS)                 ;Get store hardware select bits (because this port cannot be read)
        AND     A,11100000B
        SET     WP_BIT,A                ;Default no WP
        OR      A,C                     ;combine
        LD      B,A                     ;Store here
        IN      A,(WD2793_STATUS)       ;Check for disk write protect flag
        BIT     6,A
        LD      A,B                     ;get back info
        JR      Z,NOT_WP
        RES     WP_BIT,A                ;WP ON so lightup LED D17
NOT_WP: OUT     (PIO2_DATA_B),A                 ;Send to drive select port (Note this will also update TIL display)
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)
```

```
        XOR     A                       ;Invalidate WD2793 current Track/Sec Register
        OUT     (WD2793_SECTOR),A       ;This forces the WD2793 to reseek and get a valid head position
        DEC     A                       ;For the next sector read/write
        OUT     (WD2793_TRACK),A
        RET


;------ Check there is a current drive is valid, head can load etc.  Will do a track seek test-----------------------------

CHECK_VALID_DRIVE:                      ;CMD = 28H
        CALL    GET_DATA                ;First get the drive select byte in [A]
        CP      A,4
        LD      C,DRIVE_ERR
        JP      NC,S100_ACK_ROUTINE     ;Drive number not 0-3

        CALL    CORE_SET_DRIVE          ;Drive in [A]

        IN      A,(WD2793_Status)       ;Clear  FDC status

        XOR     A,A
        OUT     (WD2793_DATA),A                ;Put required track# in data register
        LD      A,SKCMD                 ;Then send a Seek with Verify CMD
        OUT     (WD2793_CMD),A          ;Send seek cmd to current drive

        CALL    WD2793_WAIT             ;Wait 30uS, then check status and return

        IN      A,(WD2793_Status)       ;Get FDC status
        BIT     2,A                     ;If we have a valid drive with head loaded we should see TRK 0 bit:-
        JP      Z,NO_DISK               ;x,x,x,x,no CRC error,Track 0,Index,not busy. for status bits

        LD      A,01                    ;Step in one track
        OUT     (WD2793_DATA),A                ;Put required track# in data register
        LD      A,SKCMD                 ;Then send a Seek with Verify CMD
        OUT     (WD2793_CMD),A          ;Send seek cmd to current drive

        CALL    WD2793_WAIT             ;Wait 30uS, then check status and return

        IN      A,(WD2793_Status)       ;Get FDC status
        BIT     2,A                     ;If we have a valid drive with head loaded we should see NO TRK 0 bit:-
        JP      NZ,NO_DISK              ;x,x,x,x,no CRC error,Track 0,Index,not busy. for status bits

        LD      A,RSCMD                 ;Restore to track 0
        OUT     (WD2793_CMD),A
        CALL    WD2793_WAIT             ;Wait 30uS, then check status and return

        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100

NO_DISK:BIT     5,A                     ;There is a disk, but the door is not closed
```

```
        JP      Z,NO_HEAD               ;Head could not load
        LD      A,RSCMD                 ;Restore to track 0
        OUT     (WD2793_CMD),A

        CALL    WD2793_WAIT             ;Wait 30uS, then check status and return

        LD      C,DRIVE_INACTIVE        ;Will return Drive Inactive error code
        JP      S100_ACK_ROUTINE

NO_HEAD:LD      A,RSCMD                 ;Restore to track 0
        OUT     (WD2793_CMD),A

        CALL    WD2793_WAIT             ;Wait 30uS, then check status and return

        LD      C,DRIVE_DOOR            ;Will return Drive Door Open error code
        JP      S100_ACK_ROUTINE



;-------------------  SET DRIVE FORMAT ----------------------------------
; This routine requires TWO parameters.  The first is the disk drive hardware select
; value (0,1,2,3). The second (in [B] is the DISK PARAMARER TABLE FORMAT NUMBER (0,1,2....13H).
; Note this is a simple but critical routine. Do not change it without extreme care.
;
;Note: When finished this drive is now the new @CURRENT_DRIVE_SELECT

SET_FORMAT:
        LD      A,(@CURRENT_DRIVE_SELECT)
        LD      (@OLD_DRIVE_SELECT),A  ;in case bad request below

        CALL    GET_DATA                ;First get the drive select byte in [A]
        LD      (@CURRENT_DRIVE_SELECT),A     ;save for future

        CALL    GET_DATA                ;Then get the second paramater (disk format )in [A]
        CP      A,DPL_COUNT             ;Is table valid
        JR      NC,BAD_TABLE
        LD      B,A                     ;Store as second paramater (Format #) for CORE_SET_FORMAT

        LD      A,(@CURRENT_DRIVE_SELECT) ;is it valid
        CP      A,4
        JR      NC,BAD_DRIVE_NUMBER

        CALL    CORE_SET_FORMAT                 ;Drive in [A], format in [B] (No error returns)
                                        ;Should not be needed if software is OK!
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE        ;Send all OK, ACK back to S-100

BAD_DRIVE_NUMBER:
        LD      A,(@OLD_DRIVE_SELECT)  ;Get back old value
```

```
        LD      (@CURRENT_DRIVE_SELECT),A
        LD      C,DRIVE_ERR             ;Drive number not 0-3
        JP      S100_ACK_ROUTINE        ;Send error ACK back to S-100

BAD_TABLE:
        LD      C,TABLE_ERR             ;Also default error returned
        JP      S100_ACK_ROUTINE        ;Send error ACK back to S-100



CORE_SET_FORMAT:                        ;[A] contains the drive select (A:=0, B:=1, C:=2 D:=3)
        PUSH    BC                      ;[B] contains new format table number
        LD      HL,@DRIVE_A_DRIVE_TABLE
        LD      B,A                     ;Add in so:- 0,100H,200H,300H
        LD      C,0
        ADD     HL,BC
        PUSH    HL
        POP     IY                      ;[IY] now points to that's drives RAM/Disk Table of variables

        POP     BC                      ;Now get format table number
        LD      A,B                     ;Get format table number
        ADD     A,A                     ;X2
        LD      HL,DPL_POINTERS                 ;Pointers to table list are here
        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)                  ;Low byte first
        INC     HL
        LD      D,(HL)                  ;High byte
        PUSH    DE                      ;[DE]->[IX]
        POP     IX                      ;IX now points to correct Drive table

        LD      B,21                    ;Size (21 decimal) of each DISK FORMAT PARAMATER TABLE (less title)
        PUSH    IY                      ;Save IY
        PUSH    IX                      ;Also IX

MOVE_TBL:
        LD      A,(IX+0)                ;Splice in the new format table values for this drive
        LD      (IY+0),A
        INC     IX
        INC     IY
        DJNZ    MOVE_TBL

        POP     IX                      ;pointers back to the start of both tables
        POP     IY

        LD      A,(IX+HW_BYTE)          ;Isolate the Single/Double density bit
        AND     A,00000100B
        LD      (IY+DRIVE_SS_DS_FLAG),A         ;Store it as 0 (= SS drive) or 4H (= DS drive)
```

```
        LD      C,(IY+DRIVE_CPM_DRIVE) ;First get the drive hardware select bits bits
        LD      A,(IX+HW_BYTE)         ;OR in the SD/DD bit, 8/5 bit, set to side A
        SET     SIDE_BIT,A             ;bit 2 = 1 side A
        OR      A,C                    ;Combine in the drive select bits
        LD      C,A                    ;store it here
        LD      (IY+DRIVE_PORT),A      ;Also store here this drives, density,size,sides and hardware select bits
                                       ;now select in hardware

        LD      A,(@PIO2B_BITS)               ;Get current state of DIRECTION_BIT, WD_INTRQ_BIT
        AND     A,11100000B
        SET     WP_BIT,A               ;Default no WP
        OR      A,C
        LD      B,A                    ;Store here
        IN      A,(WD2793_STATUS)      ;Check for disk write protect flag
        BIT     6,A
        LD      A,B                    ;get back info
        JR      Z,NOT_WP1
        RES     WP_BIT,A               ;WP ON, so lightup LED D17
NOT_WP1:OUT     (PIO2_DATA_B),A               ;Send to drive select port (Note this will also update TIL display)
        LD      (@PIO2B_BITS),A               ;Store hardware select bits (because this port cannot be read)

        XOR     A                      ;Invalidate WD2793 current Track/Sec Register
        OUT     (WD2793_SECTOR),A
        DEC     A
        OUT     (WD2793_TRACK),A
        RET


;---------  Set request for next Track ---------------------------------------------

SET_TRACK:                             ;CMD = 7
        CALL    GET_DATA               ;Get Track number
        CP      A,(IY+NTRKS)           ;Are we within range for this drive format
        JR      NC,BAD_TRACK

        LD      (IY+DRIVE_TRACK),A     ;Store Requested Track. (Do not update TIL's yet)

        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE       ;Send ACK back to S-100


;------ Special Set Track request (For cases where on a DS Disk:- Even tracks are on Side A, Odd on Side B.)
;       (Note: This format is NOT normally used in CPM or MS-DOS systems)

SET_TRACK_DS:                          ;CMD = 23H
        CALL    GET_DATA               ;Get Track number

        LD      C,A                    ;Save it
```

```
        LD      A,(IY+DRIVE_SS_DS_FLAG) ;Check if disk has a side B valid (04H)
        BIT     SIDE_BIT,A              ;Is it a double sided disk
        JR      Z,ONE_SIDE_ONLY                 ;No, then immediatly take care of it
        BIT     0,C                     ;Is it an ODD or EVEN track
        JR      Z,TRACK_EVEN

TRACK_ODD:
        RR      C                       ;Divide the TRACK number by 2 for hardware
        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        RES     SIDE_BIT,A              ;Set Bit 2 to 0 (Side B)
        OUT     (PIO2_DATA_B),A                 ;Display it -- AND -- Set in hardware
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)
        JR      ONE_SIDE_ONLY
TRACK_EVEN:
        RR      C                       ;Divide the TRACK number by 2 for hardware
        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        SET     SIDE_BIT,A              ;Set Bit 2 to 1 (Side A)
        OUT     (PIO2_DATA_B),A                 ;Display it -- AND -- Set in hardware
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)

ONE_SIDE_ONLY:
        LD      A,C
        CP      A,(IY+NTRKS)            ;Are we within range for this drive format
        JR      NC,BAD_TRACK

        LD      (IY+DRIVE_TRACK),A      ;Store Requested Track. (Do not update TIL's yet)

        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100

BAD_TRACK:
        LD      C,TRACK_RANGE_ERR
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100


;----------- Set request for side select for next sector to R/W.
;       0 = A, 1 = B For software flags
;       1 = A, 0 = B For hardware bit

SET_SIDE:                               ;CMD = 8
        CALL    GET_DATA
        OR      A,A                     ;0 = set to side A, 1 = set to side B
        JR      NZ,SideB

        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        SET     SIDE_BIT,A              ;Set Bit 2 to 1 for hardware Side A
        OUT     (PIO2_DATA_B),A                 ;Display it -- AND -- Set in hardware
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)
```

```
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100


SideB:  CP      A,1                     ;Must be 1 so set hardware to side B
        JR      NZ,BAD_SIDE1

        LD      A,(IY+DRIVE_SS_DS_FLAG) ;Check disk has a side B valid (04H)
        BIT     SIDE_BIT,A              ;Is it a double sided disk
        JR      Z,BAD_SIDE              ;No then flag an error

        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        RES     SIDE_BIT,A              ;Set Bit 2 to 0 for hardware Side B
        OUT     (PIO2_DATA_B),A                 ;Display it --- AND --- Set in hardware
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)

        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100

BAD_SIDE:
        LD      C,SIDE_ERR              ;No B side on this disk
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100
BAD_SIDE1:
        LD      C,SIDE_ERR1             ;Invalid side
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100




;------- Set request for set sector # for sector R/W  Note this is for CPM & CPM86 ONLY
;
;        Note for a Double sided Disk  Sector numbers > NSCTRS will be on Side B.)
;       This is the format commonly used by CPM BIOS's for 5" disks
;       CPM86 does not use a set side paramater for floppies (this is actully faster to implement)
;
;       Since MS-DOS uses the side information directly, it has its own routine (see below)

SET_SECTOR:                             ;CMD = 9
        XOR     A,A                     ;Default is 0 for Multi sector R/W
        LD      (@MULTI_SEC_COUNT),A
        LD      (@MULTI_FLAG),A                 ;NOT a multisector read by default

        CALL    GET_DATA
        OR      A,A                     ;No sector 0 allowed, 1,2,3....
        JR      Z,BAD_SECTOR

        BIT     SIDE_BIT,(IY+DRIVE_SS_DS_FLAG) ;Check if disk has a side B valid (04H)
        JR      NZ,DS_SECTORS           ;Yes, then take care of it
```

```
        CP      A,(IY+NSCTRS)           ;Are we within range for a SS disk (side A)
        JR      NC,BAD_SECTOR

        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        OUT     (WD2793_SECTOR),A       ;Update WD2793 chip

        LD      C,NO_ERRORS_FLAG        ;No hardware to check, so always OK
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100

DS_SECTORS:                             ;We have a double sided disk.
        CP      A,(IY+NSCTRS)           ;Are we within range of the first side
        JR      NC,SECOND_SIDE_SECTORS

        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        OUT     (WD2793_SECTOR),A       ;Update WD2793 chip

        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        SET     SIDE_BIT,A              ;Set Bit 2 to 1 (Side A)
        OUT     (PIO2_DATA_B),A                 ;Display it -- AND -- Set in hardware
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)

        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100

SECOND_SIDE_SECTORS:
        LD      C,(IY+NSCTRS)           ;Subtract one sides worth of sectors
        DEC     C                       ;Because NSCTRS +Sec/track+1
        SUB     A,C
        CP      A,(IY+NSCTRS)           ;Are we still within range
        JR      NC,BAD_SECTOR

        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        OUT     (WD2793_SECTOR),A       ;Update WD2793 chip

        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        RES     SIDE_BIT,A              ;Set Bit 2 to 0 (Side B)
        OUT     (PIO2_DATA_B),A                 ;Display it -- AND -- Set in hardware
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)

        LD      C,NO_ERRORS_FLAG        ;No hardware to check, so always OK
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100

BAD_SECTOR:
        LD      C,SECTOR_RANGE_ERR
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100


;------- Set request for set sector # for sector R/W  Note this is for MS-DOS ONLY
;
```

```
;       Since MS-DOS uses the side information directly, CPM & CPM86 it has its own routine (see above)

DOS_SET_SECTOR:                             ;CMD = 2EH
        XOR     A,A                     ;Default is 0 for Multi sector R/W
        LD      (@MULTI_SEC_COUNT),A
        LD      (@MULTI_FLAG),A             ;NOT a multisector read by default

        CALL    GET_DATA
        OR      A,A                     ;No sector 0 allowed, 1,2,3....
        JR      Z,BAD_SECTOR

        CP      A,(IY+NSCTRS)           ;Are we within range for disk
        JR      NC,BAD_SECTOR

        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        OUT     (WD2793_SECTOR),A       ;Update WD2793 chip

        LD      C,NO_ERRORS_FLAG        ;No hardware to check, so always OK
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100


;---------- COMMANDS Requiring the WD2793 Chip ----------------------------

; SEND A RESTORE COMMAND FOR THE CURRENT DISK.
;Note (IY+DRIVE_TRACK) is NOT updated to new track position

SET_HOME:                               ;CMD = 0AH
        CALL    CORE_RESTORE_ROUTINE    ;<<<<<< Core Head Restore routine >>>>
        JR      Z,SET_HOME_OK           ;0FFH returned it timeout, else error bits in [A] & [D]

        CP      A,0FFH                  ;Was there a timeout error
        LD      C,RESTORE_HUNG          ;"Timeout after RESTORE was sent"
        JP      Z,S100_ACK_ROUTINE      ;Z, if all OK
        LD      C,RESTORE_ERR           ;Must be a restore error
        JP      S100_ACK_ROUTINE

SET_HOME_OK:
        CALL    DisplayTrack            ;Display on TIL's 0,1
        CALL    DisplaySector           ;Display current Sector on TIL's 2,3
        XOR     A,A
        DEC     A
        LD      (IY+DRIVE_NEW),A        ;Set 0FFH if disk checked out.
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100 system all OK.



; General Seek to Track 0 (Restore) routine.
; Return 0FFH,NZ if Timeout,  else error bits (if any) in [A] & [D]
```

```
CORE_RESTORE_ROUTINE:                ;A callable "self contained" routine also use elsewhere
        LD      HL,RESTORE_DONE_CHK  ;Setup for Controller/Board for a hung condition
        CALL    ON_WD2793_INTRQ

        CALL    DELAY_30uS           ;We need to delay 30uS, previous command may have been an output to CMD port
WAIT_R: IN      A,(WD2793_STATUS)    ;Wait until chip is not busy
        BIT     7,A                  ;Check Drive Not Ready bit
        JR      NZ,WAIT_R            ;Z flag set if OK

        LD      A,RSCMD
        OUT     (WD2793_CMD),A

RESTORE_WAIT:
        JR      RESTORE_WAIT         ;Z80 will loop here until WD INTRQ triggers a NMI

RESTORE_DONE_CHK:
        CALL    OFF_WD2793_INTRQ
        LD      D,HOME_ERR_MASK
        CALL    WD2793_WAIT          ;Wait 30uS, then check status
        RET



; SEND A HEAD STEP-IN COMMAND TO THE CURRENT DRIVE
; Note (IY+DRIVE_TRACK) is NOT updated to new track position
; Also no retrys.

STEP_IN:                             ;CMD = 0BH
        LD      A,STEPIN
        OUT     (WD2793_CMD),A

        LD      D,SIN_ERR_MASK       ;Just look at DRIVE NOT READY bit
        CALL    WD2793_WAIT          ;Wait 30uS, then check status
        JR      Z,STEP_IN_OK

        CP      A,0FFH
        LD      C,STEP_IN_HUNG       ;Abort if Hung
        JP      Z,S100_ACK_ROUTINE
        LD      C,STEP_IN_ERR        ;Else must be a Drive not ready error
                                     ;because Verify bit is off
        JP      S100_ACK_ROUTINE     ;Send ACK back to S-100

STEP_IN_OK:
        CALL    DisplayTrack         ;Display on TIL's 0,1
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE     ;Send ACK back to S-100 system all OK.
```

```
; SEND A HEAD STEP-OUT COMMAND TO THE CURRENT DRIVE
; Note (IY+DRIVE_TRACK) is NOT updated to new track position
; Also no retrys.

STEP_OUT:                           ;CMD = 0CH
        LD      A,STEPOUT
        OUT     (WD2793_CMD),A

        LD      D,SOUT_ERR_MASK
        CALL    WD2793_WAIT         ;Wait 30uS, then check status
        JR      Z,STEP_OUT_OK

        CP      A,0FFH
        LD      C,STEP_OUT_HUNG
        JP      Z,S100_ACK_ROUTINE  ;Abort is Hung
        LD      C,STEP_OUT_ERR      ;Else must be a Drive not ready error
                                    ;because Verify bit is off
        JP      S100_ACK_ROUTINE    ;Send ACK back to S-100

STEP_OUT_OK:
        CALL    DisplayTrack        ;Display on TIL's 0,1
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE    ;Send ACK back to S-100 system all OK.



; Seek to track WITHOUT verify. Assumes VALID track in (IY+DRIVE_TRACK)
; Note, Re-seeks are not done here.

SEEK_NV:                            ;CMD = 0DH
        IN      A,(WD2793_TRACK)
        CP      A,(IY+DRIVE_TRACK)
        JR      Z,SEEK_NV_OK        ;If same, skip track seek

        LD      A,(IY+DRIVE_TRACK)
        OUT     (WD2793_DATA),A             ;Put required track# in data register
        LD      A,SKNCMD            ;Then send a Seek without Verify CMD
        OUT     (WD2793_CMD),A      ;Send seek cmd

        LD      D,SK_ERR_MASK
        CALL    WD2793_WAIT         ;Wait 30uS, then check status (0FFH if timeout)
        JR      Z,SEEK_NV_OK

        CP      A,0FFH
        LD      C,SEEK_NV_HUNG
```

```
        JP      Z,S100_ACK_ROUTINE     ;Abort is Hung
        BIT     7,A
        LD      C,SEEK_NV_ERR1         ;Seek NV, NOT READY flag
        JP      NZ,S100_ACK_ROUTINE    ;Send ACK back to S-100
        LD      C,SEEK_NV_ERR2         ;Must be SEEK ERROR flag
        JP      S100_ACK_ROUTINE       ;Send ACK back to S-100

SEEK_NV_OK:
        CALL    DisplayTrack           ;Display on TIL's 0,1
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE       ;Send ACK back to S-100 system all OK.



; General TRACK seek NON VERIFY routine.  Required TRACK in [A].
; Return 0FFH,NZ if Timeout,  else error bits (if any) in [A]

CORE_NV_SEEK_ROUTINE:                  ;A callable "self contained" routine also use elsewhere
        LD      HL,SEEK_NV_DONE_CHK    ;Setup for Controller/Board for a hung condition
        CALL    ON_WD2793_INTRQ               ;Turn on WD INTRQ for NMI jump to [HL]

        OUT     (WD2793_DATA),A               ;Put required track# in data register
        LD      A,SKNCMD               ;Then send a Seek with NO Verify CMD
        OUT     (WD2793_CMD),A         ;Send seek cmd to current drive

SEEK_NV_WAIT:
        JR      SEEK_NV_WAIT           ;Will wait here until a WD2793 INTRQ NMI forces Z80 to next line

SEEK_NV_DONE_CHK:
        CALL    OFF_WD2793_INTRQ
        LD      D,SK_ERR_MASK          ;Mask for Seek with verify bits
        CALL    WD2793_WAIT            ;Wait 30uS, then check status and return
        RET



; -------- This is the main seek to track command -----------
; Seek to track, WITH verify. Assumes VALID track in (IY+DRIVE_TRACK)
; Note, Re-seeks ARE done here.

SEEK_TRACK:                            ;CMD = 0EH
        LD      A,SEEK_RETRY_MAX       ;Put Seek retry count here
        LD      (@SEEK_RT),A

SEEK_TRACK1:
        IN      A,(WD2793_TRACK)
        CP      A,(IY+DRIVE_TRACK)     ;Get requested track
        JR      Z,SAME_TRACK           ;If same no need for track seek
```

```
        LD      A,(IY+DRIVE_TRACK)
        CALL    CORE_SEEK_ROUTINE       ;Seek to track in[A], Return NZ, if errors, Status in [D]
        JR      Z,SAME_TRACK

        CP      A,0FFH                  ;Was there a timeout error
        LD      C,SEEK_TRK_HUNG
        JP      Z,S100_ACK_ROUTINE      ;If so just abort
        JR      SEEK_TRK_RETRY          ;Try a restore and then try again

SAME_TRACK:
        CALL    DisplayTrack            ;Display on TIL's 0,1
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE        ;Send ACK back to S-100 system all OK.

SEEK_TRK_RETRY:
        LD      A,(@SEEK_RT)            ;Get Seek retry count
        DEC     A
        LD      (@SEEK_RT),A
        JP      Z,SEEK_TRK_ERROR        ;If Z, retrys did not work

        CALL    CORE_RESTORE_ROUTINE    ;Core Head Restore routine
        JR      Z,SEEK_TRACK1           ;Retry seek after this restore command

        CP      A,0FFH                  ;Was there a timeout error with restore
        LD      C,SEEK_REST_HUNG
        JP      Z,S100_ACK_ROUTINE
        LD      C,SEEK_REST_ERR                 ;Must be a WD2793 status RESTORE (DRIVE NOT READY) error
        JP      S100_ACK_ROUTINE

SEEK_TRK_ERROR:
        BIT     7,A
        LD      C,SEEK_TRK_ERR1                 ;Seek, DRIVE NOT READY flag
        JP      Z,S100_ACK_ROUTINE
        LD      C,SEEK_TRK_ERR2                 ;Must be SEEK ERROR flag
        JP      S100_ACK_ROUTINE



; General TRACK seek routine.  Required TRACK in [A].
; Return 0FFH,NZ if Timeout,  else error bits (if any) in [A]
; This is the main seek routine of the system

CORE_SEEK_ROUTINE:                      ;A callable "self contained" routine also use elsewhere
        LD      HL,SEEK_DONE_CHK        ;Setup for Controller/Board for a hung condition
        CALL    ON_WD2793_INTRQ

        OUT     (WD2793_DATA),A                 ;Put required track# in data register
        LD      A,SKCMD                 ;Then send a Seek with Verify CMD
```

```
        OUT     (WD2793_CMD),A          ;Send seek cmd to current drive

SEEK_WAIT:
        JR      SEEK_WAIT               ;Will wait here until a WD2793 INTRQ NMI forces Z80 to next line

SEEK_DONE_CHK:
        CALL    OFF_WD2793_INTRQ
        LD      D,SK_ERR_MASK           ;Mask for Seek with verify bits
        CALL    WD2793_WAIT             ;Wait 30uS, then check status and return
        RET



; Read an ID from the Track that the head of the current disk is on.
; Note any Sector ID from the track will be obtained depending where the head falls
; The currently selected side at (SIDE) is read. Six bytes of ID data are passed back to S-100 system

GET_TRACK_ID:                           ;CMD = 0FH
        LD      HL,ID_DONE_CHK          ;Setup for Controller/Board for a hung condition
        CALL    ON_WD2793_INTRQ

        LD      HL,SECTOR_ID_BUFFER     ;Will store the 6 bytes here
        LD      B,6

        LD      C,WD2793_DATA           ;6 bytes in B, Data port in [C] below
        LD      A,RDACMD                ;Send the Read Track ID CMD
        DI                              ;Just in case
        OUT     (WD2793_CMD),A

        CALL    DELAY_30uS

TID2:   IN      A,(WD2793_Status)       ;Get FDC status
        RRA                             ;C = Busy
        RRA                             ;C = DRQ
        JR      NC,TID2                 ;if no data
        INI                             ;INI is [C]->HL++ and B--
        JR      NZ,TID2                 ;get more data

ID3:    JR      ID3                     ;Wait until interrupted by WD2793 IOREQ pulse to NMI line
                                        ;The NMI routine will jump to ID_DONE_CHECK

ID_DONE_CHK:                            ;Will get to here when there is an INTRQ NMI
        CALL    OFF_WD2793_INTRQ

        EI
        LD      D,ID_ERR_MASK
        CALL    WD2793_WAIT             ;Wait 30uS, then check status and return
        JR      Z,GET_TRACK_ID_OK
```

```
        CP      A,0FFH                  ;Was there a timeout error
        LD      C,ID_ERR_HUNG
        JP      Z,S100_ACK_ROUTINE      ;Abort is Hung
        BIT     7,A
        LD      C,ID_ERR1               ;Get Track ID, NOT READY flag
        JP      NZ,S100_ACK_ROUTINE
        BIT     4,A
        LD      C,ID_ERR2               ;Get Track ID,  RNF ERROR flag
        JP      NZ,S100_ACK_ROUTINE
        BIT     2,A
        LD      C,ID_ERR3               ;Get Track ID,  LOST DATA ERROR flag
        JP      NZ,S100_ACK_ROUTINE
        LD      C,ID_ERR4               ;Must be CRC ERROR flag
        JP      S100_ACK_ROUTINE

GET_TRACK_ID_OK:
        LD      C,NO_ERRORS_FLAG
        CALL    SEND_DATA               ;Indicate we got data, now ready to send it to S-100 System

        LD      HL,SECTOR_ID_BUFFER     ;Get Location of ID Buffer
        LD      B,6                     ;Six bytes total
SEND_ID:LD      C,(HL)
        CALL    SEND_DATA               ;Send 6 bytes to S-100 System
        INC     HL
        DJNZ    SEND_ID

        CALL    DisplayTrack            ;Display on TIL's 0,1
        LD      A,(SECTOR_ID_BUFFER+2) ;Sector number will be here (can display raw data)
        OUT     (WD2793_SECTOR),A       ;Update sector #
        CALL    DisplaySector

        LD      C,NO_ERRORS_FLAG
        CALL    S100_ACK_ROUTINE        ;Send ACK back to S-100 system all OK.
        RET


; Get the current track the head is on. Not used currently!
; Note any Sector ID from the track will be obtained depending where the head falls
; The currently selected side at (SIDE) is read. Six bytes of ID data are passed back to S-100 system

CORE_GET_CURRENT_TRACK:                         ;A callable "self contained" routine also use elsewhere
        LD      HL,TRK_ID_DONE_CHK      ;Setup for Controller/Board for a hung condition
        CALL    ON_WD2793_INTRQ

        LD      HL,SECTOR_ID_BUFFER     ;Will store the 6 bytes here
        LD      B,6

        LD      C,WD2793_DATA           ;6 bytes in B, Data port in [C] below
        LD      A,RDACMD                ;Send the Read Track ID CMD
```

```
        DI                          ;Just in case
        OUT     (WD2793_CMD),A

        CALL    DELAY_30uS

XTID2:  IN      A,(WD2793_Status)   ;Get FDC status
        RRA                         ;C = Busy
        RRA                         ;C = DRQ
        JR      NC,XTID2            ;if no data
        INI                         ;INI is [C]->HL++ and B--
        JR      NZ,XTID2            ;get more data

XID3:   JR      XID3                ;Wait until interrupted by WD2793 IOREQ pulse to NMI line
                                    ;The NMI routine will jump to ID_DONE_CHECK

TRK_ID_DONE_CHK:                    ;Will get to here when there is an INTRQ NMI
        CALL    OFF_WD2793_INTRQ

        EI
        LD      D,ID_ERR_MASK
        CALL    WD2793_WAIT         ;Wait 30uS, then check status and return

        LD      A,0FFH              ;Was there an error
        RET     NZ                  ;If there was an error return an invalid track Number

        LD      HL,SECTOR_ID_BUFFER
        LD      A,(HL)
        RET                         ;Return with track


; --------------------------- <<< CORE MULTI-SECTOR READ ROUTINES >>>-----------------------------------------------------

; This routine first reads info from disk (via WD2793 chip) to RAM on the ZFDC board.
; Then a check is made to see if another sector remains to be read. If so, that sector is read.
; The current track & sector numbers are updated. This continues until all sectors are read.
; THEN, the complete set of data is sent to the S-100 system.
; The complication below is for double sided disks where once the last sector of one side
; is read we have to switch sides/tracks.
;
; NOTE: This routine is for CPM and CPM86 only.  For CPM side B = side A + IY+NSCTRS
;       For MS-DOS use DOS_READ_MULTI_SECTOR: (see below)

READ_MULTI_SECTOR:                  ;CMD = 29H
        CALL    GET_DATA            ;Get number of sectors to read
        LD      (@MULTI_SEC_COUNT),A   ;Store it
        LD      (@MULTI2_SEC_COUNT),A  ;Store it again
        LD      A,0FFH
        LD      (@MULTI_FLAG),A             ;Falg to indicate this IS a multisector read
```

```
        LD      HL,SECTOR_BUFFER        ;Need a special case where the SECTOR_BUFFER changes
        LD      (@DMA),HL

MORE_MULTI_RD:
        LD      A,SEEK_RETRY_MAX        ;Put Seek retry count here
        LD      (@SEEK_RT),A
        LD      A,SECTOR_RETRY_MAX      ;Put Sector read retry count here
        LD      (@SEC_RT),A

        LD      HL,(@DMA)               ;Update for multi-sector reads
        LD      A,TOP_OF_RAM            ;Free RAM up to (8000H + 7000H) = F800H, note allow for up to 1K sectors
        CP      A,H                     ;Check there is enough RAM to hold all the sector data in one request

        LD      C,BUFFER_OVERFLOW       ;Normally this will not be a problem for CPM since the only
        JP      C,S100_ACK_ROUTINE      ;use is for this routine is the Boot Loader (52X128 bytes)

        CALL    CORE_READ_SECTOR        ;We will come back here, ALWAYS
        RET     NZ                      ;Abort with an error if a problem

        LD      A,(@MULTI_SEC_COUNT)
        DEC     A
        LD      (@MULTI_SEC_COUNT),A    ;For next time
        JP      Z,SEND_MULTI_DATA       ;All done so send the complete package back to S-100 stystem

        BIT     SIDE_BIT,(IY+DRIVE_SS_DS_FLAG)        ;Check if disk has a side B valid (04H)
        JR      NZ,MULTI_DS_SECTORS     ;Yes, then take care of it

        LD      A,(IY+DRIVE_SECTOR)     ;Get the last requested Sector (Numbered 1,2,3....)
        INC     A                       ;Point to the next one
        CP      A,(IY+NSCTRS)           ;Are we within range for a SS disk (side A)
        JR      NC,NEW_TRACK_RD

        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        JP      MORE_MULTI_RD           ;Read the next sector

NEW_TRACK_RD:
        LD      A,(IY+DRIVE_TRACK)      ;Go to next track
        INC     A
        LD      (IY+DRIVE_TRACK),A      ;Note if we go past the final track we will get a seek error below.
        LD      A,1                     ;Back to sector 1
        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        JP      MORE_MULTI_RD           ;Read the next sector


MULTI_DS_SECTORS:                       ;We have a double sided disk. More complicated...
        LD      A,(IY+DRIVE_SECTOR)     ;Get the last requested Sector (Numbered 1,2,3....)
        INC     A                       ;Point to the next one

        CP      A,(IY+NSCTRS)           ;Are we within range of the first side
```

```
        JR      NC,MULTI_SECOND_SIDE_SECTORS

        LD      (IY+DRIVE_SECTOR),A    ;Store Requested Sector (Numbered 1,2,3....)
        LD      A,(@PIO2B_BITS)                ;Get stored hardware select bits (because this port cannot be read)
        SET     SIDE_BIT,A             ;Set Bit 2 to 1 (Side A)
        OUT     (PIO2_DATA_B),A                ;Display it -- AND -- Set in hardware
        LD      (@PIO2B_BITS),A                ;Store hardware select bits (because this port cannot be read)
        JP      MORE_MULTI_RD          ;Read the next sector

MULTI_SECOND_SIDE_SECTORS:
        LD      C,(IY+NSCTRS)          ;Subtract one sides worth of sectors
        DEC     C                      ;Because NSCTRS +Sec/track+1
        SUB     A,C
        CP      A,(IY+NSCTRS)          ;Are we still within range
        JR      NC,NEW_TRACK_RD2

        LD      (IY+DRIVE_SECTOR),A    ;Store Requested Sector (Numbered 1,2,3....)
        LD      A,(@PIO2B_BITS)                ;Get stored hardware select bits (because this port cannot be read)
        RES     SIDE_BIT,A             ;Set Bit 2 to 0 (Side B)
        OUT     (PIO2_DATA_B),A                ;Display it -- AND -- Set in hardware
        LD      (@PIO2B_BITS),A                ;Store hardware select bits (because this port cannot be read)
        JP      MORE_MULTI_RD          ;Read the next sector

NEW_TRACK_RD2:
        LD      A,(IY+DRIVE_TRACK)     ;Go to next track, but back on side A
        INC     A
        LD      (IY+DRIVE_TRACK),A
        LD      A,1
        LD      (IY+DRIVE_SECTOR),A    ;Store Requested Sector (Numbered 1,2,3....)
        LD      A,(@PIO2B_BITS)                ;Get stored hardware select bits (because this port cannot be read)
        SET     SIDE_BIT,A             ;Set Bit 2 to 1 (Side A)
        OUT     (PIO2_DATA_B),A                ;Display it -- AND -- Set in hardware
        LD      (@PIO2B_BITS),A                ;Store hardware select bits (because this port cannot be read)
        JP      MORE_MULTI_RD          ;Read the next sector


SEND_MULTI_DATA:                       ;Now send all the sectors worth of data back to the S-100 system in one block
        LD      C,NO_ERRORS_FLAG
        CALL    SEND_DATA              ;Indicate we got data, now ready to send it

        LD      HL,SECTOR_BUFFER       ;Get Location of first Sector Buffer

        CALL    DIRECTION_OUT          ;Set direction for output

SEND_MULTI_DATA1:
        LD      E,(IY+SEC_SIZE_BYTES)  ;128,256,512 or 1024 byte sector size
        LD      D,(IY+SEC_SIZE_BYTES+1)

MSEND_RD:
```

```
        LD      A,(HL)                  ;Next is actully a local copy of SEND_DATA
        DI                              ;Don't want to INT's while here since we are resetting a flag
        OUT     (PIO1_DATA_A),A                 ;First put it in output port
        XOR     A,A
        DEC     A
        LD      (@OUT_DATA_FLAG),A      ;Set flag to 0FFH. Will go to 0 when byte is recieved on the S100 side
        EI                              ;Allow INT's back on
MSEND_RD1:
        LD      A,(@OUT_DATA_FLAG)      ;Wait until S-100 signals it has recieved the data
        OR      A,A
        JR      NZ,MSEND_RD1            ;Previous byte not yet recieved by S100 system ---- so wait!
        INC     HL
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,MSEND_RD

        LD      A,(@MULTI2_SEC_COUNT)   ;More sectors to send?
        DEC     A
        LD      (@MULTI2_SEC_COUNT),A   ;Store for next loop
        JP      NZ,SEND_MULTI_DATA1

        CALL    DIRECTION_IN            ;Set back to default IN mode

        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE        ;Acknowledge we are done this wit RET to LOOP if 1 sector
                                        ;or back up to READ_MULTI_SECTOR if multiple sectors



; NOTE: This routine is for MS-DOS.  For MS-DOS side B = Head Select + IY+NSCTRS
;       For CPM & CPM86 use READ_MULTI_SECTOR: (see above)
;       [A],    0 = A, 1 = B For software flags
;       bit 2,  1 = A, 0 = B For hardware bit
;       DRIVE_SS_DS_FLAG = 40H if DS disk,  or 0H is SS disk

DOS_READ_MULTI_SECTOR:                  ;CMD = 2BH
        CALL    GET_DATA                ;Get number of sectors to read
        LD      (@MULTI_SEC_COUNT),A    ;Store it
        LD      (@MULTI2_SEC_COUNT),A   ;Store it again
        LD      A,0FFH
        LD      (@MULTI_FLAG),A                 ;Falg to indicate this IS a multisector read sector call

        LD      HL,SECTOR_BUFFER        ;Need a special case where the SECTOR_BUFFER changes
        LD      (@DMA),HL

DOS_MORE_MULTI_RD:
        LD      A,SEEK_RETRY_MAX        ;Put Seek retry count here
        LD      (@SEEK_RT),A
```

```
        LD      A,SECTOR_RETRY_MAX      ;Put Sector read retry count here
        LD      (@SEC_RT),A

        LD      HL,(@DMA)               ;Update for multi-sector reads
        LD      A,TOP_OF_RAM            ;Free RAM up to (8000H + 7000H) = F800H, note allow for up to 1K sectors
        CP      A,H                     ;Check there is enough RAM to hold all the sector data in one request

        LD      C,BUFFER_OVERFLOW       ;Normally this will not be a problem for MSDOS since MSDOS
        JP      C,S100_ACK_ROUTINE      ;says it will only use up IY+NSCTRS
                                        ;However later versions (MSDOS?) may need to send data back in big chunks.
                                        ;For now we will just Abort

        CALL    CORE_READ_SECTOR        ;We will come back here, --- ALWAYS ---
        RET     NZ                      ;Abort with an error if a problem

        LD      A,(@MULTI_SEC_COUNT)
        DEC     A
        LD      (@MULTI_SEC_COUNT),A    ;For next time
        JP      Z,SEND_MULTI_DATA       ;All done, so send the complete package back to S-100 stystem

        LD      A,(IY+DRIVE_SECTOR)     ;Get the last requested Sector (Numbered 1,2,3....)
        INC     A                       ;Point to the next one
        CP      A,(IY+NSCTRS)           ;Are we within range for sectors/track
        JR      NC,DOS_NEW_TRACK_RD     ;Next track required

        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        JP      DOS_MORE_MULTI_RD       ;Read the next sector

DOS_NEW_TRACK_RD:
        BIT     SIDE_BIT,(IY+DRIVE_SS_DS_FLAG)        ;Check if disk has a valid side B (04H)
        JR      NZ,DOS_MULTI_DS_SECTORS       ;Yes, then take care of it

        LD      A,(IY+DRIVE_TRACK)      ;SS, simple just go to next track
        INC     A
        LD      (IY+DRIVE_TRACK),A      ;Note if we go past the final track we will get a seek error below.
        LD      A,1                     ;Back to sector 1
        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        JP      DOS_MORE_MULTI_RD       ;Read the next sector


DOS_MULTI_DS_SECTORS:                   ;We have a double sided disk. More complicated...
        LD      A,(@PIO2B_BITS)               ;Get stored hardware select bits (because this port cannot be read)
        BIT     SIDE_BIT,A              ;Bit 2 = 0 for side B in hardware
        JR      Z,ALREADY_R_ON_B        ;Already on B

        RES     SIDE_BIT,A              ;Is on A, switch to side B
        OUT     (PIO2_DATA_B),A               ;Display it --- AND --- Set in hardware
        LD      (@PIO2B_BITS),A               ;Store hardware select bits (because this port cannot be read)
        LD      A,1                     ;Back to sector 1
```

```
        LD      (IY+DRIVE_SECTOR),A    ;Store Requested Sector (Numbered 1,2,3....)
        JP      DOS_MORE_MULTI_RD      ;Read the next sector

ALREADY_R_ON_B:
        SET     SIDE_BIT,A             ;Is on B, so switch to side A (AND increase track number)
        OUT     (PIO2_DATA_B),A               ;Display it --- AND --- Set in hardware
        LD      (@PIO2B_BITS),A               ;Store hardware select bits (because this port cannot be read)
        LD      A,(IY+DRIVE_TRACK)     ;go to next track
        INC     A
        LD      (IY+DRIVE_TRACK),A     ;Note if we go past the final track we will get a seek error below.
        LD      A,1                    ;Back to sector 1
        LD      (IY+DRIVE_SECTOR),A    ;Store Requested Sector (Numbered 1,2,3....)
        JP      DOS_MORE_MULTI_RD      ;Read the next sector



; --------------------------- <<< CORE SECTOR READ ROUTINE >>>-------------------------------------------------------

; This routine first reads info from disk (via WD2793 chip) to RAM on the ZFDC board.
; Once a complete sector is read, the sector data is the sent from that RAM to the S-100 system.
; Be sure [IY] points to the correct Drive Table.

READ_SECTOR:                           ;CMD = 10H
        LD      A,SEEK_RETRY_MAX       ;Put Seek retry count here
        LD      (@SEEK_RT),A
        LD      A,SECTOR_RETRY_MAX     ;Put Sector read retry count here
        LD      (@SEC_RT),A

        LD      HL,SECTOR_BUFFER       ;Except for Multi sector reads, this will always be SECTOR_BUFFER
        LD      (@DMA),HL

CORE_READ_SECTOR:
        IN      A,(WD2793_TRACK)
        CP      A,(IY+DRIVE_TRACK)     ;Get requested value
        JR      Z,SAME_RD_TRACK               ;If Same, no need to track seek

        LD      A,(IY+DRIVE_TRACK)
        CALL    CORE_SEEK_ROUTINE      ;<<<< Seek to track in[A], Return NZ, if errors [Status in [D]
        JR      Z,SAME_RD_TRACK

        CP      A,0FFH                 ;Was there a timeout error
        LD      C,RS_SEEK_TRK_HUNG
        JP      Z,S100_ACK_ROUTINE     ;Just Abort
        JP      READ_SECTOR_SEEK_RETRY ;Else try a restore and then try again

SAME_RD_TRACK:
        LD      E,(IY+SEC_SIZE_BYTES)  ;128,256,512 or 1024 byte sector size
        LD      D,(IY+SEC_SIZE_BYTES+1)
```

```
        LD      HL,RD_DONE_CHK          ;Setup for Controller/Board for a hung condition
        CALL    ON_WD2793_INTRQ

        DI                              ;Just in case

        LD      HL,(@DMA)               ;Update for multi-sector reads

        LD      A,(IY+DRIVE_SECTOR)     ;Get Requested Sector
        OUT     (WD2793_SECTOR),A       ;Send Sector request to WD2793 chip

        LD      C,WD2793_DATA           ;Data port in [C] below

        LD      A,RDCMD93               ;Send the Read Sector CMD
        OUT     (WD2793_CMD),A

        CALL    DELAY_30uS

RD2:    IN      A,(WD2793_Status)       ;Get FDC status
        RRA                             ;C = Busy
        RRA                             ;C = DRQ
        JR      NC,RD2                  ;if no data
        INI                             ;(C)->[HL++]
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,RD2                  ;At 6MHz, the Z80 seems to be fast enough here to keep up
        LD      (@DMA_NEW),HL           ;Cannot use new [HL]->@DMA yet because there may be an error and we
                                        ;will have to do this sector read to the origional @DMA address again

RD3:    JR      RD3                     ;Wait until interrupted by WD2793 IOREQ pulse to NMI line
                                        ;The NMI routine will jump to RD_DONE_CHECK
RD_DONE_CHK:                            ;Will get to here when there is an INTRQ NMI
        CALL    OFF_WD2793_INTRQ        ;we have to repeat the above read

        EI

        LD      D,RS_ERR_MASK
        CALL    WD2793_WAIT             ;Wait until chip is ready and check status, was there a problem?
        JR      Z,READ_SECTOR_OK

        CP      A,0FFH                  ;Was there a timeout error
        LD      C,SEC_READ_HUNG
        JP      Z,S100_ACK_ROUTINE      ;Abort is Hung
        JR      READ_SECTOR_RETRY       ;Else try reading a few more times

READ_SECTOR_OK:
        LD      A,(@MULTI_FLAG)                 ;Is this for a multi-sec read
        OR      A,A
        JR      Z,READ_SECTOR_OK1       ;Z if just a "normal" single sector read
```

```
        CALL    DisplayTrack            ;Display on TIL's 0,1
        CALL    DisplaySector           ;Display Sector on TIL's 2,3

        LD      HL,(@DMA_NEW)           ;For multi sector reads just update @DMA and read the next one
        LD      (@DMA),HL
        XOR     A,A
        RET

READ_SECTOR_OK1:
        LD      C,NO_ERRORS_FLAG
        CALL    SEND_DATA               ;Indicate we got data, now ready to send it

        LD      HL,SECTOR_BUFFER        ;Get Location of Sector Buffer (for single sector reads)
        LD      E,(IY+SEC_SIZE_BYTES)   ;128,256,512 or 1024 byte sector size
        LD      D,(IY+SEC_SIZE_BYTES+1)

                                        ;<< Now Send data to S-100 System
        CALL    DIRECTION_OUT           ;Set direction for output

SEND_RD:LD      A,(HL)                  ;Next is actully a local copy of SEND_DATA
        DI                              ;Don't want to INT's while here since we are resetting a flag
        OUT     (PIO1_DATA_A),A                 ;First put it in output port
        XOR     A,A
        DEC     A
        LD      (@OUT_DATA_FLAG),A      ;Set flag to 0FFH. Will go to 0 when byte is recieved on the S100 side
        EI                              ;Allow INT's back on
SEND_RD1:
        LD      A,(@OUT_DATA_FLAG)      ;Wait until S-100 signals it has recieved the data
        OR      A,A
        JR      NZ,SEND_RD1             ;Previous byte not yet recieved by S100 system ---- so wait!
        INC     HL
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,SEND_RD

        CALL    DIRECTION_IN            ;Set back to default IN mode

        CALL    DisplayTrack            ;Display on TIL's 0,1
        CALL    DisplaySector           ;Display Sector on TIL's 2,3

        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE        ;Acknowledge we are done this wit RET to LOOP if 1 sector
                                        ;or back up to READ_MULTI_SECTOR if multiple sectors

READ_SECTOR_SEEK_RETRY:
        LD      A,(@SEEK_RT)            ;Seek retry count here
        DEC     A
```

```
        LD      (@SEEK_RT),A            ;Retry again?

        JP      Z,RD_SEEK_RETRY_ERROR   ;If Z, retrys did not work

        CALL    CORE_RESTORE_ROUTINE    ;<<<<<< Core Head Restore routine
        JP      Z,CORE_READ_SECTOR      ;Retry seek after this restore command

        CP      A,0FFH                  ;Was there a timeout error within restore
        LD      C,RS_RESTORE_HUNG
        JP      Z,S100_ACK_ROUTINE
        LD      C,RS_RESTORE_ERR        ;Must be a "regular" WD2793 restore
        JP      S100_ACK_ROUTINE

RD_SEEK_RETRY_ERROR:
        BIT     7,A
        LD      C,RS_SEEK_TRK_ERR1      ;Seek retry, NOT READY flag
        JP      Z,S100_ACK_ROUTINE
        LD      C,RS_SEEK_TRK_ERR2      ;Must be SEEK ERROR flag
        JP      S100_ACK_ROUTINE

READ_SECTOR_RETRY:
        LD      A,(@SEC_RT)             ;Retry count here
        DEC     A
        LD      (@SEC_RT),A             ;Retry again?
        JP      NZ,SAME_RD_TRACK        ;Retry seek after this restore command

        BIT     7,A
        LD      C,SEC_READ_ERR1                 ;Read Sector, NOT READY flag
        JP      NZ,S100_ACK_ROUTINE
        BIT     4,A
        LD      C,SEC_READ_ERR2                 ;Read Sector,  RNF ERROR flag
        JP      NZ,S100_ACK_ROUTINE
        BIT     2,A
        LD      C,SEC_READ_ERR3                 ;Read Sector,  LOST DATA ERROR flag
        JP      NZ,S100_ACK_ROUTINE
        LD      C,SEC_READ_ERR4                 ;Must be CRC ERROR flag
        JP      S100_ACK_ROUTINE


; --------------------------- <<< CORE MULTI-SECTOR WRITE ROUTINE >>>-----------------------------------------------------
; This routine first reads multiple sectors worth of data from the S100 system to RAM on the ZFDC board.
; Once this is done multiple sectors are written to the floppy disk in one "gulp"
; The current track & sector numbers are updated. This continues until all sectors are written.
; The complication below is for double sided disks where once the last sector of one side
; is written we have to switch sides/tracks.

; NOTE: This routine is for CPM and CPM86.  For CPM side B = side A + IY+NSCTRS
;       For MS-DOS use DOS_WRITE_MULTI_SECTOR: (see below)
```

```
WRITE_MULTI_SECTOR:                 ;CMD = 2AH
        CALL    GET_DATA            ;Get number of sectors to write
        LD      (@MULTI_SEC_COUNT),A   ;Store it
        LD      (@MULTI2_SEC_COUNT),A  ;Store it again
        LD      A,0FFH
        LD      (@MULTI_FLAG),A             ;Falg to indicate this IS a multisector write

        LD      C,NO_ERRORS_FLAG
        CALL    SEND_DATA           ;Indicate we got WR CMD, we are now to get data

        LD      HL,SECTOR_BUFFER    ;Need a special case where the SECTOR_BUFFER changes
        LD      (@DMA),HL

        CALL    DIRECTION_IN        ;Set direction for input (probably is, but just in case)

MULTI_GET_WR1:
        LD      E,(IY+SEC_SIZE_BYTES)  ;128,256,512 or 1024 byte sector size
        LD      D,(IY+SEC_SIZE_BYTES+1)

MULTI_GET_WR:                       ;<< Get data from S-100 system
        LD      A,(@IN_DATA_FLAG)   ;This is actully a local copy of GET_DATA
        OR      A,A
        JR      Z,MULTI_GET_WR      ;Loop until we get something (>>> Add watchdog later <<<<)
        DI                          ;Don't want to INT's while here since we are resetting flag
        XOR     A,A
        LD      (@IN_DATA_FLAG),A   ;Set flag to 0. Will go to 0FFH with new data comming in from S100 side for next byte
        IN      A,(PIO1_DATA_B)             ;Get actual data
        EI                          ;Now allow another INT, if more data is to be gotten or sent to S100 side
        LD      (HL),A
        INC     HL
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,MULTI_GET_WR

        LD      A,(@MULTI2_SEC_COUNT)  ;More sectors to write?
        DEC     A
        LD      (@MULTI2_SEC_COUNT),A  ;Store for next loop
        JP      NZ,MULTI_GET_WR1

        CALL    DIRECTION_IN        ;Set direction for input (probably is, but just in case)

                                    ;NOW WRITE DATA TO DISK
        LD      HL,SECTOR_BUFFER    ;Note a special case where the SECTOR_BUFFER changes
        LD      (@DMA),HL

MORE_MULTI_WR:
        LD      A,SEEK_RETRY_MAX    ;Put Seek retry count here
        LD      (@SEEK_RT),A
```

```
        LD      A,SECTOR_RETRY_MAX      ;Put Sector read retry count here
        LD      (@SEC_RT),A

        LD      HL,(@DMA)               ;Update for multi-sector writes
        LD      A,TOP_OF_RAM            ;Free RAM up to (8000H + 7000H) = F800H, note allow for up to 1K sectors
        CP      A,H                     ;Check there is enough RAM to hold all the sector data in one request

        LD      C,BUFFER_OVERFLOW       ;Normally this will not be a problem for CPM since the only
        JP      C,S100_ACK_ROUTINE      ;use is for this routine is the Boot Loader (52X128 bytes)
                                        ;However later versions (MSDOS?) may need to send data back in big chunks.
                                        ;For now we will just Abort

        CALL    CORE_WRITE_SECTOR       ;<<<<<<< We will come back here, ALWAYS
        RET     NZ                      ;Abort with an error if a problem

        LD      A,(@MULTI_SEC_COUNT)
        DEC     A
        LD      (@MULTI_SEC_COUNT),A    ;For next time
        JP      Z,DONE_MULTI_WDATA      ;All done so send NO errors flag back to S-100 stystem

        BIT     SIDE_BIT,(IY+DRIVE_SS_DS_FLAG)        ;Check if disk has a side B valid (04H)
        JR      NZ,WMULTI_DS_SECTORS    ;Yes, then take care of it

        LD      A,(IY+DRIVE_SECTOR)     ;Get the last requested Sector (Numbered 1,2,3....)
        INC     A                       ;Point to the next one
        CP      A,(IY+NSCTRS)           ;Are we within range for a SS disk (side A)
        JR      NC,WNEW_TRACK_RD

        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        JP      MORE_MULTI_WR           ;Write the next sector

WNEW_TRACK_RD:
        LD      A,(IY+DRIVE_TRACK)      ;Go to next track
        INC     A
        LD      (IY+DRIVE_TRACK),A      ;Note if we go past the final track we will get a seek error below.
        LD      A,1                     ;Back to sector 1
        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        JP      MORE_MULTI_WR           ;Write the next sector


WMULTI_DS_SECTORS:                      ;We have a double sided disk. More complicated...
        LD      A,(IY+DRIVE_SECTOR)     ;Get the last requested Sector (Numbered 1,2,3....)
        INC     A                       ;Point to the next one

        CP      A,(IY+NSCTRS)           ;Are we within range of the first side
        JR      NC,WMULTI_SECOND_SIDE_SECTORS

        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
```

```
        SET     SIDE_BIT,A              ;Set Bit 2 to 1 (Side A)
        OUT     (PIO2_DATA_B),A                 ;Display it -- AND -- Set in hardware
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)
        JP      MORE_MULTI_WR           ;Read the next sector

WMULTI_SECOND_SIDE_SECTORS:
        LD      C,(IY+NSCTRS)           ;Subtract one sides worth of sectors
        DEC     C                       ;Because NSCTRS +Sec/track+1
        SUB     A,C
        CP      A,(IY+NSCTRS)           ;Are we still within range
        JR      NC,WNEW_TRACK_RD2

        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        RES     SIDE_BIT,A              ;Set Bit 2 to 0 (Side B)
        OUT     (PIO2_DATA_B),A                 ;Display it -- AND -- Set in hardware
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)
        JP      MORE_MULTI_WR           ;Read the next sector

WNEW_TRACK_RD2:
        LD      A,(IY+DRIVE_TRACK)      ;Go to next track, but back on side A
        INC     A
        LD      (IY+DRIVE_TRACK),A
        LD      A,1
        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        SET     SIDE_BIT,A              ;Set Bit 2 to 1 (Side A)
        OUT     (PIO2_DATA_B),A                 ;Display it -- AND -- Set in hardware
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)
        JP      MORE_MULTI_WR           ;Read the next sector

DONE_MULTI_WDATA:
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE        ;We are done here



; NOTE: This routine is for MS-DOS.  For MS-DOS side B = side A + select head
;       For CPM use WRITE_MULTI_SECTOR: (see above)

DOS_WRITE_MULTI_SECTOR:                         ;CMD = 2CH
        CALL    GET_DATA                ;Get number of sectors to write
        LD      (@MULTI_SEC_COUNT),A    ;Store it
        LD      (@MULTI2_SEC_COUNT),A   ;Store it again
        LD      A,0FFH
        LD      (@MULTI_FLAG),A                 ;Falg to indicate this IS a multisector write

        LD      C,NO_ERRORS_FLAG
        CALL    SEND_DATA               ;Indicate we got WR CMD, we are now to get data
```

```
        LD      HL,SECTOR_BUFFER        ;Need a special case where the SECTOR_BUFFER changes
        LD      (@DMA),HL

        CALL    DIRECTION_IN            ;Set direction for input (probably is, but just in case)

DOS_MULTI_1GET_WR:
        LD      E,(IY+SEC_SIZE_BYTES)   ;128,256,512 or 1024 byte sector size
        LD      D,(IY+SEC_SIZE_BYTES+1)

DOS_MULTI_GET_WR:                       ;<< Get data from S-100 system
        LD      A,(@IN_DATA_FLAG)       ;This is actully a local copy of GET_DATA
        OR      A,A
        JR      Z,DOS_MULTI_GET_WR      ;Loop until we get something (>>> Add watchdog later <<<<)
        DI                              ;Don't want to INT's while here since we are resetting flag
        XOR     A,A
        LD      (@IN_DATA_FLAG),A       ;Set flag to 0. Will go to 0FFH with new data comming in from S100 side for next byte
        IN      A,(PIO1_DATA_B)                 ;Get actual data
        EI                              ;Now allow another INT, if more data is to be gotten or sent to S100 side
        LD      (HL),A
        INC     HL
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,DOS_MULTI_GET_WR

        LD      A,(@MULTI2_SEC_COUNT)   ;More sectors to write?
        DEC     A
        LD      (@MULTI2_SEC_COUNT),A   ;Store for next loop
        JP      NZ,DOS_MULTI_1GET_WR

        CALL    DIRECTION_IN            ;Set direction for input (probably is, but just in case)

                                        ;NOW WRITE DATA TO DISK
        LD      HL,SECTOR_BUFFER        ;Note a special case where the SECTOR_BUFFER changes
        LD      (@DMA),HL

DOS_MORE_MULTI_WR:
        LD      A,SEEK_RETRY_MAX        ;Put Seek retry count here
        LD      (@SEEK_RT),A
        LD      A,SECTOR_RETRY_MAX      ;Put Sector read retry count here
        LD      (@SEC_RT),A

        LD      HL,(@DMA)               ;Update for multi-sector writes
        LD      A,TOP_OF_RAM            ;Free RAM up to (8000H + 7000H) = F800H, note allow for up to 1K sectors
        CP      A,H                     ;Check there is enough RAM to hold all the sector data in one request

        LD      C,BUFFER_OVERFLOW       ;Normally this will not be a problem for CPM since the only
        JP      C,S100_ACK_ROUTINE      ;use is for this routine is the Boot Loader (52X128 bytes)
```

```
                                ;However later versions (MSDOS?) may need to send data back in big chunks.
                                ;For now we will just Abort

        CALL    CORE_WRITE_SECTOR       ;We will come back here, --- ALWAYS -----
        RET     NZ                      ;Abort with an error if a problem

        LD      A,(@MULTI_SEC_COUNT)
        DEC     A
        LD      (@MULTI_SEC_COUNT),A    ;For next time
        JP      Z,DONE_MULTI_WDATA      ;All done so send NO errors flag back to S-100 stystem

        LD      A,(IY+DRIVE_SECTOR)     ;Get the last requested Sector (Numbered 1,2,3....)
        INC     A                       ;Point to the next one
        CP      A,(IY+NSCTRS)           ;Are we within range for a sectors/track
        JR      NC,DOS_WNEW_TRACK_WR

        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        JP      DOS_MORE_MULTI_WR       ;Write the next sector

DOS_WNEW_TRACK_WR:
        BIT     SIDE_BIT,(IY+DRIVE_SS_DS_FLAG)        ;Check if disk has a side B valid (04H)
        JR      NZ,DOS_WMULTI_DS_SECTORS      ;Yes, then take care of it

        LD      A,(IY+DRIVE_TRACK)      ;Go to next track
        INC     A
        LD      (IY+DRIVE_TRACK),A      ;Note if we go past the final track we will get a seek error below.
        LD      A,1                     ;Back to sector 1
        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        JP      DOS_MORE_MULTI_WR       ;Write the next sector


DOS_WMULTI_DS_SECTORS:                  ;We have a double sided disk. More complicated...
        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        BIT     SIDE_BIT,A              ;Hardware Bit 2 = 0 for side B
        JR      Z,ALREADY_W_ON_B        ;Already on B

        RES     SIDE_BIT,A              ;Is on A, switch to side B
        OUT     (PIO2_DATA_B),A                 ;Display it --- AND --- Set in hardware
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)
        LD      A,1                     ;Back to sector 1
        LD      (IY+DRIVE_SECTOR),A     ;Store Requested Sector (Numbered 1,2,3....)
        JP      DOS_MORE_MULTI_WR       ;Read the next sector

ALREADY_W_ON_B:
        SET     SIDE_BIT,A              ;Is on B, so switch to side A (AND increase track number)
        OUT     (PIO2_DATA_B),A                 ;Display it --- AND --- Set in hardware
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)
        LD      A,(IY+DRIVE_TRACK)      ;go to next track
        INC     A
```

```
        LD      (IY+DRIVE_TRACK),A     ;Note if we go past the final track we will get a seek error below.
        LD      A,1                    ;Back to sector 1
        LD      (IY+DRIVE_SECTOR),A    ;Store Requested Sector (Numbered 1,2,3....)
        JP      DOS_MORE_MULTI_WR      ;Read the next sector



; --------------------------- <<< CORE SECTOR WRITE ROUTINE >>>-----------------------------------------------------
; This routine first reads info from the S100 system to RAM on the ZFDC board.
; Once a complete sector is read from the S100 system it is written via the WD2793 to the disk.
; Be sure [IY] points to the correct Drive Table.

WRITE_SECTOR:                          ;CMD = 11H
        LD      C,NO_ERRORS_FLAG
        CALL    SEND_DATA              ;Indicate we got WR CMD, we are now to get data

        LD      HL,SECTOR_BUFFER       ;Get Location of Sector Buffer, never changes for single sector Writes
        LD      (@DMA),HL
        LD      E,(IY+SEC_SIZE_BYTES)  ;128,256,512 or 1024 byte sector size
        LD      D,(IY+SEC_SIZE_BYTES+1)

                                       ;<< First, get data from S-100 system
        CALL    DIRECTION_IN           ;Set direction for input

GET_WR: LD      A,(@IN_DATA_FLAG)      ;This is actully a local copy of GET_DATA
        OR      A,A
        JR      Z,GET_WR               ;Loop until we get something (>>> Add watchdog later <<<<)
        DI                             ;Don't want to INT's while here since we are resetting flag
        XOR     A,A
        LD      (@IN_DATA_FLAG),A      ;Set flag to 0. Will go to 0FFH with new data comming in from S100 side for next byte
        IN      A,(PIO1_DATA_B)                ;Get actual data
        EI                             ;Now allow another INT, if more data is to be gotten or sent to S100 side
        LD      (HL),A
        INC     HL
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,GET_WR

        CALL    DIRECTION_IN           ;Just to be safe

                                       ;OK Now write to disk from buffer
        LD      A,SEEK_RETRY_MAX       ;Put Seek retry count here
        LD      (@SEEK_RT),A
        LD      A,SECTOR_RETRY_MAX     ;Put Sector read retry count here
        LD      (@SEC_RT),A
```

```
CORE_WRITE_SECTOR:                  ;Will come directly here for Multi-sector writes
        IN      A,(WD2793_TRACK)
        CP      A,(IY+DRIVE_TRACK)  ;Get requested track
        JR      Z,SAME_WR_TRACK             ;If Same, no need to track seek

        LD      A,(IY+DRIVE_TRACK)  ;Get requested track
        CALL    CORE_SEEK_ROUTINE   ;<<<< Seek to track in[A], Return NZ, if errors [Status in [D]
        JR      Z,SAME_WR_TRACK

        CP      A,0FFH              ;Was there a timeout error
        LD      C,WS_SEEK_TRK_HUNG
        JP      Z,S100_ACK_ROUTINE  ;Just Abort
        JP      WRITE_SECTOR_SEEK_RETRY     ;Try a restore and then try again

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

        LD      HL,WR_DONE_CHK      ;Setup for Controller/Board for a hung condition
        CALL    ON_WD2793_INTRQ

        DI                          ;Just in case

        LD      HL,(@DMA)           ;Get Location of Sector Buffer

        LD      A,(IY+DRIVE_SECTOR) ;Get Requested Sector
        OUT     (WD2793_SECTOR),A   ;Send Sector request to WD2793 chip

        LD      C,WD2793_DATA       ;Data port in [C] below

        LD      A,WRCMD93           ;Send the Write Sector CMD
        OUT     (WD2793_CMD),A

        CALL    DELAY_30uS

WR2:    IN      A,(WD2793_Status)   ;Get FDC status
        RRA                         ;C = Busy
        RRA                         ;C = DRQ
        JR      NC,WR2              ;if no data
        OUTI
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,WR2              ;At 6MHz, the Z80 seems to be fast enough here to keep up
        LD      (@DMA_NEW),HL       ;Cannot use new [HL]->@DMA yet because there may be an error and we

WD3:    JR      WD3                 ;Wait until interrupted by WD2793 IOREQ pulse to NMI line
                                    ;The NMI routine will jump to WR_DONE_CHECK
```

```
WR_DONE_CHK:                        ;Will get to here when the is an INTRQ NMI
        CALL    OFF_WD2793_INTRQ

        EI

        LD      D,WS_ERR_MASK
        CALL    WD2793_WAIT         ;Wait until chip is ready and check status, was there a problem?

        JR      Z,WRITE_SECTOR_OK

        IN      A,(WD2793_STATUS)   ;Check is disk was write protected
        BIT     6,A
        LD      C,DISK_WP_ERR       ;Disk write protect error
        JP      NZ,S100_ACK_ROUTINE


        CP      A,0FFH              ;Was there a timeout error
        LD      C,SEC_WRITE_HUNG
        JP      Z,S100_ACK_ROUTINE  ;Abort is Hung

        JR      WRITE_SECTOR_RETRY  ;Try reading a few more times

WRITE_SECTOR_OK:
        CALL    DisplayTrack        ;Display on TIL's 0,1
        CALL    DisplaySector       ;Display Sector on TIL's 2,3

        LD      A,(@MULTI_FLAG)             ;Is this for a multi-sec read
        OR      A,A
        LD      C,NO_ERRORS_FLAG
        JP      Z,S100_ACK_ROUTINE  ;Acknowledge we are done for single sector write

        LD      HL,(@DMA_NEW)       ;For multi sector reads just update @DMA and read the next one
        LD      (@DMA),HL           ;Doew not matter if "normal single sec write, will not need it again
        XOR     A,A
        RET                         ;Return and ask for another if multi-sector reads

WRITE_SECTOR_SEEK_RETRY:
        LD      A,(@SEEK_RT)        ;Seek retry count here
        DEC     A
        LD      (@SEEK_RT),A        ;Retry again?

        JP      Z,WR_SEEK_RETRY_ERROR ;If Z, retrys did not work

        CALL    CORE_RESTORE_ROUTINE  ;<<<<<< Core Head Restore routine
        JP      Z,CORE_WRITE_SECTOR   ;Retry seek after this restore command

        CP      A,0FFH              ;Was there a timeout error within restore
        LD      C,WS_RESTORE_HUNG
        JP      Z,S100_ACK_ROUTINE
```

```
        LD      C,WS_RESTORE_ERR        ;Must be a "regular" WD2793 restore error
        JP      S100_ACK_ROUTINE

WR_SEEK_RETRY_ERROR:
        BIT     7,A
        LD      C,WS_SEEK_TRK_ERR1      ;Seek retry, NOT READY flag
        JP      Z,S100_ACK_ROUTINE
        LD      C,WS_SEEK_TRK_ERR2      ;Must be SEEK ERROR flag
        JP      S100_ACK_ROUTINE

WRITE_SECTOR_RETRY:
        LD      A,(@SEC_RT)             ;Retry count here
        DEC     A
        LD      (@SEC_RT),A             ;Retry again?

        JP      NZ,SAME_WR_TRACK        ;Retry seek after this restore command
        BIT     7,A
        LD      C,SEC_WRITE_ERR1        ;Read Sector, NOT READY flag
        JP      NZ,S100_ACK_ROUTINE
        BIT     4,A
        LD      C,SEC_WRITE_ERR2        ;Read Sector,  RNF ERROR flag
        JP      NZ,S100_ACK_ROUTINE
        BIT     2,A
        LD      C,SEC_WRITE_ERR3        ;Read Sector,  LOST DATA ERROR flag
        JP      NZ,S100_ACK_ROUTINE
        LD      C,SEC_WRITE_ERR4        ;Must be CRC ERROR flag
        JP      S100_ACK_ROUTINE



; READ TRACK ROUTINE. This routine first reads a complete track from a disk (via WD2793 chip) to
; RAM on the ZFDC board. Once a complete track is read the S100 system gets the data.
; Be sure [IY] points to the correct Drive Table.

READ_TRACK:                             ;CMD = 15H
        LD      HL,FBUFFER              ;Will will load the complete track image here (A000H)
        LD      BC,MAX_TRACK_SIZE       ;so for debugging it can be seen with the monitor command
        XOR     A,A

RDO_FILL:
        LD      (HL),A                  ;Fill the whole Buffer area with "blank" bytes
        INC     HL
        DEC     C
        JR      NZ,RDO_FILL
        DEC     B
        JR      NZ,RDO_FILL             ;Do 4K, (should be overkill)!

        LD      HL,RTR_DONE_CHK                 ;We need to setup the NMI return address for when a WD INTRQ arrives
        CALL    ON_WD2793_INTRQ
```

```
        DI                          ;Just in case

        LD      HL,FBUFFER          ;Will will build the complete track image here
        LD      C,WD2793_DATA       ;Data port for collection

        LD      A,RDTCMD            ;Send the Read Track CMD
        OUT     (WD2793_CMD),A

        CALL    DELAY_30uS

RT2:    IN      A,(WD2793_Status)   ;Get FDC status
        RRA                         ;C = Busy
        RRA                         ;C = DRQ
        JR      NC,RT2              ;if no data
        INI                         ;INI is [C]->HL++ and B--
        JR      RT2                 ;Get more data until INTRQ pulse NMI

RTR_DONE_CHK:
        CALL    OFF_WD2793_INTRQ

        EI

        LD      D,RT_ERR_MASK
        CALL    WD2793_WAIT         ;Wait 30uS, then check status and return
        JR      Z,READ_TRACK_OK

        CP      A,0FFH              ;Was there a timeout error
        LD      C,RT_ERR_HUNG
        JP      Z,S100_ACK_ROUTINE  ;Abort is Hung
        LD      C,RT_ERR
        JP      S100_ACK_ROUTINE    ;Abort is error

READ_TRACK_OK:
        LD      C,NO_ERRORS_FLAG    ;Send second NO_ERRORS_FLAG
        CALL    SEND_DATA           ;Indicate we got data, now ready to send it
                                    ;Will ALWAYS send MAX_TRACK_SIZE of data
        LD      HL,FBUFFER
        LD      DE,MAX_TRACK_SIZE   ;3K right now
SEND_TK:LD      C,(HL)
        CALL    SEND_DATA           ;Send MAX_TRACK_SIZE bytes to S-100 System
        INC     HL
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,SEND_TK

        CALL    DisplayTrack        ;Display on TIL's 0,1
        CALL    DisplaySector       ;Display Sector on TIL's 2,3
```

```
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE
        RET



;---------- FORMAT a disk track using the current disk paramater table ------------------
; This is by far the most complex module within the PROM software.
; Do not modify it unless you are sure what you are doing!

FORMAT_TRACK:                           ;CMD = 16H
        CALL    GET_DATA                ;Get the Track number
        LD      (F_TRK),A               ;Store Track number locally for speed

        CALL    GET_DATA                ;Did we get the required confirmation flag

        CP      A,CONFIRM_FORMAT
        LD      C,FORMAT1_ERR           ;Let S100 system know there was a problem and the format is aborted
        JP      NZ,S100_ACK_ROUTINE     ;Acknowledge we are not done

        IN      A,(WD2793_STATUS)       ;Check is disk write protect flag on
        BIT     6,A
        LD      C,DISK_WP_ERR           ;Disk write protect error
        JP      NZ,S100_ACK_ROUTINE

        LD      A,(IY+FORMAT_NUM)       ;Do not allow an unformatted disk
        OR      A,A
        LD      C,TABLE_ERR
        JP      Z,S100_ACK_ROUTINE
                                        ;<<<< FOR NEW TRACK FORMATS
                                        ;If you add a new track format, activate the lines below. It will display
                                        ;the total track byte count. See comments at the start of the tables to
                                        ;show how you calculate the total track size.
                                        ;Comment out when done as there is a 2 second delay here.

;       LD      E,(IY+TRACK_SIZE)       ;Length of a track in bytes
;       LD      D,(IY+TRACK_SIZE+1)
;       LD      A,D                     ;For debugging purposes show the total track size at the start
;       CALL    DisplayHEX_01
;       LD      A,E
;       CALL    DisplayHEX_23
;       CALL    DELAY_2S                ;Delay a little so user can read value

        XOR     A                       ;Zero sector display
        CALL    DisplayHEX_23

        LD      A,(F_TRK)               ;If track 0 or 2 then clear the track buffer area first
        OR      A,A
```

```
        CALL    Z,CLEAR_BUFFER
        CP      A,02H
        CALL    Z,CLEAR_BUFFER          ;Note this assumes we are doing a normal CPM type disk.

        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        SET     SIDE_BIT,A              ;Port hardware for side selection (1=Side A)
        OUT     PIO2_DATA_B,A           ;Data to port and into TIL 4. Start on side A
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)

        LD      A,(F_TRK)
        OUT     (WD2793_DATA),A                 ;Put required track# in data register
        CALL    DisplayHEX_01

        LD      A,SKNCMD                ;Then send a Seek with NO Verify CMD
        OUT     (WD2793_CMD),A          ;Send seek cmd to current drive

        LD      D,SK_ERR_MASK           ;Mask for Seek with no verify bits
        CALL    WD2793_WAIT             ;Wait 30uS, then check status and return
        JR      NZ,F_ERROR1             ;<<<<  Later seek to track!!!!!!!!!!!!

        XOR     A,A                     ;Setup for the build sector below (Side A)
        LD      (F_SIDE),A
        CALL    BUILD_TRACK             ;<<<<<<< Build a complete track image in RAM @ FBUFFER >>>>>>>>>>>>

;       LD      HL,WTR_DONE_CHK                 ;May need to setup the NMI return address for when a WD INTRQ arrives
;       CALL    ON_WD2793_INTRQ

        LD      HL,FBUFFER              ;To start of Buffer
        LD      E,(IY+TRACK_SIZE)
        LD      D,(IY+TRACK_SIZE+1)

        LD      A,(F_TRK)               ;Output the required track number
        OUT     (WD2793_TRACK),A

        LD      C,WD2793_DATA           ;Data port in [C] below

        DI                              ;Disable Interrupts
        LD      A,WRTCMD                ;Send the Write Track CMD
        OUT     (WD2793_CMD),A

        CALL    DELAY_30uS              ; <<<<<<<<<<<<<<<<<<<< WRITE_TRACK - SIDE A >>>>>>>>>>>>>>>>>>>>

WT2:    IN      A,(WD2793_Status)       ;Get FDC status
        RRA                             ;C = Busy
        RRA                             ;C = DRQ
        JR      NC,WT2                  ;if no data
        LD      A,(HL)
        OUT     (C),A
        INC     HL
```

```
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,WT2                  ;At 6MHz, the Z80 seems to be fast enough here to keep up
                                        ;When a track is done the WD2793 INTRQ will trigger the NMI routine
                                        ;which will jump to WTR_DONE_CHECK

                                        ;Note: I have disabled INT's for faster formatting. No need for them here
                                        ;normally
;WT3:   JR      WT3                     ;Z80 will stay here until WD2793 INTRQ is sent.

WTR_DONE_CHK:
;       CALL    OFF_WD2793_INTRQ
        EI

        LD      D,WT_ERR_MASK
        CALL    WD2793_WAIT             ;Wait until chip is ready
        JR      Z,SIDE_A_DONE
F_ERROR1:
        CALL    CORE_RESET_2793_ROUTINE         ;In case the chip got confused!
        LD      C,FORMAT2_ERR           ;Track Format error Side A
        JP      NZ,S100_ACK_ROUTINE     ;Abort because we cannnot Write track correctly!

SIDE_A_DONE:
;       CALL    DELAY_30uS              ; <<<<<<<<<<<<<<<<<<<< WRITE_TRACK - SIDE B >>>>>>>>>>>>>>>>>>>>

        BIT     SIDE_BIT,(IY+DRIVE_SS_DS_FLAG)          ;Is it a double sided disk (0=SS, 1=DS)
        JP      Z,FORMAT_TRACK_DONE

FORMAT_SIDE_B:                          ;----------- If a DS disk then do same for side B ------
        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        RES     SIDE_BIT,A              ;Port hardware for side selection (0=Side B)
        OUT     PIO2_DATA_B,A           ;Data to port and into TIL 4
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)

        LD      HL,FBUFFER              ;Back to start of Buffer

        LD      A,02H                   ;Always Side 2
        LD      (F_SIDE),A

        CALL    BUILD_TRACK             ;<<<<<<< Build a complete track image in RAM >>>>>>>>>>>>

;       LD      HL,WTR_DONE_CHK_B       ;We need to setup the NMI return address for when a WD INTRQ arrives
;       CALL    ON_WD2793_INTRQ

        LD      HL,FBUFFER
        LD      E,(IY+TRACK_SIZE)
        LD      D,(IY+TRACK_SIZE+1)
```

```
        LD      A,(F_TRK)               ;output the required track number
        OUT     (WD2793_TRACK),A

        LD      C,WD2793_DATA           ;Data port in [C] below

        DI                              ;Disable Interrupts
        LD      A,WRTCMD                ;Send the Write Track CMD
        OUT     (WD2793_CMD),A

        CALL    DELAY_30uS              ; <<<<<<<<<<<<<<<<<<<< WRITE_TRACK - SIDE B >>>>>>>>>>>>>>>>>>>>

WT2B:   IN      A,(WD2793_Status)       ;Get FDC status
        RRA                             ;C = Busy
        RRA                             ;C = DRQ
        JR      NC,WT2B                 ;if no data
        LD      A,(HL)
        OUT     (C),A
        INC     HL
        DEC     DE
        LD      A,E
        OR      A,D
        JR      NZ,WT2B                 ;At 6MHz, the Z80 seems to be fast enough here to keep up
                                        ;When a track is done the WD2793 INTRQ will trigger the NMI routine
                                        ;which will jump to WTR_DONE_CHECK

;WT3B:  JR      WT3B                    ;Z80 will stay here until Wd2793 INTRQ is sent.

;WTR_DONE_CHK_B:
;       CALL    OFF_WD2793_INTRQ
        EI

        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        SET     SIDE_BIT,A              ;Port hardware for side selection (1=Side A)
        OUT     PIO2_DATA_B,A           ;Data to port and into TIL 4
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)

        LD      D,WT_ERR_MASK
        CALL    WD2793_WAIT             ;Wait until chip is ready
        JR      Z,FORMAT_TRACK_DONE

        CALL    CORE_RESET_2793_ROUTINE         ;In case the chip got confused!
        LD      C,FORMAT3_ERR           ;Track Format error Side B
        JP      NZ,S100_ACK_ROUTINE     ;Abort because we cannnot Write track correctly!

FORMAT_TRACK_DONE:
        LD      C,NO_ERRORS_FLAG
        CALL    SEND_DATA
        RET
;
```

```
;
;
CLEAR_BUFFER:                           ;Fill Buffer araa with 00's or 4E's
        LD      HL,FBUFFER              ;Will will build the complete sector image here
        LD      BC,MAX_TRACK_SIZE       ;Do 3K, (should be overkill)!
        BIT     DENSITY_BIT,(IY+DRIVE_PORT)   ;is is a SD or DD disk format
        JR      Z,SD_FILL
        LD      A,4EH                   ;For double density flush to end of track with 4E's (IBM 34 Format)
        JR      DO_FILL
SD_FILL:XOR     A,A                     ;For SD flush with 0's (IBM 3740 Format)
DO_FILL:LD      (HL),A                  ;Fill the whole Buffer area with "blank" bytes
        INC     HL
        DEC     C
        JR      NZ,DO_FILL
        DJNZ    DO_FILL                 ;Do 4K, (should be overkill)!
        RET

;
;---------------- BUILD TRACK IMAGE IN RAM ------------------------
;       Note for debugging you can examine the track image in the RAM of the
;       ZFDC board using the Monitor command. It starts at RAM_START+2000H.
;
;       Build in memory at FBUFFER a complete SD/DD Track, repeat for whole disk.
;
BUILD_TRACK:
        EXX                             ;the sector numbers on a track (usually 1,2,3,4...
        LD      A,(IY+SEC_SKEW_TABLE)   ;Set [HL'] to point to the table containing the order of
        LD      l,A
        LD      A,(IY+SEC_SKEW_TABLE+1)
        LD      h,A
        EXX                             ;Sector skew table in [HL']

        LD      HL,FBUFFER              ;This will be at FBUFFER where track is built in RAM
        BIT     DENSITY_BIT,(IY+DRIVE_PORT)   ;Will build a different track image for SD or DD disks

        JP      Z,DD_TRACKS             ;Seperate sector image for DD tracks

SD_TRACKS:
        CALL    SD_TRK_HEADER           ;Drop in header and Index mark BEFORE first sector
        EXX
        LD      A,(hl)                  ;sector# in D, usually 1 at the start
        EXX
        LD      D,A
        LD      E,(IY+NSCTRS)           ;Total Sectors/side
        DEC     E                       ;Because sec/track +1 in table
NEXT_SEC:
        CALL    SD_BUILD_SEC            ;<<< Build a sector >>>>>
        EXX
        INC     hl                      ;point to next sector number in sec skew table
```

```
        LD      A,(hl)
        EXX
        LD      D,A                     ;store sector # in D
        DEC     E
        JR      NZ,NEXT_SEC             ;All sectors are in RAM

        CALL    SD_TRK_END              ;Now need to flush out track to end
        RET
;
;       Build in memory at FBUFFER a complete Single Density Track.
;
SD_TRK_HEADER:                          ;Lay down the track header before the 1st sector
        LD      A,(IY+GAP_FILL_CHAR)    ;Do not alter [D]= Sec# or [E]= Sec/side
        LD      B,(IY+HEADR)            ;Header has 40 (FF's)
        CALL    DROP                    ;drop it at the end of the growing image (Count in B)
        XOR     A,A                     ;Now 6 0's
        LD      B,6
        CALL    DROP
        LD      A,0FCH                  ;Index ID mark
        LD      (HL),A                  ;drop into image
        INC     HL
        LD      A,(IY+GAP_FILL_CHAR)    ;Now 26 (FF's)
        LD      B,26                    ;Header has the count of fill characters required
        CALL    DROP                    ;drop it at the end of the growing image (Count in B)
        RET                             ;return with [HL] pointing to first sector byte
;
SD_BUILD_SEC:                           ;Lay down a sector at current [HL]. Do not alter [D] or [E]
        XOR     A,A
        LD      B,(IY+GAP1)             ;<---- (eg. 6,0's for IMB 3740, 8")
        CALL    DROP
        LD      A,0FEH                  ;Sector ID Address mark
        LD      (HL),A                  ;drop it in the growing image
        INC     HL
        LD      A,(F_TRK)               ;Drop in the track #
        LD      (HL),A
        INC     HL
        LD      A,(F_SIDE)              ;Side#, 0 for A side, 02H for B side
        OR      A,A
        LD      A,00
        JR      Z,BLD_ASIDE
        LD      A,01
BLD_ASIDE:
        LD      (HL),A                  ;0 here for A side, 1 for B side
        INC     HL
        LD      (HL),D                  ;Drop in sector #
        INC     HL
        LD      A,(IY+SEC_SIZE_FLAG)    ;128=0,256=1,512=2, 1024=3
        LD      (HL),A                  ;drop in sector length byte
        INC     HL
```

```
        LD      A,0F7H                  ;Dropping in a 0F7 will cause the 179x
        LD      (HL),A                  ;to write in the 2 CRC bytes
        INC     HL
        LD      A,(IY+GAP_FILL_CHAR)
        LD      B,(IY+GAP2)             ;<---- (eg. 11,FF's for IMB 3740, 8")
        CALL    DROP
        XOR     A,A
        LD      B,(IY+GAP1)             ;<---- (eg. 6,0's for IMB 3740, 8")
        CALL    DROP
        LD      A,0FBH                  ;Data address mark for 1791/5
        LD      (HL),A                  ;to write in the 2 CRC bytes
        INC     HL
        LD      A,D
        CP      A,1                     ;If first sector then store data marker
        JR      NZ,SDATA_FIELD
        LD      (S_DATA_MARK),HL        ;Pointer to start of sector data area
SDATA_FIELD:                            ;Now write in the sector data field itself
        LD      A,(IY+SEC_SIZE_FLAG)    ;128,256,512 or 1024 byte sector size
        LD      B,128
        LD      C,1                     ;1 loop of 128 bytes in WR_DATA_FIELD below
        OR      A,A
        JR      Z,SD_DATA_FIELD                 ;Do 128 byte write (B=128)
        LD      B,0                     ;Need 256 bytes for the rest of possible sectors
        LD      C,1                     ;One loop
        CP      A,1
        JR      Z,SD_DATA_FIELD                 ;Do 256 byte write
        LD      C,2                     ;2 loops of 256 bytes in WRITE_DATA_FIELD
        CP      A,2
        JR      Z,SD_DATA_FIELD                 ;Do 512 byte read
        LD      C,4                     ;(must be 3) so 1024 byte sector
SD_DATA_FIELD:
        LD      A,(IY+DATA_FILL_CHAR)   ;get the sector fill character (usually E5)
SDF0:   CALL    DROP
        DEC     C
        JR      NZ,SDF0                 ;Decrese [C] to 0
        LD      A,D
        CP      A,1                     ;If first sector then store image mark
        JR      NZ,NOT_FIRST
        LD      (E_DATA_MARK),HL        ;For first sector will diaplay data late
NOT_FIRST:
        LD      A,0F7H                  ;Dropping in a 0F7 will cause the 1791/5
        LD      (HL),A                  ;to write in the 2 CRC bytes
        INC     HL
        LD      A,(IY+GAP_FILL_CHAR)
        LD      B,(IY+GAP3)             ;<---- (eg. 27,FF's for IMB 3740, 8")
        CALL    DROP
        LD      A,D
        CP      A,1                     ;If first sector then store image mark
        RET     NZ
```

```
        LD      (E_SEC_MARK),HL              ;For first sector will diaplay data later
        RET
;
SD_TRK_END
        LD      (S_GAP4_MARK),HL      ;Mark beginning of end of track field (GAP4)
        LD      A,(IY+GAP_FILL_CHAR)
        LD      B,(IY+GAP4)           ;<---- (eg 247,FF's  for IMB 3740, 8")
        LD      C,(IY+GAP4R)          ;Times to repeat DROP
SD_TRK1:CALL    DROP
        DEC     C
        JR      NZ,SD_TRK1
        LD      (E_GAP4_MARK),HL      ;Mark end of Track
        RET
;
DROP:   LD      (HL),A                ;DATA block loader
        INC     HL                    ;B= byte count,HL pointer
        DJNZ    DROP                  ;A = value to drop into image. Count in B
        RET
;
;
;
;       Build in memory at FBUFFER a complete Double Density Track.
;
DD_TRACKS:                            ;Same thing for DD Trdcks. Extra stuff req.
        CALL    DD_TRK_HEADER         ;Drop in header and Index mark BEFORE first sector

        EXX
        LD      A,(hl)
        EXX
        LD      D,A                   ;sector# in D, usually 1 at the start
        LD      E,(IY+NSCTRS)         ;Total Sectors/side
        DEC     E                     ;Because sec/track +1 in table
DD_NEXT_SEC:
        CALL    DD_BUILD_SEC          ;<<< Build a DD sector >>>>>
        EXX
        INC     hl                    ;point to next sector number in sec skew table
        LD      A,(hl)
        EXX
        LD      D,A                   ;store sector # in D
        DEC     E
        JR      NZ,DD_NEXT_SEC        ;All sectors are in RAM

        CALL    DD_TRK_END            ;Now need to flush out track to end
        RET
;
;
DD_TRK_HEADER:                        ;Lay down the track header before the 1st sector
        LD      A,(IY+GAP_FILL_CHAR)  ;Do not alter [D]= Sec# or [E]= Sec/side
        LD      B,(IY+HEADR)          ;Header has 80 (4E's)
```

```
        CALL    DROP                    ;drop it at the end of the growing image (Count in B)
        XOR     A,A                     ;Now 12 0's
        LD      B,12
        CALL    DROP
        LD      A,0F6H                  ;3 of F6's
        LD      B,3
        CALL    DROP
        LD      A,0FCH                  ;Index ID mark (FC)
        LD      (HL),A                  ;drop into image
        INC     HL
        LD      A,(IY+GAP_FILL_CHAR)    ;Now 50 (4Es)
        LD      B,50                    ;Header has the count of fill characters required
        CALL    DROP                    ;drop it at the end of the growing image (Count in B)
        RET                             ;return with [HL] pointing to first sector byte
;
DD_BUILD_SEC:                           ;Lay down a sector at current [HL]. Do not alter [D] or [E]
        XOR     A,A
        LD      B,(IY+GAP1)             ;<---- (eg. 12,0's for IBM System 34 Format)
        CALL    DROP
        LD      A,0F5H                  ;Special DD bytes
        LD      B,3
        CALL    DROP
        LD      A,0FEH                  ;Sector ID Address mark
        LD      (HL),A                  ;drop it in the growing image
        INC     HL
        LD      A,(F_TRK)               ;Drop in the track #
        LD      (HL),A
        INC     HL
        LD      A,(F_SIDE)              ;Side#, 0 for A side, 02H for B side
        OR      A,A
        LD      A,0
        JR      Z,DBLD_ASIDE
        LD      A,1                     ;1 for side B
DBLD_ASIDE:
        LD      (HL),A                  ;0 here for A side, 1 for B side
        INC     HL
        LD      (HL),D                  ;Drop in sector #
        INC     HL
        LD      A,(IY+SEC_SIZE_FLAG)            ;128=0,256=1,512=2, 1024=3
        LD      (HL),A                  ;drop in sector length byte
        INC     HL
        LD      A,0F7H                  ;Dropping in a 0F7 will cause the 179x
        LD      (HL),A                  ;to write in the 2 CRC bytes
        INC     HL
        LD      A,(IY+GAP_FILL_CHAR)
        LD      B,(IY+GAP2)             ;<----(eg 22,4E's for IBM System 34 Format)
        CALL    DROP
        XOR     A,A
        LD      B,(IY+GAP1)             ;<---- (eg 8,0's for IBM System 34 Format)
```

```
        CALL    DROP
        LD      A,0F5H                  ;Special DD bytes
        LD      B,3
        CALL    DROP
        LD      A,0FBH                  ;Data address mark for 1791/5
        LD      (HL),A                  ;to write in the 2 CRC bytes
        INC     HL
        LD      A,D                     ;Get sec #
        CP      A,1                     ;If first sector then store data marker
        JR      NZ,DDATA_FIELD
        LD      (S_DATA_MARK),HL        ;Pointer to start of sector data area
DDATA_FIELD:                            ;Now write in the sector data field itself
        LD      A,(IY+SEC_SIZE_FLAG)    ;128,256,512 or 1024 byte sector size
        LD      B,128
        LD      C,1                     ;1 loop of 128 bytes in WR_DATA_FIELD below
        OR      A,A
        JR      Z,DD_DATA_FIELD                 ;Do 128 byte write (B=128)
        LD      B,0                     ;Need 256 bytes for the rest of possible sectors
        LD      C,1                     ;One loop
        CP      A,1
        JR      Z,DD_DATA_FIELD                 ;Do 256 byte write
        LD      C,2                     ;2 loops of 256 bytes in WRITE_DATA_FIELD
        CP      A,2
        JR      Z,DD_DATA_FIELD                 ;Do 512 byte read
        LD      C,4                     ;(must be 3) so 1024 byte sector
DD_DATA_FIELD:
        LD      A,(IY+DATA_FILL_CHAR)   ;get the sector fill character (usually E5)
DDF0:   CALL    DROP
        DEC     C
        JR      NZ,DDF0                 ;Decrease [C] to 0
        LD      A,D                     ;Get back sec #
        CP      A,1                     ;If first sector then store image mark
        JR      NZ,DNOT_FIRST
        LD      (E_DATA_MARK),HL        ;For first sector will diaplay data late
DNOT_FIRST:
        LD      A,0F7H                  ;Dropping in a 0F7 will cause the 1791/5
        LD      (HL),A                  ;to write in the 2 CRC bytes
        INC     HL
        LD      A,(IY+GAP_FILL_CHAR)
        LD      B,(IY+GAP3)             ;<---- (54,4E's for IBM System 34 Format)
        CALL    DROP
        LD      A,D
        CP      A,1                     ;If first sector then store image mark
        RET     NZ
        LD      (E_SEC_MARK),HL                 ;For first sector will diaplay data later
        RET
;
DD_TRK_END
        LD      (S_GAP4_MARK),HL        ;Mark beginning of end of track field (GAP4)
```

```
        LD      A,(IY+GAP_FILL_CHAR)
        LD      B,(IY+GAP4)             ;<---- (eg 598 4E's for IBM System 34 Format)
        LD      C,(IY+GAP4R)            ;Times to repeat DROP
DD_TRK1:CALL    DROP
        DEC     C
        JR      NZ,DD_TRK1
        LD      (E_GAP4_MARK),HL        ;Mark end of Double density Track
        RET

;-- END OF TRACK BUILD / FORMAT ------------



;----------------- Get the WD2793 Track Register value.
GET_WD_TRACK:                           ;CMD = 12H
        IN      A,(WD2793_TRACK)
        LD      C,A
        CALL    DisplayTrack
        CALL    SEND_DATA               ;Send current TRACK number in [C]
        LD      C,NO_ERRORS_FLAG
        JP      Z,S100_ACK_ROUTINE


;----------------- Get the WD2793 Sector Register value.
GET_WD_SECTOR:                          ;CMD = 13H
        IN      A,(WD2793_SECTOR)
        LD      C,A
        CALL    DisplaySector
        CALL    SEND_DATA               ;Send current SECTOR number in [C]
        LD      C,NO_ERRORS_FLAG
        JP      Z,S100_ACK_ROUTINE


;--------------- Get the WD2793 Status Register value.
GET_WD_STATUS:                          ;CMD = 14H
        IN      A,(WD2793_STATUS)
        LD      C,A
        CALL    SEND_DATA               ;Send WD2793 Status Bits in [C]
        LD      C,NO_ERRORS_FLAG
        JP      Z,S100_ACK_ROUTINE


;---------------- Turn on the Debug mode.
SET_DEBUG_ON:                           ;CMD = 17H
        LD      A,0FFH
        LD      (@DEBUG_FLAG),A
        LD      C,NO_ERRORS_FLAG
        JP      Z,S100_ACK_ROUTINE
```

```
;----------------- Turn on the Debug mode.
SET_DEBUG_OFF:                          ;CMD = 18H
        XOR     A,A
        LD      (@DEBUG_FLAG),A
        LD      C,NO_ERRORS_FLAG
        JP      Z,S100_ACK_ROUTINE



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

RAM_DUMP:                               ;CMD = 19H
        LD      C,NO_ERRORS_FLAG
        CALL    S100_ACK_ROUTINE

        PUSH    IY                      ;Save [IY],

        LD      IY,@DRIVE_A_DRIVE_TABLE       ;Start with Drive A:
        CALL    SEND_DRIVE_DUMP

        CALL    GET_DATA                ;check if CMD_ABORT was sent
        CP      A,CMD_ABORT
        JP      Z,RAM_DUMP_DONE

        LD      IY,@DRIVE_B_DRIVE_TABLE       ;then Drive B:
        CALL    SEND_DRIVE_DUMP

        CALL    GET_DATA                ;check if CMD_ABORT was sent
        CP      A,CMD_ABORT
        JP      Z,RAM_DUMP_DONE

        LD      IY,@DRIVE_C_DRIVE_TABLE       ;then Drive C:
        CALL    SEND_DRIVE_DUMP

        CALL    GET_DATA                ;check if CMD_ABORT was sent
        CP      A,CMD_ABORT
        JP      Z,RAM_DUMP_DONE

        LD      IY,@DRIVE_D_DRIVE_TABLE       ;lastly Drive D:
        CALL    SEND_DRIVE_DUMP

        CALL    GET_DATA                ;check if CMD_ABORT was sent
        CP      A,CMD_ABORT
        JP      Z,RAM_DUMP_DONE

        PUSH    HL                      ;Next send current [SP] value
        PUSH    BC
```

```
        LD      HL,@TEMP_STACK
        LD      (@TEMP_STACK),SP
        INC     HL
        LD      C,(HL)                  ;Switch order, High byte first
        CALL    SEND_DATA               ;Send data in [C]
        DEC     HL
        LD      C,(HL)
        CALL    SEND_DATA               ;Send data in [C]
        POP     BC
        POP     HL

        LD      A,(@CURRENT_DRIVE_SELECT) ;Current selected drive. 1,2,4,8
        LD      C,A
        CALL    SEND_DATA               ;Send data in [C]

RAM_DUMP_DONE:
        POP     IY                      ;Get back [IY]
        JP      S100_ACK_ROUTINE        ;Finish up and return to loop


SEND_DRIVE_DUMP:                        ;Send Drive variables addressed in [IY]
        LD      C,(IY+DRIVE_PORT)       ;First the hardware select port bit values
        CALL    SEND_DATA

        LD      C,(IY+NSCTRS)           ;Sectors/Track for disk
        CALL    SEND_DATA

        LD      C,(IY+NTRKS)            ;Tracks/Side
        CALL    SEND_DATA

        LD      C,(IY+HEADR)            ;For Formatting
        CALL    SEND_DATA

        LD      C,(IY+GAP1)             ;       "
        CALL    SEND_DATA

        LD      C,(IY+GAP2)             ;       "
        CALL    SEND_DATA

        LD      C,(IY+GAP3)             ;       "
        CALL    SEND_DATA

        LD      C,(IY+GAP4)             ;       "
        CALL    SEND_DATA

        LD      C,(IY+GAP4R)            ;       "
        CALL    SEND_DATA

        LD      C,(IY+SEC_SIZE_FLAG)    ;0=128 Byte sectors, 1 = 256, 2 = 512, 4=1024 Byte sectors
```

```
        CALL    SEND_DATA

        LD      C,(IY+GAP_FILL_CHAR)   ;Byte used in disk formating
        CALL    SEND_DATA

        LD      C,(IY+DATA_FILL_CHAR)  ;     "       "       "
        CALL    SEND_DATA

        LD      C,(IY+SPECIAL_FLAG)    ;Flag byte for cases where after formatting disk need to be initilized. Normally 0, CPM86_FLAG = 1
        CALL    SEND_DATA

        LD      C,(IY+SEC_SKEW_TABLE+1)       ;High address of sector skew table
        CALL    SEND_DATA

        LD      C,(IY+SEC_SKEW_TABLE)  ;Low address of sector skew table
        CALL    SEND_DATA

        LD      C,(IY+FORMAT_NUM)      ;Each format will have a unique number in the table list below.
        CALL    SEND_DATA

        LD      C,(IY+SYS_TRKS)               ;How many tracks for system (usually 2 for 8-inch disks)
        CALL    SEND_DATA

        LD      C,(IY+SEC_SIZE_BYTES+1)       ;Two Bytes. (128,256,512 or 1024)
        CALL    SEND_DATA

        LD      C,(IY+SEC_SIZE_BYTES)
        CALL    SEND_DATA

        LD      C,(IY+TRACK_SIZE+1)    ;Two Bytes. Track size (in bytes) of that disks format
        CALL    SEND_DATA

        LD      C,(IY+TRACK_SIZE)
        CALL    SEND_DATA
                                       ;Now get CPM/DOS data storage values
        LD      C,(IY+DRIVE_TRACK)     ;Current CPM requested Track
        CALL    SEND_DATA

        LD      C,(IY+DRIVE_SECTOR)    ;Current CPM requested Sector
        CALL    SEND_DATA

        LD      C,(IY+DRIVE_SS_DS_FLAG)       ;Current CPM requested Side Flag
        CALL    SEND_DATA
        RET


;--------------- Send Test string back to S-100 system expalining the error codes
```

```
SEND_ERRORS_STRING:                 ;Send back an error string for last ZFDC board error #
        CALL    GET_DATA            ;check if CMD_ABORT was sent

        CP      A,CMD_ABORT
        JR      Z,STRING_DONE

        CP      A,CMD_RANGE_ERR             ;See if we are within range
        JR      C,RANGE_OK
        LD      A,CMD_RANGE_ERR

RANGE_OK:
        LD      HL,ERROR_TBL        ;Point to start of Table Pointers
        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

MORE_STRING:
        LD      A,(HL)              ;Get character
        OR      A,A                 ;0 at end of string
        JR      Z,STRING_DONE
        LD      C,A
        CALL    SEND_DATA           ;Send data back to S-100 system
        INC     HL
        JR      MORE_STRING

STRING_DONE:
        LD      C,NO_ERRORS_FLAG
        JP      S100_ACK_ROUTINE




;===================  Support Routines ================================================

; WD2793_WAIT, just waits for the WD2793 chip to go not busy, or it times out.
; IF, not busy it then checks the WD2793 status register against the error mask bits in [D]
; Returns Z if no errors, NZ & 0FFH in [A] if Timeout, else Masked status error bits in [A].

WD2793_WAIT:
        CALL    DELAY_30uS          ;We need to delay 30uS, previous command may have been an output to CMD port
        LD      BC,0                ;been an output to CMD port
        LD      E,STATUS_DELAY      ;Timeout, (about 5 seconds)
WAIT_1: IN      A,(WD2793_STATUS)   ;Wait until chip is not busy
        BIT     0,A                 ;Check Busy bit
        JR      Z,CHECK_STATUS      ;Z flag set if OK
```

```
        DJNZ    WAIT_1                  ;Try for ~0.5 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
        RET                             ;Return NZ flag set if timeout, 0FFH in [A]
CHECK_STATUS:
        AND     A,D                     ;Check against mask error bits for this command (I,II or III)
        RET



; General ERROR reporting/ACK byte back to S-100 System.

S100_ACK_ROUTINE:
        LD      A,C

        PUSH    AF
        LD      (@CURRENT_ACK),A        ;Store it here also
        OR      A,A
        CALL    NZ,DisplayACK           ;If in display mode also show CMD ACK data
        POP     AF

        CP      A,NO_ERRORS_FLAG
        JR      Z,ACK1
        CALL    SEND_DATA               ;Send back ERROR CODE to S-100 System
        XOR     A                       ;Return NZ Flag set = Errors
        DEC     A
        RET

ACK1:   CALL    SEND_DATA               ;Send back NO_ERRORS_CODE to S-100 System
        XOR     A                       ;Return Z Flag set = NO Errors
        RET


; ==================  Main routine to get data a data byte in [A] from S-100 System via PIO #1 Port B

INPUT_INTS:                             ;Input INT routine (Data FROM S-100 System) when data arrives
        DI                              ;Stop any more ints temporly
        EX      AF,AF'
        XOR     a,a
        DEC     a
        LD      (@IN_DATA_FLAG),a       ;0FFH in flag to indicate we have data in port B for GET_DATA below
        EX      AF,AF'                  ;Switch back to [A]
```

```
        EI                          ;Must turn back on for OUTPUT_INTS (see below)
        RETI                        ;Note PIO is an automatic NON-EOI type of device, this resets the ready line.

GET_DATA:
        CALL    DIRECTION_IN        ;Set direction for input
GET_DATA1:
        LD      A,(@IN_DATA_FLAG)   ;<<<<<<<<< LOOP IF NOTHING >>>>>>>>>
        OR      A,A
        JR      Z,GET_DATA1         ;Loop until we get something (>>> Add watchdog later <<<<)
        DI                          ;Don't want to INT's while here since we are resetting flag
        XOR     A,A
        LD      (@IN_DATA_FLAG),A   ;Set flag to 0. Will go to 0FFH with new data comming in from S100 side (INPUT_INTS)
        IN      A,(PIO1_DATA_B)             ;Get actual data (OK with above flag set because INT's still not active
        EI                          ;Now allow another INT, if more data is to be gotten or sent to S100 side
        RET
;

; ================  Main routine to send data in [C] to S-100 System via PIO #1 Port A
;
OUTPUT_INTS:                        ;Note. Int is generated AFTER data is actully recieved on S100 side (not when placed in the PIO)
        DI
        EX      AF,AF'              ;Switch to [A']
        XOR     a,a
        LD      (@OUT_DATA_FLAG),a  ;Flag data has been recieved by S100 system. Put 0H because all RAM is initilized to 0H
        EX      AF,AF'              ;Switch back to [A]
        EI                          ;Must be on for INPUT_INTS
        RETI                        ;Note PIO is an automatic NON-EOI type of device.

                                    ;VIP --- Flags must be returned unchanged for SEND_DATA!
SEND_DATA:                          ;Send byte of data to S-100 Bus. WAIT - UNTIL acknowledged!
        PUSH    AF                  ;Save Flags
        CALL    DIRECTION_OUT       ;Set direction for output
        DI                          ;Don't want to INT's while here since we are resetting a flag
        XOR     A,A
        DEC     A
        LD      (@OUT_DATA_FLAG),A  ;Set flag to 0FFH. Will go to 0 when byte is recieved on the S100 side, see (INPUT_INTS) below
        LD      A,C                 ;Remember data always in [C]!
        OUT     (PIO1_DATA_A),A             ;First put it in output port
        EI                          ;Allow INT's back on
        XOR     A,A
        LD      (@WAIT_FLAG),A      ;To avoid the ZFDC getting hung up waiting for the S-100 system to acknowledge
SEND_DATA1:                         ;a sent data byte from here, put in a timeout loop.
        LD      A,(@WAIT_FLAG)      ;Note if @WAIT_FLAG is ever 0FFH, then SEND_DATA timed out
        INC     A
        JR      Z,SEND_ABORT        ;0,1,2,3... FFH,->0
        LD      (@WAIT_FLAG),A
        LD      A,(@OUT_DATA_FLAG)  ;Wait until S-100 signals it has actully recieved the data
        OR      A,A
        JR      NZ,SEND_DATA1       ;Previous byte not yet recieved by S100 system ---- so wait!
```

```
        CALL    DIRECTION_IN            ;Back to default IN mode -- no matter what
        POP     AF                      ;Important, some routines count on [AF] being unchanged
        RET
                                        ;A hangup here usually occours because the S-100 recieving end expected
                                        ;less data that the ZFDC sent, so the extra data (or "S100_ACK_ROUTINE" byte)
SEND_ABORT:                             ;is never recieved/acknowledged correctly.
        LD      A,0EEH                  ;Flag to let user know (Note for a properly written BIOS this should never occour)
        CALL    DisplayHEX_01
        LD      A,0E1H                  ;Flag to let user know (Note for a properly written BIOS this should never occour)
        CALL    DisplayHEX_23
        CALL    DELAY_1S                ;Allow time to show info
        CALL    DIRECTION_IN            ;Back to default IN mode -- no matter what
        POP     AF                      ;Important, some routines count on [AF] being unchanged
        RET


DIRECTION_IN:                           ;Set Direction flag to input
        PUSH    AF
        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        RES     DIRECTION_BIT,A                 ;Low for input mode
        OUT     (PIO2_DATA_B),A
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)
        POP     AF
        RET

DIRECTION_OUT:                          ;Set Direction flag to output
        PUSH    AF
        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        SET     DIRECTION_BIT,A                 ;High for output mode
        OUT     (PIO2_DATA_B),A
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)
        POP     AF
        RET


DisplayTrack:                           ;Display the current Track register of the WD2793
        LD      A,(@DEBUG_FLAG)                 ;Are we in dubug mode?
        OR      A,A
        RET     NZ                      ;If so skip display
        IN      A,(WD2793_TRACK)
        CALL    DisplayHEX_01
        RET
DisplayCMD:
        LD      A,(@CURRENT_CMD)
        CALL    DisplayHEX_01
        RET

DisplayHEX_01:                          ;Shortcut to immediatly dump [A] on Track TILs
        PUSH    AF                      ;Change nothing
```

```
        PUSH    BC
        PUSH    AF                  ;Do Low Nibble first
        AND     A,0FH               ;Get low byte first
        LD      B,11010000B         ;To lower strobe for TIL 1
        OR      A,B
        OUT     PIO2_DATA_A,A
        LD      B,11110000B         ;Raise strobe for TIL 1
        OR      A,B
        OUT     PIO2_DATA_A,A       ;Data locked into TIL 1

        POP     AF                  ;Get back stored [A], High Nibble
        RRA
        RRA
        RRA
        RRA
        AND     A,0FH
        LD      B,11100000B         ;To lower strobe for TIL 0
        OR      A,B
        OUT     PIO2_DATA_A,A
        LD      B,11110000B         ;Raise strobe for TIL 0
        OR      A,B
        OUT     PIO2_DATA_A,A       ;Data locked into TIL 0
        POP     BC
        POP     AF
        RET




DisplaySector:                      ;Display the current Sector register of the WD2793
        LD      A,(@DEBUG_FLAG)             ;Are we in dubug mode? If so skip
        OR      A,A
        RET     NZ                  ;if so skip display
        IN      A,(WD2793_SECTOR)
        CALL    DisplayHEX_23
        RET
DisplayACK:
        LD      A,(@CURRENT_ACK)
        CALL    DisplayHEX_23
        RET

DisplayHEX_23:                      ;Shortcut to immediatly dump [A] on Sector TILs
        PUSH    AF                  ;Change nothing
        PUSH    BC
        PUSH    AF
        AND     A,0FH               ;Get low Nibble first
        LD      B,01110000B         ;To lower strobe for TIL 1
        OR      A,B
        OUT     PIO2_DATA_A,A
```

```
        LD      B,11110000B             ;Raise strobe for TIL 1
        OR      A,B
        OUT     PIO2_DATA_A,A           ;Data locked into TIL 1

        POP     AF                      ;Get back stored [A]
        RRA
        RRA
        RRA
        RRA
        AND     A,0FH
        LD      B,10110000B             ;To lower strobe for TIL 0
        OR      A,B
        OUT     PIO2_DATA_A,A
        LD      B,11110000B             ;Raise strobe for TIL 0
        OR      A,B
        OUT     PIO2_DATA_A,A           ;Data locked into TIL 0
        POP     BC
        POP     AF
        RET

                                        ;Display current Drive on TIL #4
DisplayDrive:                           ;Drive # can be from 0 to 3H in [A]
        PUSH    BC
        PUSH    AF
        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        AND     A,0FCH
        LD      C,A
        POP     AF
        AND     A,3H                    ;Isolate lower 2 bits (drive selection bits)
        OR      A,C                     ;Add in other bits
        OUT     PIO2_DATA_B,A           ;Data locked into TIL 3
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)
        POP     BC
        RET


; Turn ON the WD2795 INTRQ ints. Set the NMI jump address to the pointer in[HL]
; No registers changed

ON_WD2793_INTRQ:
        PUSH    AF
        LD      (@INTR_ADDRESS),HL      ;Put the NMI return address here

        LD      A,(@PIO2B_BITS)                 ;Get stored hardware select bits (because this port cannot be read)
        SET     WD_INTRQ_BIT,A          ;Set WD2793 INTRQ
        OUT     PIO2_DATA_B,A           ;send to hardware
        LD      (@PIO2B_BITS),A                 ;Store hardware select bits (because this port cannot be read)

        POP     AF
```

```
        RET

; Turn OFF the WD2793 INTRQ ints. Set the NMI jump address to the pointer in[HL]
; No registers changed

OFF_WD2793_INTRQ:
        PUSH    AF

        LD      A,(@PIO2B_BITS)               ;Get stored hardware select bits (because this port cannot be read)
        RES     WD_INTRQ_BIT,A        ;Reset WD2793 INTRQ
        OUT     PIO2_DATA_B,A
        LD      (@PIO2B_BITS),A               ;Store hardware select bits (because this port cannot be read)

        LD      HL,DEFAULT_NMI        ;Put back the catch-all NMI trap in case there was an NMI/Watach-dog problem
        LD      (@INTR_ADDRESS),HL    ;Put the default NMI routine address here
        POP     AF
        RET
;
;
; Very short Software time delay ~30 uSec @ 6MHZ (No registers changed)
DELAY_30uS:
        PUSH    AF
        LD      A,20H
WAIT_5: DEC     A
        JR      NZ,WAIT_5
        POP     AF
        RET

; Software time delay for ROLL Hex display
DELAY_100mS:
        PUSH    AF
        PUSH    BC
        LD      BC,0005H              ;Delay adjust
WAIT_6: DJNZ    WAIT_6
        DEC     C
        JR      NZ,WAIT_6
        POP     BC
        POP     AF
        RET

; Software time delay for waiting for board initilization
DELAY_200mS:
        PUSH    AF
        PUSH    BC
        LD      BC,0010H              ;Delay adjust
WAIT_7: DJNZ    WAIT_7
        DEC     C
        JR      NZ,WAIT_7
        POP     BC
```

```
        POP     AF
        RET

; Software time delay ~1 Sec @ 6MHZ
DELAY_1S:
        PUSH    AF
        PUSH    BC
        LD      A,2H
        LD      BC,0                    ;Delay
        LD      E,4                     ;Timeout, (about 1 second @ 6 MHz)
WAIT_4: DJNZ    WAIT_4                  ;Delay for ~0.5 seconds
        DEC     B                       ;Reset B to 0FFH
        DEC     C
        JR      NZ,WAIT_4
        DEC     A
        JR      NZ,WAIT_4
        POP     BC
        POP     AF
        RET

; Software time delay ~2 Sec @ 6MHZ
DELAY_2S:
        PUSH    AF
        PUSH    BC
        LD      A,10H
        LD      BC,0                    ;Delay
        LD      E,4                     ;Timeout, (about 2 seconds @ 6 MHz)
WAIT_3: DJNZ    WAIT_3                  ;Delay for ~0.5 seconds
        DEC     B                       ;Reset B to 0FFH
        DEC     C
        JR      NZ,WAIT_3
        DEC     A
        JR      NZ,WAIT_3
        POP     BC
        POP     AF
        RET


; Use Sector Display to display debug info of what is in [A]
DEBUG:  PUSH    BC
        PUSH    DE
        PUSH    HL
        Call    DisplayHEX_23
        POP     HL
        POP     DE
        POP     BC
        RET

;-------------------------- Internal Zapple like monitor. Used for debugging. -----------------
```

```
;       NOTE TABLE MUST BE WITHIN 0-FFH BOUNDRY

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

;MONITOR COMMAND BRANCH TABLE
TBL:    DW  MEMMAP    ; "A"  DISPLAY A MAP OF MEMORY
        DW  START     ; "B"
        DW  START     ; "C"
        DW  DISP      ; "D"  DISPLAY MEMORY (IN HEX & ASCII)
        DW  ECHO      ; "E"  ECHO CHAR IN TO CHAR OUT
        DW  FILL      ; "F"  FILL MEMORY WITH A CONSTANT
        DW  GOTO      ; "G"  GO TO [ADDRESS]
        DW  MATH      ; "H"  HEX MATH (GIVES SUM & DIFFERENCE OF TWO NOS.)
        DW  START     ; "I"
        DW  RAMTEST   ; "J"  NON-DESTRUCTIVE MEMORY TEST
        DW  KCMD      ; "K"  DISPLAY THE MENU
        DW  START     ; "L"
        DW  MOVE      ; "M"  MOVE BLOCK OF MEMORY (START,FINISH,DESTINATION)
        DW  START     ; "N"
        DW  START     ; "O"
        DW  START     ; "P"
        DW  QUERY     ; "Q"  QUERY PORT (IN OR OUT)
        DW  WD_REGS   ; "R"  READ WD2793 REGS
        DW  SUBS      ; "S"  SUBSTITUTE &/OR EXAMINE MEMORY
        DW  TYPE      ; "T"  TYPE ASCII PRESENT IN MEMORY
        DW  START     ; "U"
        DW  VERIFY    ; "V"  COMPARE MEMORY
        DW  START     ; "W"
        DW  SHOW_REGS ; "X"  DISPLAY Z80 Alt Register Set
        DW  WHERE     ; "Y"  FIND A SEQUENCE OF BYTES IN MEMORY
        DW  SIZE      ; "Z"  FIND HIGHEST R/W RAM

MONITOR:
        LD      HL,MSG0         ;Have a Stack, so we can use CALL
        CALL    TOMM

START:  LD      DE,START
        PUSH    DE              ;EXTRA UNBALANCED POP & [DE] WOULD END UP IN [PC]
        CALL    CRLF
        LD      C,'-'
        CALL    SEND_DATA
        LD      C,'>'
        CALL    SEND_DATA

STARO:  CALL    TI              ;Main loop. Monitor will stay here until cmd.
        AND     7FH
        JR      Z,STARO
```

```
        CP      ESC             ;Esc will abort monitor back to WD2793 Chip
        JP      Z,LOOP          ;<<<--- Back to main LOOP
        SUB     41H
        RET     M
        CP      1AH
        RET     NC
        ADD     A,A
        LD      HL,TBL
        ADD     A,L
        LD      L,A
        LD      A,(HL)
        INC     HL
        LD      H,(HL)
        LD      L,A
        LD      C,02H
        JP      (HL)    ;JUMP TO COMMAND TABLE

;----- GO CARRY OUT COMMAND AND POP BACK TO START----------

;SEND MESSAGE TO CONSOL MESSAGE IN [HL],LENGTH IN [B]
TOM:    LD      C,(HL)
        INC     HL
        CALL    SEND_DATA
        DJNZ    TOM
        RET
;
TOMM:   LD      A,(HL)          ;A ROUTINE TO PRINT OUT A STRING @ [HL]
        INC     HL              ;UP TO THE FIRST '$'.
        CP      '$'
        RET     Z
        LD      C,A
        CALL    SEND_DATA
        JR      TOMM
;
;RESTORE SYSTEM AFTER ERROR
ERROR:  CALL    MEMSIZ          ;GET RAM AVAILABLE - WORKSPACE IN [HL]
        LD      SP,HL           ;SET STACK UP IN WORKSPACE AREA
        LD      C,'*'
        CALL    SEND_DATA
        JP      START
;
;PRINT HIGHEST MEMORY FROM BOTTOM
SIZE:   CALL    MEMSIZ          ;RETURNS WITH [HL]= RAM AVAILABLE-WORKSPACE
;
LFADR:  CALL    CRLF
;
;PRINT [HL] AND A SPACE
HLSP:   PUSH    HL
        PUSH    BC
```

```
        CALL    LADR
        LD      C,SPACE
        CALL    SEND_DATA
        POP     BC
        POP     HL
        RET
;
;PRINT A SPACE
SF488:  LD      C,SPACE
        JP      SEND_DATA
;
;CONVERT HEX TO ASCII & PUT IT IN [C]
CONV:   AND     0FH
        ADD     A,90H
        DAA
        ADC     A,40H
        DAA
        LD      C,A
        RET
;
;GET TWO PARAMETERS AND PUT THEM IN [HL] & [DE] THEN CRLF
EXLF:   CALL    HEXSP
        POP     DE
        POP     HL
;
;SEND TO CONSOL CR/LF
CRLF:   PUSH    BC
        LD      C,LF
        CALL    SEND_DATA
        LD      C,CR
        CALL    SEND_DATA
        POP     BC
        RET
;
;
;PUT THREE PARAMETERS IN [BC] [DE] [HL] THEN CR/LF
EXPR3:  INC     C               ;ALREADY HAD [C]=2 FROM START
        CALL    HEXSP
        CALL    CRLF
        POP     BC
        POP     DE
        POP     HL
        RET

;GET ONE PARAMETER
EXPR1:  LD      C,01H
HEXSP:  LD      HL,0000
EX0:    CALL    TI
EX1:    LD      B,A
```

```
        CALL    NIBBLE
        JR      C,EX2X
        ADD     HL,HL
        ADD     HL,HL
        ADD     HL,HL
        ADD     HL,HL
        OR      L
        LD      L,A
        JR      EX0
EX2X:   EX      (SP),HL
        PUSH    HL
        LD      A,B
        CALL    QCHK
        JR      NC,SF560
        DEC     C
        RET     Z
SF560:  JP      NZ,ERROR
        DEC     C
        JR      NZ,HEXSP
        RET
EXF:    LD      C,01H
        LD      HL,0000H
        JR      EX1
;
;RANGE TEST ROUTINE CARRY SET = RANGE EXCEEDED
HILOX:  CALL    HILO
        RET     NC
        POP     DE              ;DROP ONE LEVEL BACK TO START
        RET
HILO:   INC     HL              ;RANGE CHECK SET CARRY IF [DE]=[HL]
        LD      A,H
        OR      L
        SCF
        RET     Z
        LD      A,E
        SUB     L
        LD      A,D
        SBC     A,H
        RET
;
;PRINT [HL] ON CONSOL
LADR:   LD      A,H
        CALL    LBYTE
        LD      A,L
LBYTE:  PUSH    AF              ;Print [A] on Consol
        RRCA
        RRCA
        RRCA
        RRCA
```

```
        CALL    SF598
        POP     AF
SF598:  CALL    CONV            ;Return with ASCII in [C]
        JP      SEND_DATA
;
;THIS IS A CALLED ROUTINE USED TO CALCULATE TOP OF RAM IS USED BY
;THE ERROR TO RESET THE STACK
;
MEMSIZ: PUSH    BC              ;SAVE [BC]
MEMSZ1: LD      HL,0FFFFH       ;START FROM THE TOP DOWN
MEMSZ2: LD      A,(HL)
        CPL
        LD      (HL),A
        CP      (HL)
        CPL                     ;PUT BACK WHAT WAS THERE
        LD      (HL),A
        JP      Z,GOTTOP
        DEC     H               ;TRY 100H BYTES LOWER
        JR      MEMSZ2          ;KEEP LOOKING FOR RAM
GOTTOP: POP     BC              ;RESTORE [BC]
        RET
;
NIBBLE: SUB     30H
        RET     C
        CP      17H
        CCF
        RET     C
        CP      LF
        CCF
        RET     NC
        SUB     07H
        CP      LF
        RET
;
COPCK:  LD      C,'-'
        CALL    SEND_DATA
;
PCHK:   CALL    TI
;
;TEST FOR DELIMITERS
;
QCHK:   CP      SPACE
        RET     Z
        CP      ','
        RET     Z
        CP      CR
        SCF
        RET     Z
        CCF
```

```
        RET
;
;KEYBOARD HANDELING ROUTINE (WILL NOT ECHO CR/LF)
;IT CONVERTS LOWER CASE TO UPPER CASE FOR LOOKUP COMMANDS
;ALSO ^C WILL FORCE A JUMP TO START OF ZFDC CODE
;ALL OTHERE CHARACTERS ARE ECHOED ON CONSOL
;
TI:     CALL    GET_DATA
        CP      CR
        RET     Z
        CP      'C'-40H         ;^C TO BOOT ZFDC EEPROM
        JP      Z,ROM_START
        PUSH    BC
        LD      C,A
        CALL    SEND_DATA
        LD      A,C
        POP     BC
        CP      40H             ;LC->UC
        RET     C
        CP      7BH
        RET     NC
SF754:  AND     5FH
        RET
;
ZBITS:  PUSH    DE              ;DISPLAY 8 BITS OF [A]
        PUSH    BC
        LD      E,A
        CALL    BITS
        POP     BC
        POP     DE
        RET
;
BITS:   LD      B,08H           ;DISPLAY 8 BITS OF [E]
        CALL    SF488
SF76E:  SLA     E
        LD      A,18H
        ADC     A,A
        LD      C,A
        CALL    SEND_DATA
        DJNZ    SF76E
        RET
;
MEMMAP: CALL    CRLF
        LD      HL,0
        LD      B,1
MAP1:   LD      E,'R'           ;PRINT R FOR RAM
        LD      A,(HL)
        CPL
        LD      (HL),A
```

```
        CP      (HL)
        CPL
        LD      (HL),A
        JR      NZ,MAP2
        CP      (HL)
        JR      Z,PRINT
MAP2:   LD      E,'p'
MAP3:   LD      A,0FFH
        CP      (HL)
        JR      NZ,PRINT
        INC     L
        XOR     A
        CP      L
        JR      NZ,MAP3
        LD      E,'.'
PRINT:  LD      L,0
        DEC     B
        JR      NZ,NLINE
        LD      B,16
        CALL    CRLF
        CALL    HXOT4
NLINE:  LD      A,SPACE
        CALL    OTA
        LD      A,E
        CALL    OTA
        INC     H
        JR      NZ,MAP1
        CALL    CRLF
        CALL    CRLF
        JP      START
;
;16 HEX OUTPUT ROUTINE
;
HXOT4:  LD      C,H
        CALL    HXO2
        LD      C,L
HXO2:   LD      A,C
        RRA
        RRA
        RRA
        RRA
        CALL    HXO3
        LD      A,C
HXO3:   AND     0FH
        CP      10
        JR      C,HADJ
        ADD     A,7
HADJ:   ADD     A,30H
OTA:    PUSH    BC
```

```
        LD      C,A
        CALL    SEND_DATA       ;SEND TO CONSOL
        POP     BC
        RET
;
;DISPLAY MEMORY IN HEX
;
DISP:   CALL    EXLF            ;GET PARAMETERS IN [HL],[DE]
        LD      A,L             ;ROUND OFF ADDRESSES TO XX00H
        AND     0F0H
        LD      L,A
        LD      A,E             ;FINAL ADDRESS LOWER HALF
        AND     0F0H
        ADD     A,10H           ;FINISH TO END 0F LINE
SF172:  CALL    LFADR
SF175:  CALL    BLANK
        LD      A,(HL)
        CALL    LBYTE
        CALL    HILOX
        LD      A,L
        AND     0FH
        JR      NZ,SF175
        LD      C,TAB           ;INSERT A TAB BETWEEN DATA
        CALL    SEND_DATA
        LD      B,4H            ;ALSO 4 SPACES
TA11:   LD      C,SPACE
        CALL    SEND_DATA
        DJNZ    TA11
        LD      B,16            ;NOW PRINT ASCII (16 CHARACTERS)
        PUSH    DE              ;TEMPORLY SAVE [DE]
        LD      DE,0010H
        SBC     HL,DE
        POP     DE
T11:    LD      A,(HL)
        AND     7FH
        CP      ' '             ;FILTER OUT CONTROL CHARACTERS'
        JR      NC,T33
T22:    LD      A,'.'
T33:    CP      07CH
        JR      NC,T22
        LD      C,A             ;SET UP TO SEND
        CALL    SEND_DATA
        INC     HL
        DJNZ    T11             ;REPEAT FOR WHOLE LINE
        JR      SF172
;
BLANK:  LD      C,' '
        JP      SEND_DATA
;
```

```
;INSPECT AND / OR MODIFY MEMORY
;
SUBS:   LD      C,1
        CALL    HEXSP
        POP     HL
SF2E3:  LD      A,(HL)
        CALL    LBYTE
        LD      C,'-'
        CALL    SEND_DATA
        CALL    PCHK
        RET     C
        JR      Z,SF2FC
        CP      5FH
        JR      Z,SF305
        PUSH    HL
        CALL    EXF
        POP     DE
        POP     HL
        LD      (HL),E
        LD      A,B
        CP      CR
        JP      Z,SUB_DONE
SF2FC:  INC     HL
SF2FD:  LD      A,L
        AND     07H
        CALL    Z,LFADR
        JR      SF2E3
SF305:  DEC     HL
        JR      SF2FD
SUB_DONE:
        CALL    CRLF
        RET
;
;FILL A BLOCK OF MEMORY WITH A VALUE
;
FILL:   CALL    EXPR3
SF1A5:  LD      (HL),C
        CALL    HILOX
        JR      NC,SF1A5
        POP     DE
        JP      START
;
;
; GET OR OUTPUT TO A PORT
;
QUERY:  CALL    PCHK
        CP      'O'             ;OUTPUT TO PORT
        JR      Z,SF77A
        CP      'I'             ;INPUT FROM PORT
```

```
        JP      Z,QQQ1
        LD      C,'*'
        JP      SEND_DATA       ;WILL ABORT IF NOT 'I' OR 'O'
QQQ1:   LD      C,1
        CALL    HEXSP
        POP     BC
        IN      A,(C)
        LD      E,A
        JP      ZBITS
;
SF77A:  CALL    HEXSP
        POP     DE
        POP     BC
        OUT     (C),E
        CALL    CRLF
        RET
;
;
; MEMORY TEST
;
RAMTEST:CALL    EXLF
SF200:  LD      A,(HL)
        LD      B,A
        CPL
        LD      (HL),A
        XOR     (HL)
        JR      Z,SF215
        PUSH    DE
        LD      D,B
        LD      E,A             ;TEMP STORE BITS
        CALL    HLSP
        CALL    BLANK
        LD      A,E
        CALL    ZBITS
        CALL    CRLF
        LD      B,D
        POP     DE
SF215:  LD      (HL),B
        CALL    HILOX
        JR      SF200

;MOVE A BLOCK OF MEMORY TO ANOTHER LOCATION
;
MOVE:   CALL    EXPR3
SF21E:  LD      A,(HL)
        LD      (BC),A
        INC     BC
        CALL    HILOX
        JR      SF21E
```

```
;
;VERIFY ONE BLOCK OF MEMORY WITH ANOTHER
;
VERIFY: CALL    EXPR3
VERIO:  LD      A,(BC)
        CP      (HL)
        JR      Z,SF78E
        PUSH    BC
        CALL    CERR
        POP     BC
SF78E:  INC     BC
        CALL    HILOX
        JR      VERIO
        RET
;
CERR:   LD      B,A
        CALL    HLSP
        LD      A,(HL)
        CALL    LBYTE
        CALL    BLANK
        LD      A,B
        CALL    LBYTE
        JP      CRLF
;
ECHO:   CALL    GET_DATA        ;Routine to check keyboard etc.
        CP      'C'-40H         ;Loop until  ^C
        RET     Z
        CP      'Z'-40H
        RET     Z
        LD      C,A
        CALL    SEND_DATA
        JR      ECHO
;
;
; GET SUM & DIFFERENCE OF TWO HEX NOS
;
MATH:   LD      HL,MSG9
        CALL    TOMM
        LD      C,2             ;2 PARAMETERS REQ
        CALL    EXLF            ;GET DATA IN [DE] & [HL]
        PUSH    HL              ;SAVE [HL] FOR LATER
        ADD     HL,DE
        PUSH    HL              ;SAVE THIS [HL] ALSO
        LD      HL,MSG6         ;'SUM= '
        CALL    TOMM
        POP     HL              ;PUT SUM IN [HL]
        CALL    HLSP            ;PRINT [HL]
        POP     HL              ;GET BACK FIRST [HL]
        OR      A               ;CLEAR CARRY
```

```
        SBC     HL,DE           ;GET DIFFERENCE
        PUSH    HL              ;SAVE IT
        LD      HL,MSG7         ;'DIFFERENCE='
        CALL    TOMM
        POP     HL              ;GET BACK DIFFERENCE IN [HL]
        JP      HLSP            ;PRINT [HL]
;
;
;SEARCH FOR AN ASCII STRING IN MEMORY
;
WHERE:  PUSH    IX              ;dangerous! but we need it
        LD      D,00H           ;This is from Zapple monitor have not had time
SF32A:  LD      C,1             ;to modify!
        CALL    HEXSP
        POP     HL
        LD      H,L
        PUSH    HL
        INC     SP
        INC     D
        LD      A,B
        SUB     0DH
        JR      NZ,SF32A
        LD      B,A
        LD      C,A
        LD      H,A
        LD      L,D
        DEC     L
        ADD     HL,SP
        PUSH    HL
        PUSH    BC
FINDC:  PUSH    BC
        CALL    CRLF
        POP     BC
FIND:   POP     HL
        POP     IX
        LD      E,D
        LD      A,(IX+0)
        CPIR
        JP      PO,DONE2
        PUSH    IX
        PUSH    HL
FOUND:  DEC     E
        JR      Z,TELL
        LD      A,(IX-1)
        CP      (HL)
        JR      NZ,FIND
        INC     HL
        DEC     IX
        JR      FOUND
```

```
TELL:   POP     HL
        PUSH    HL
        DEC     HL
        PUSH    BC
        CALL    LADR
        POP     BC
        JR      FINDC
DONE2:  INC     SP
        DEC     E
        JR      NZ,DONE2
        POP     IX              ;get IX back
        RET
;
;
;
;PRINT SIGNON AND MENU STRING ON CRT
;
KCMD:   LD      HL,MSG0         ;Signon Msg again
        CALL    TOMM
        LD      HL,MENUMSG      ;Then Menu Message
        JP      TOMM
;
;
;READ ASCII FROM MEMORY
;
TYPE:   CALL    EXLF
SF30B:  CALL    LFADR
        LD      B,56
SF310:  LD      A,(HL)
        AND     7FH
        CP      SPACE
        JR      NC,SF319
SF317:  LD      A,2EH
SF319:  CP      7CH
        JR      NC,SF317
        LD      C,A
        CALL    SEND_DATA
        CALL    HILOX
        DJNZ    SF310
        JR      SF30B
;
;
BACK:   LD      HL,BACK_MSG     ;Announce we are going back to FD Controller
        CALL    TOMM
        JP      LOOP


;  Go To an Address in RAM
GOTO:   LD      C,1             ;SIMPLE GOTO, FIRST GET PARMS.
```

```
        CALL    HEXSP
        CALL    CRLF
        POP     HL              ;GET PARAMETRE PUSHED BY EXF
        JP      (HL)

; Display All WD2793  Status Registers

 WD_REGS:
        CALL    CRLF
        LD      HL,WD_STATUS            ;First show status register
        CALL    TOMM
        IN      A,(WD2793_BASE)                 ;Get WD 2793 Status Register
        CALL    ZBITS
        CALL    CRLF
        LD      HL,WD_TRACK             ;Next show track register
        CALL    TOMM
        IN      A,(WD2793_BASE+1)
        CALL    LBYTE
        CALL    CRLF
        LD      HL,WD_SECTOR            ;Next show sector register
        CALL    TOMM
        IN      A,(WD2793_BASE+2)
        CALL    LBYTE
        CALL    CRLF
        LD      HL,WD_DATA              ;Next show data register
        CALL    TOMM
        IN      A,(WD2793_BASE+3)
        CALL    LBYTE
        JP      CRLF                    ;Will pop stack back to start.



; Display the Z80 Reg Set

 SHOW_REGS:
        DI                              ;No Ints while this is going on
        PUSH    AF                      ;Save everything just in case
        PUSH    BC
        PUSH    DE
        PUSH    HL

        PUSH    IY                      ;Save in the order we need them
        PUSH    IX
        PUSH    HL
        PUSH    DE
        PUSH    BC
        PUSH    AF

        LD      HL,A_REG                ;Show [A] register
```

```
        CALL    TOMM
        POP     AF
        CALL    LBYTE                   ;Print [A]

        LD      HL,BC_REG               ;Show [BC], register
        CALL    TOMM
        POP     HL                      ;[BC]->[HL]
        CALL    LADR

        LD      HL,DE_REG               ;Show [DE] register
        CALL    TOMM
        POP     HL                      ;[DE]->[HL]
        CALL    LADR

        LD      HL,HL_REG               ;Show [HL] register
        CALL    TOMM
        POP     HL                      ;[HL]->[HL]
        CALL    LADR

        LD      HL,IX_REG               ;Show [IX] register
        CALL    TOMM
        POP     HL                      ;[IX]->[HL]
        CALL    LADR

        LD      HL,IY_REG               ;Show [IY] register
        CALL    TOMM
        POP     HL                      ;[IY]->[HL]
        CALL    LADR

        LD      HL,SP_REG               ;Show [SP] register
        CALL    TOMM
        LD      HL,@TEMP_STACK
        LD      (@TEMP_STACK),SP
        CALL    LADR

        POP     HL
        POP     DE
        POP     BC
        POP     AF
        JP      CRLF                    ;Will pop stack back to start.


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

;       LOOKUP TABLES OF DISK PARAMETERS
; 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 1.4M 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


;-----------------------------------------------------------------------------------------------
;       LOOKUP TABLES OF DISK PARAMETERS
;Note in V2.8 and later, support has been added support for IBM 1.2M/5" and 1.44M/3.5" disk formats.
;As far as the WD2793;is concerned these behave as 8" disks. It's just the disk capacity is larger.
;(Only the old 360K type SD & DD, 300RPM, 5" disks need a clock speed adjustment).
;So whenever I referr to 8" disks, I am also including these 1.2M and 1.4M disks.
;
;To calculate the total formatted track size it is easiest to just build a table (using a previous
;track size). Start formatting a disk and stop with ESC. The total track image can be seen in
;in RAM starting at A000H up to ~ C000H (depending on the format). Look for where the last block of
;GAP4 E5's stop. Subtract A000H from this value and place in your new table. Burn a new EPROM and
;format your disk. This beats calculating bytes!  If you find a new format hangs the WD2793 it's because
;your track size is too long. You can usually chizel a few bytes from the GAP4 region.


;       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
        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



;       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      1380H           ;<--- Size in bytes of 1 formatted track (see top of tables)
        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      9+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      50              ;NBR GAP 3
        DB      180  ;(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      2780H           ;Size in bytes of 1 formatted track
        DB      '8" DDSS, 9 X 1024 Byte Sectors.',0
SKEW_1K:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H
```

```
;
;       8" DOUBLE DENSITY (1024 BYTE SECTORS - Double Sided)
DDT1K2: DB      00001100B       ;Disk HW_BYTE (DDDS) ;
        DB      9+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      50              ;NBR GAP 3
        DB      180  ;(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, 9 X 1024 Byte Sectors.',0
SKEW_1KDS:
        DB      1H,2H,3H,4H,5H,6H,7H,8H,9H

;
;---------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 X2
        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      0E5H            ;Data area fill character
        DB      0H              ;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      0E5H            ;Data area fill character
        DB      0H              ;Special formating modifications of disk req (+++ NOT DONE YET)
        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", 15 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", 15 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

;-----------------------------------------
MSG0:           DB SCROLL,QUIT,NO_ENHANCEMENT,FAST,BELL,CR,LF,LF
                DB 'ZFDC ROM MONITOR V3.5  (John Monahan, 08/20/2011) $'
BACK_MSG:       DB 'Returning control back to WD2793 Chip',CR,LF,'$'
MSG6:           DB CR,LF
                DB 'SUM = $'
MSG7:           DB ' DIFF = $'
MSG8:           DB '$'
MSG9:           DB CR,LF
                DB 'HEX MATH, ENTER:- $'
MSG12           DB '    $'

WD_STATUS:      DB 'WD2793 Status Register = $'
WD_TRACK:       DB 'WD2793 Track Register = $'
WD_SECTOR:      DB 'WD2793 Sector Register = $'
WD_DATA:        DB 'WD2793 Data Register = $'

A_REG:          DB  CR,LF,'[A]  = $'
BC_REG:         DB  CR,LF,'[BC] = $'
DE_REG:         DB  CR,LF,'[DE] = $'
HL_REG:         DB  CR,LF,'[HL] = $'
IX_REG:         DB  CR,LF,'[IX] = $'
IY_REG:         DB  CR,LF,'[IY] = $'
SP_REG:         DB  CR,LF,'[SP] = $'

MENUMSG:        DB CR,LF
                DB 'A=Memmap        D=Disp Memory      E=Echo Keyboard    F=Fill Memory',CR,LF
                DB 'G=GoTo RAM      H=Hex Math         J=Test RAM         K=Display Menu',CR,LF
                DB 'M=Move Memory   Q=Query Port       R=WD2793 Regs      S=Substitute',CR,LF
                DB 'T=Type ASCII    V=Verify Memory    X=Z80 Regs         Y=Find Bytes',CR,LF
                DB 'Z=Top of RAM    ESC=Back to WD2793 Chip',CR,LF,'$'
```

```
;

;       NOTE TABLE MUST BE WITHIN 0-FFH BOUNDRY

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

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

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

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

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

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

         DW       PT_ID_ERR_HUNG              ;18H,  WD2793 Timeout Error after READ TRACK ID Command
         DW       PT_ID_ERR1           ;19H,  Track ID Error, DRIVE NOT READY ERROR
         DW       PT_ID_ERR2           ;1AH,  Track ID Error, RNF ERROR
         DW       PT_ID_ERR3           ;1BH,  Track ID Error, LOST DATA ERROR
         DW       PT_ID_ERR4           ;1CH,  Track ID Error, CRC ERROR

         DW       PT_RS_HUNG           ;1DH,  WD2793 Timeout Error after Read Sector Command was sent
         DW       PT_RS_ERR1           ;1EH,  Sector read error, DRIVE NOT READY ERROR
         DW       PT_RS_ERR2           ;1FH,  Sector read error, RNF ERROR
         DW       PT_RS_ERR3           ;20H,  Sector read error, LOST DATA ERROR
         DW       PT_RS_ERR4           ;21H,  Sector read error, CRC ERROR
         DW       PT_RS_SK_TRK_HUNG    ;22H,  WD2793 Timeout Error after SEEK within READ SECTOR Command
         DW       PT_RS_RES_HUNG              ;23H,  WD2793 Timeout Error after RESTORE command within READ SECTOR Command
```

```
        DW      PT_RS_RES_ERR               ;24H,   Restore to track 0, DRIVE NOT READY ERROR within READ SECTOR Command
        DW      PT_RS_SKTRK_ERR1    ;25H,   Seek to track error, DRIVE NOT READY ERROR bit set within READ SECTOR Command
        DW      PT_RS_SKTRK_ERR2    ;26H,   Seek to track error with SEEK ERROR bit set within READ SECTOR Command

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

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

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

       DW      PT_DRIVE_INACTIVE    ;3AH    Drive is inactive
       DW      PT_DRIVE_DOOR         ;3BH    Drive door open
       DW      PT_BUFFER_OVERFLOW   ;3CH    Too many sectors were requested in a multi sector R/W

       DW      PT_ABORT_FLAG        ;3DH    Special error flag to signify the user aborted a command
        DW      PT_CMD_RANGE_ERR    ;3EH,   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,'Invalid Drive. (Must be 0,1,2,or 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_BUFFER_OVERFLOW    DB     CR,LF,BELL, 'Two many sectors were requested in a single multi sector R/W request',0

PT_CMD_RANGE_ERR      DB     CR,LF,BELL, 'Error = 3DH,  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


;END
```