

```

; ASSEMBLER ASIDE 1.14
; _FDCMON
;
; FLOPPY DISK UTILITY PROGRAM
; AUTHOR: ANDREW LYNCH
; CODE RE-ORGANIZED BY DAN WERNER
; CODE FOR THE ASSEMBLER ASIDE 1.14 BY ROLF HARRMANN
; HR = ROLF HARRMANN
;
;
; DATA CONSTANTS
;
; REGISTER      IO PORT      ; FUNCTION
FMSR:          EQU          036H      ; ADDRESS OF MAIN STATUS REGISTER
FDATA:         EQU          037H      ; FLOPPY DATA REGISTER
FLATCH:        EQU          03AH      ; FLOPPY CONFIGURATION LATCH
FDMA:          EQU          03CH      ; PSEUDO DMA ADDRESS

; CODE BY HR
; 74LS574
FLATCH0:       EQU          00000001B ; TC      HIGH D0
FLATCH1:       EQU          00000010B ; /MOTOR LOW D1
FLATCH2:       EQU          00000100B ; MINI   HIGH D2
FLATCH3:       EQU          00001000B ; SETPS2 D3
FLATCH4:       EQU          00010000B ; SETPS1 D4
FLATCH5:       EQU          00100000B ; SETPS0 D5
FLATCH6:       EQU          01000000B ; /REDWC D6
FLATCH7:       EQU          10000000B ; /DISKCHG_L D7
; *****
; CODE BY HR
MOTOR_ON_OFF:  EQU          1
; *****

;
; FDC CONFIGURATION LATCH OUTPUT BIT PATTERNS
MOTOR:         EQU          %00000000 ; BIT PATTERN IN LATCH FOR MOTOR CONTROL (ON)

TERMCN:        EQU          %00000001 ; BIT PATTERN IN LATCH TO WRITE A TC STROBE

RESETL:        EQU          %00000010 ; BIT PATTERN IN LATCH TO RESET ALL BITS
MINI:          EQU          %00000100 ; BIT PATTERN IN LATCH TO SET MINI MODE FDC9229 LOW DENS=1, HIGH DENS=0
PRECOMP:       EQU          %00100000 ; BIT PATTERN IN LATCH TO SET WRITE PRECOMP 125 NS:
FDDENSITY:     EQU          %01000000 ; BIT PATTERN IN LATCH TO FLOPPY LOW DENSITY (HIGH IS 0)
FDREADY:       EQU          %10000000 ; BIT PATTERN IN LATCH TO FLOPPY READY (P-34):

;
; CP/M STRING RELATED CONSTANTS
CR:            EQU          0Dh        ; CARRIAGE RETURN CHARACTER
LF:            EQU          0Ah        ; LINE FEED CHARACTER
ESC:           EQU          1Bh

; CODE BY HR
; ENDS instead of END because the Assembler does not allow END
ENDS:          EQU          '$'        ; LINE TERMINATOR FOR CP/M STRINGS
; *****
BS:            EQU          08H        ; ASCII backspace character
;
;
; MAIN PROGRAM BEGINS HERE
;
;
; ORG          0100h
;
; CALL         SETUPDRIVE      ; SETUP DRIVE PARAMETERS
; CALL         OUTFLATCH      ; OUTPUT TO CONTROLLER

MAIN_LOOP:
LD             DE,MSG_START      ;
LD             C,09H            ; CP/M WRITE START STRING TO CONSOLE CALL
CALL          0005H            ;
LD             C,01H            ; CP/M console input call
CALL          0005H            ; GET K.B. DATA
CP             '1'              ; MOTOR ON

```

```

        JP      Z,MENU_MOTOR_ON      ;
        CP      '2'                  ; MOTOR OFF
        JP      Z,MENU_MOTOR_OFF    ;
        CP      '3'                  ; SET TRACK
        JP      Z,MENU_SET_TRACK     ;

; CODE BY HR
;*****
        CP      '4'                  ; FORMAT TRACK
        JP      Z,YES_NO_FORMAT      ; SURE YES OR NO
;*****
        CP      '5'                  ; READ SECTOR
        JP      Z,MENU_READ          ;

; CODE BY HR
;*****
        CP      '6'                  ; WRITE SECTOR
        JP      Z,YES_NO_WRITE       ; SURE YES OR NO
;*****
        CP      '7'                  ; DUMP BUFFER
        JP      Z,MENU_DUMP          ;
        CP      '8'                  ; SENSE INT
        JP      Z,MENU_SENSE         ;
        CP      '9'                  ; SENSE INT
        JP      Z,MENU_CLEAR         ;
        CP      'Q'                  ; IS QUIT
        JP      Z,MENU_QUIT          ;
        CP      'q'                  ; IS QUIT
        JP      Z,MENU_QUIT          ;
        JP      MAIN_LOOP            ; LOOP TO MENU

; CODE BY HR
;*****
YES_NO_FORMAT:
        LD      DE,MSG_YES_NO        ;
        LD      C,09H                ; CP/M WRITE START STRING TO CONSOLE CALL
        CALL    0005H                ;
        LD      C,01H                ; CP/M console input call
        CALL    0005H                ; GET K.B. DATA
        CP      'Y'                  ;
        JP      Z,MENU_FORMAT        ;
        CP      'Y'                  ;
        JP      Z,MENU_FORMAT        ;
        CP      'Q'                  ; IS QUIT
        JP      Z,MENU_QUIT          ;
        CP      'q'                  ; IS QUIT
        JP      Z,MENU_QUIT          ;
        JP      MAIN_LOOP            ; LOOP TO MENU
;*****
; CODE BY HR
;*****
YES_NO_WRITE:
        LD      DE,MSG_YES_NO        ;
        LD      C,09H                ; CP/M WRITE START STRING TO CONSOLE CALL
        CALL    0005H                ;
        LD      C,01H                ; CP/M console input call
        CALL    0005H                ; GET K.B. DATA
        CP      'Y'                  ;
        JP      Z,MENU_WRITE         ;
        CP      'Y'                  ;
        JP      Z,MENU_WRITE         ;
        CP      'Q'                  ; IS QUIT
        JP      Z,MENU_QUIT          ;
        CP      'q'                  ; IS QUIT
        JP      Z,MENU_QUIT          ;
        JP      MAIN_LOOP            ; LOOP TO MENU
;*****
; CODE BY HR
;*****
TT:
        PUSH    AF                  ; Store AF
        LD      A,"T"              ; LOAD A "T"
        CALL    COUT                ; SCREEN IT
        POP     AF                  ; RESTORE AF
        RET                        ; DONE
;*****
; CODE BY HR
;*****

```

SS:

```

PUSH    AF                      ; Store AF
LD      A,"S"                   ; LOAD A "S"
CALL    COUT                    ; SCREEN IT
POP     AF                      ; RESTORE AF
RET     ; DONE

```

```

;*****

```

; CODE BY HR

MOTOR_ON:

```

LD      HL,FLATCH_STORE        ; POINT TO FLATCH
RES     1,(HL)                 ; SET MOTOR ON
LD      A,(FLATCH_STORE)       ; SET A TO SETTINGS
OUT     (FLATCH),A             ; OUTPUT TO CONTROLLER
RET

```

```

;*****

```

; CODE BY HR

MOTOR_OFF:

```

LD      HL,FLATCH_STORE        ; POINT TO FLATCH
SET     1,(HL)                 ; SET MOTOR OFF
LD      A,(FLATCH_STORE)       ; SET A TO SETTINGS
OUT     (FLATCH),A             ; OUTPUT TO CONTROLLER
RET

```

```

;*****

```

```

;_____

```

```

;
; MENU OPERATIONS
;_____

```

MENU_MOTOR_ON:

```

LD      HL,FLATCH_STORE        ; POINT TO FLATCH
RES     1,(HL)                 ; SET MOTOR ON
CALL    OUTFLATCH              ; OUTPUT TO CONTROLLER
JP      MAIN_LOOP              ; LOOP TO MENU

```

MENU_MOTOR_OFF:

```

LD      HL,FLATCH_STORE        ; POINT TO FLATCH
SET     1,(HL)                 ; SET MOTOR OFF
CALL    OUTFLATCH              ; OUTPUT TO CONTROLLER
JP      MAIN_LOOP              ; LOOP TO MENU

```

MENU_SET_TRACK:

```

LD      DE,MSG_ENTER_TRACK_NUMBER
LD      C,09H                  ; CP/M WRITE STRING TO CONSOLE CALL
CALL    0005H                  ;
;
LD      HL,KEYBUF              ; SET POINTER TO KEYBUF AREA
CALL    GETLN                  ; GET A LINE OF INPUT FROM THE USER
LD      HL,KEYBUF              ; SET POINTER TO KEYBUF AREA
CALL    HEXIN                  ; GET TRACK
LD      (TRACK),A              ; STORE TRACK
CALL    SETTRACK               ; DO IT
JP      MAIN_LOOP              ; LOOP TO MENU

```

MENU_FORMAT:

```

LD      DE,MSG_ENTER_TRACK_NUMBER
LD      C,09H                  ; CP/M WRITE STRING TO CONSOLE CALL
CALL    0005H                  ;
;
LD      HL,KEYBUF              ; SET POINTER TO KEYBUF AREA
CALL    GETLN                  ; GET A LINE OF INPUT FROM THE USER
LD      HL,KEYBUF              ; SET POINTER TO KEYBUF AREA
CALL    HEXIN                  ; GET TRACK
LD      (TRACK),A              ; STORE TRACK
CALL    FORMAT                 ; FORMAT A FLOPPY DISK TRACK
JP      MAIN_LOOP              ; LOOP TO MENU

```

MENU_READ:

```

LD      DE,MSG_ENTER_TRACK_NUMBER
LD      C,09H                  ; CP/M WRITE STRING TO CONSOLE CALL
CALL    0005H                  ;
;
LD      HL,KEYBUF              ; SET POINTER TO KEYBUF AREA
CALL    GETLN                  ; GET A LINE OF INPUT FROM THE USER

```

```

        LD      HL,KEYBUF          ; SET POINTER TO KEYBUF AREA
        CALL    HEXIN              ; GET TRACK
        LD      (TRACK),A          ; STORE TRACK
        LD      DE,MSG_ENTER_SECTOR_NUMBER
        LD      C,09H              ; CP/M WRITE STRING TO CONSOLE CALL
        CALL    0005H              ;
        ;
        LD      HL,KEYBUF          ; SET POINTER TO KEYBUF AREA
        CALL    GETLN              ; GET A LINE OF INPUT FROM THE USER
        LD      HL,KEYBUF          ; SET POINTER TO KEYBUF AREA
        CALL    HEXIN              ; GET SECTOR
        LD      (SECTOR),A         ; STORE SECTOR
        ;
        CALL    READ                ; READ A FLOPPY DISK SECTOR
        CP      0FFH               ;
        JP      NZ,DOREADEXIT      ; NO ERROR IF READ RETURNS OTHER THAN -1
        ; ELSE, PRINT ERROR MESSAGE
        ;
;*****

        LD      DE,MSG_ERROR        ;
        LD      C,09H              ; CP/M WRITE ERROR STRING TO CONSOLE CALL
        CALL    0005H              ;
DOREADEXIT:
        JP      MAIN_LOOP          ;

MENU_WRITE:
        LD      DE,MSG_ENTER_TRACK_NUMBER
        LD      C,09H              ; CP/M WRITE STRING TO CONSOLE CALL
        CALL    0005H              ;
        ;
        LD      HL,KEYBUF          ; SET POINTER TO KEYBUF AREA
        CALL    GETLN              ; GET A LINE OF INPUT FROM THE USER
        LD      HL,KEYBUF          ; SET POINTER TO KEYBUF AREA
        CALL    HEXIN              ; GET TRACK
        LD      (TRACK),A          ; STORE TRACK
        LD      DE,MSG_ENTER_SECTOR_NUMBER
        LD      C,09H              ; CP/M WRITE STRING TO CONSOLE CALL
        CALL    0005H              ;
        ;
        LD      HL,KEYBUF          ; SET POINTER TO KEYBUF AREA
        CALL    GETLN              ; GET A LINE OF INPUT FROM THE USER
        LD      HL,KEYBUF          ; SET POINTER TO KEYBUF AREA
        CALL    HEXIN              ; GET SECTOR
        LD      (SECTOR),A         ; STORE SECTOR
        ;
        CALL    WRITE              ; WRITE A FLOPPY DISK SECTOR
        CP      0FFH               ;
        JP      NZ,DOWRITEEXIT     ; NO ERROR IF WRITE RETURNS OTHER THAN -1
        ; ELSE, PRINT ERROR MESSAGE
        ;
;*****

        LD      DE,MSG_ERROR        ;
        LD      C,09H              ; CP/M WRITE ERROR STRING TO CONSOLE CALL
        CALL    0005H              ;
DOWRITEEXIT:
        JP      MAIN_LOOP          ;

MENU_SENSE:
        CALL    CHECKINT           ;
        JP      MAIN_LOOP          ;

MENU_DUMP:
        LD      HL,SECTOR_BUFFER    ; SET ADDRESS TO DUMP
        LD      DE,SECTOR_BUFFER    ; SET END ADDRESS
        INC     D                   ;
        INC     D                   ;
        CALL    DUMP_BUFFER         ; DUMP BUFFER TO CONSOLE
        JP      MAIN_LOOP          ;

MENU_CLEAR:
        LD      HL,SECTOR_BUFFER    ; SET ADDRESS TO DUMP

        LD      C,00H               ;
CLEAR_LOOP:
        LD      A,(D1)              ;

```

```

        LD      (HL),A          ;
        DEC     C               ;
        INC     HL              ;
        JP      NZ,CLEAR_LOOP  ;
        LD      C,00H           ;
CLEAR_LOOP1:
        LD      A,(D1)          ;
        LD      (HL),A          ;
        DEC     C               ;
        INC     HL              ;
        JP      NZ,CLEAR_LOOP1 ;
        JP      MAIN_LOOP       ;

MENU_QUIT:
        LD      DE,MSG_END      ;
        LD      C,09H           ; CP/M WRITE END STRING TO CONSOLE CALL
        CALL    0005H           ;
        LD      C,00H           ; CP/M SYSTEM RESET CALL
        CALL    0005H           ; RETURN TO PROMPT

;
;
; MAIN PROGRAM ENDS HERE
;

;__SETUPDRIVE__
;
;
;      SETUP DRIVE SETTINGS
;

;
;
;
SETUPDRIVE:
        LD      A,RESETL        ; RESET SETTINGS
        OR      MINI            ; SELECT MINI FLOPPY (low dens=1, high dens=0)
        OR      PRECOMP         ; SELECT PRECOMP
        OR      FDDENSITY       ; SELECT DENSITY
        OR      FDREADY         ; SELECT READY SIGNAL
        LD      (FLATCH_STORE),A ; SAVE SETTINGS
        LD      A,01H           ;
        LD      (UNIT),A        ; SET UNIT 1
        LD      A,2             ; DENSITY
        LD      (DENS),A        ;
        LD      A,09            ;
        LD      (EOTSEC),A      ; LAST SECTOR OF TRACK
        LD      A,7FH           ;
        LD      (SRTHUT),A      ; STEP RATE AND HEAD UNLOAD TIME
        LD      A,05H           ;
        LD      (HLT),A         ; HEAD LOAD TIME
        LD      A,0E5H          ;
        LD      (D1),A          ; FILLER BYTE FOR FORMAT
        LD      A,2AH           ;
        LD      (FMTGAP),A      ; GAP FOR FORMAT
        LD      A,0DH           ;
        LD      (GAP),A         ; GAP
        LD      A,80H           ;
        LD      (SECSIZ),A      ; SECTOR SIZE /4
        RET

;__OUTFLATCH__
;
;
;      SEND SETTINGS TO FLOPPY CONTROLLER
;

;
OUTFLATCH:
        LD      A,(FLATCH_STORE) ; SET A TO SETTINGS

```

```

        OUT      (FLATCH),A          ; OUTPUT TO CONTROLLER
        RET

;__READ__
;
;      READ A SECTOR
;
;
READ:
        LD       A,46H              ; BIT 6 SETS MFM, 06H IS READ COMMAND
        LD       (CMD),A
        JP       DSKOP

;__WRITE__
;
;      WRITE A SECTOR
;
;
WRITE:
        LD       A,45H              ; BIT 6 SETS MFM, 05H IS WRITE COMMAND
        LD       (CMD),A
        JP       DSKOP

;__FORMAT__
;
;      FORMAT A TRACK
;
;
FORMAT:
        LD       A,4DH              ; BIT 6 SETS MFM, 0DH IS FORMAT COMMAND
        LD       (CMD),A
        JP       DSKOP

;__DSKOP__
;
;      PERFORM A DISK OPERATION
;
;
DSKOP:
        IF MOTOR_ON_OFF            ; CODE BY HR
;***** TURN OFF MOTOR *****
;*****
        LD       HL,FLATCH_STORE    ; POINT TO FLATCH
        SET     1,(HL)              ; SET MOTOR OFF
        CALL    OUTFLATCH           ; OUTPUT TO CONTROLLER
;*****
        ENDIF                      ; CODE BY HR

        CALL    CHECKINT            ; CHECK INTERRUPT STATUS, MAKE SURE IT IS CLEAR
        CP      0FFH                ; DID IT RETURN WITH ERROR CODE?
        JP      Z,DSKEXIT           ; IF YES, EXIT WITH ERROR CODE
;
        LD      A,(UNIT)            ; GET DISK UNIT NUMBER
        AND     03H                 ; MASK FOR FOUR DRIVES.
        LD      B,A                 ; PARK IT IN B
        LD      A,(HEAD)            ; GET HEAD SELECTION
        AND     01H                 ; INSURE SINGLE BIT
        RLA
        RLA                         ; MOVE HEAD TO BIT 2 POSITION
        OR      B                   ; OR HEAD TO UNIT BYTE IN COMMAND BLOCK
        LD      (UNIT),A            ; STORE IN UNIT
;
        IF MOTOR_ON_OFF            ; CODE BY HR
;***** TURN ON MOTOR *****

```

```

;*****
LD      HL,FLATCH_STORE      ; POINT TO FLATCH
RES     1,(HL)                ; SET MOTOR ON
CALL    OUTFLATCH            ; OUTPUT TO CONTROLLER
;*****
ENDIF                          ; CODE BY HR
;
LD      A,03H                 ; SPECIFY COMMAND
CALL    PFDATA               ; PUSH IT
LD      A,(SRTHUT)           ; STEP RATE AND HEAD UNLOAD TIME
CALL    PFDATA               ; PUSH THAT
LD      A,(HLT)              ;
CALL    PFDATA               ; PUSH THAT
;
CALL    SETTRACK              ; PERFORM SEEK TO TRACK
;
JP      NZ,DSKEXIT            ; IF ERROR, EXIT
;
LD      A,(CMD)               ; WHAT COMMAND IS PENDING?
OR      A                    ; SET FLAGS
CP      4DH                   ; IS IT A FORMAT COMMAND?
JP      Z,FMT                 ; YES, DO FORMAT
JP      NZ,DOS04              ; NO, MUST BE READ OR WRITE COMMAND
DSKEXIT:
IF MOTOR_ON_OFF              ; CODE BY HR
;***** TURN OFF MOTOR *****
;*****
LD      HL,FLATCH_STORE      ; POINT TO FLATCH
SET     1,(HL)                ; SET MOTOR OFF
CALL    OUTFLATCH            ; OUTPUT TO CONTROLLER
;*****
ENDIF                          ; CODE BY HR
OR      0FFH                  ; SET -1 IF ERROR
RET
;
;*****
FMT:                          ; FORMAT TRACK COMMAND
LD      DE,MSG_BEGIN_FORMAT
LD      C,09H                 ; CP/M WRITE START STRING TO CONSOLE CALL
CALL    0005H
;
LD      A,00H
LD      B,A
LD      A,(EOTSEC)
INC     A
LD      C,A
;
LD      A,(CMD)
CALL    PFDATA               ; PUSH FORMAT COMMAND TO I8272
LD      A,(UNIT)
CALL    PFDATA               ; WHICH DRIVE UNIT TO FORMAT
LD      A,(DENS)
CALL    PFDATA               ; WHAT DENSITY
LD      A,(EOTSEC)
CALL    PFDATA               ; ASSUME SC (SECTOR COUNT) EOT
LD      A,(FMTGAP)
CALL    PFDATA               ; WHAT GAP IS NEEDED
LD      A,(D1)
CALL    PFDATA               ; FILLER BYTE FOR SECTORS
;
FMT1:
IN      A,(FMSR)
OR      A                    ; test if byte ready RQM1
JP      P,FMT1
;
AND     20H
JP      Z,DSKOPEND
;
LD      A,(TRACK)
OUT     (FDATA),A            ; SEND CYLINDER NUMBER
;
FMT1A:
IN      A,(FMSR)
OR      A                    ; test if byte ready RQM1
JP      P,FMT1A
;

```

```

        AND        20H                ;
        JP         Z,DSKOPEND          ; jump if in results mode
        ;
        LD         A,(HEAD)            ;
        OUT        (FDATA),A          ; WHICH DRIVE HEAD TO FORMAT
FMT1B:
        IN         A,(FMSR)            ;
        OR         A                  ; test if byte ready RQM1
        JP         P,FMT1B            ;
        ;
        AND        20H                ;
        JP         Z,DSKOPEND          ; jump if in results mode
        ;
        LD         A,B                ;
        OUT        (FDATA),A          ; WHAT SECTOR NUMBER
FMT1C:
        IN         A,(FMSR)            ;
        OR         A                  ; test if byte ready RQM1
        JP         P,FMT1C            ;
        ;
        AND        20H                ;
        JP         Z,DSKOPEND          ; jump if in results mode
        ;
        LD         A,(DENS)            ;
        OUT        (FDATA),A          ; NUMBER OF BYTES PER SECTOR (N2)
        ;
        INC        B                  ; INCREASE SECTOR COUNT
        ;
        LD         A,C                ; IS THIS LAST SECTOR OF TRACK?
        CP         B                  ;
        JP         NZ,FMT1            ; IF NO, SEND ANOTHER SECTOR
        JP         DSKOPEND           ;

;***** RESULT *****
;*****
RESULT:
        LD         DE,MSG_END_OPERATION ;
        LD         C,09H              ; CP/M WRITE START STRING TO CONSOLE CALL
        CALL       0005H              ;
        ;
        LD         C,07H              ; LOAD C WITH NUMBER OF STATUS BYTES
        LD         HL,ST0             ; POINT TO STATS STORAGE
RS3:
        CALL       GFDATA             ; GET FIRST BYTE
        LD         (HL),A             ; SAVE IT
        INC        HL                 ; POINTER++
        DEC        C                  ; CC-1
        JP         NZ,RS3             ; LOOP TIL C0
        ;
RSTEXIT:
        CALL       PRINTRESULTS       ; PRINT RESULTS OF COMMAND
        CALL       CHECKINT           ; CHECK INTERRUPT STATUS, MAKE SURE IT IS CLEAR
        IF MOTOR_ON_OFF              ; CODE BY HR
;***** TURN OFF MOTOR *****
;*****
        LD         HL,FLATCH_STORE    ; POINT TO FLATCH
        SET        1,(HL)             ; SET MOTOR OFF
        CALL       OUTFLATCH          ; OUTPUT TO CONTROLLER
;*****
        ENDIF                        ; CODE BY HR
        ;
        RET                          ; DONE RETURN TO CALLER.

PRINTRESULTS:
        PUSH       AF
        CALL       CRLF
        LD         A,(ST0)            ; GET BYTE
        CALL       HXOUT              ; PRINT IT
        CALL       SPACE              ;
        LD         A,(ST1)            ; GET BYTE
        CALL       HXOUT              ; PRINT IT
        CALL       SPACE              ;
        LD         A,(ST2)            ; GET BYTE
        CALL       HXOUT              ; PRINT IT
        CALL       SPACE              ;
        LD         A,(SCYL)           ; GET BYTE

```



```

        CALL    HXOUT                ; PRINT IT
        CALL    SPACE                ;
        LD      A,(SHEAD)            ; GET BYTE
        CALL    HXOUT                ; PRINT IT
        CALL    SPACE                ;
        LD      A,(SREC)              ; GET BYTE
        CALL    HXOUT                ; PRINT IT
        CALL    SPACE                ;
        LD      A,(SNBIT)             ; GET BYTE
        CALL    HXOUT                ; PRINT IT
        CALL    SPACE                ;
        CALL    SPACE                ;
; CODE BY HR
        CALL    TT                    ; PRINT "T"
        LD      A,(TRACK)             ; TRACK
        CALL    HXOUT                ; PRINT IT
        CALL    SPACE                ;
        CALL    SPACE                ;
;*****
; CODE BY HR
        CALL    SS                    ; PRINT "S"
        LD      A,(SECTOR)            ; SECTOR
        CALL    HXOUT                ; PRINT IT
        CALL    SPACE                ;
        CALL    SPACE                ;
;*****
; CODE BY HR
        CALL    CRLF
        POP     AF
        RET
;*****
;*****
DOS04:
        LD      DE,MSG_BEGIN_READ    ;
        LD      C,09H                ; CP/M WRITE START STRING TO CONSOLE CALL
        CALL    0005H                ;
        ;
        ;
        LD      HL,SECTOR_BUFFER     ; GET BUFFER ADDRESS TO HL
        LD      A,(SECSIZ)           ; XFERLEN
        LD      C,A                  ; C WILL BE THE NUMBER OF TRANSACTIONS
        ; DIVIDED BY 4
        ;
        LD      A,(CMD)               ;
        CALL    PFDATA               ; PUSH COMMAND TO I8272
        LD      A,(UNIT)              ;
        CALL    PFDATA               ;
        LD      A,(TRACK)             ;
        CALL    PFDATA               ;
        LD      A,(HEAD)              ;
        CALL    PFDATA               ;
        LD      A,(SECTOR)            ;
        CALL    PFDATA               ;
        LD      A,(DENS)              ;
        CALL    PFDATA               ; WHAT DENSITY
        LD      A,(EOTSEC)            ;
        CALL    PFDATA               ; ASSUME SC (SECTOR COUNT) EOT
        LD      A,(GAP)               ;
        CALL    PFDATA               ; WHAT GAP IS NEEDED
        LD      A,(DTL)               ; DTL, IS THE LAST COMMAND BYTE TO I8272
        CALL    PFDATA               ;
        LD      A,(CMD)               ; READ IS 0 IS THIS A READ OR WRITE?
        AND     %00000001            ; WRITE IS 1
        JP      NZ,WRR               ; JMP WRITE IF 1
;
;
; PERFORM READ
; LOOP EXECUTES 4X, THIS ALLOWS C RATHER THAN BC AS COUNTER
; SAVING A FEW TSTATES. MAKES UP TO 1024 BYTE SECTORS POSSIBLE.
; FROM READ TO READ MUST NOT EXCEED 25US WORST CASE MIN.
; (76T STATES FOR 3MHZ 8085) or (100 T STATES FOR 4MHZ Z80)
;
RDD_POLL:
FDC_READP0:

```

```

        IN      A, (FMSR)           ;
        OR      A                   ; test if byte ready RQM1
        JP      P, FDC_READP0      ;
                                        ;
        AND     20H                 ;
        JP      Z, DSKOPEND         ; jump if in results mode
                                        ;
        IN      A, (FDATA)          ;
        LD      (HL), A             ;
        INC     HL                  ;

FDC_READP1:
        IN      A, (FMSR)           ;
        OR      A                   ;
        JP      P, FDC_READP1      ;
                                        ;
        AND     20H                 ;
        JP      Z, DSKOPEND         ;
                                        ;
        IN      A, (FDATA)          ;
        LD      (HL), A             ;
        INC     HL                  ;
                                        ;

FDC_READP2:
        IN      A, (FMSR)           ;
        OR      A                   ;
        JP      P, FDC_READP2      ;
                                        ;
        AND     20H                 ;
        JP      Z, DSKOPEND         ;
                                        ;
        IN      A, (FDATA)          ;
        LD      (HL), A             ;
        INC     HL                  ;
                                        ;

FDC_READP3:
        IN      A, (FMSR)           ; 11
        OR      A                   ; 4
        JP      P, FDC_READP3      ; 10
                                        ;
        AND     20H                 ; 7
        JP      Z, DSKOPEND         ; 10
                                        ;
        IN      A, (FDATA)          ; 11
        LD      (HL), A             ; 10
        INC     HL                  ; 11
                                        ;
        DEC     C                   ; 4
        JP      NZ, FDC_READP0      ; 11

DSKOPEND:
;***** SET TC *****
;*****
        LD      HL, FLATCH_STORE    ; POINT TO FLATCH
        SET     0, (HL)             ; SET TC
        CALL    OUTFLATCH           ; OUTPUT TO CONTROLLER
;*****
        NOP                      ;
        NOP                      ; 2 MICROSECOND DELAY

;***** RESET TC *****
;*****
        RES     0, (HL)             ; RESET TC
        CALL    OUTFLATCH           ; OUTPUT TO CONTROLLER
;*****
        NOP                      ;
        NOP                      ; 2 MICROSECOND DELAY
        NOP                      ;
        NOP                      ; 2 MICROSECOND DELAY
        IF     MOTOR_ON_OFF         ; CODE BY HR
;***** TURN OFF MOTOR *****
;*****
        SET     1, (HL)             ; TURN OFF MOTOR
        CALL    OUTFLATCH           ; OUTPUT TO CONTROLLER
;*****
        ENDIF                      ; CODE BY HR

```

```

        JP          RESULT                ; GET STATUS BYTES <RESULT PHASE>

;*****
WRR:
FDC_WRITEP0:
        IN          A,(FMSR)              ;
        OR          A                     ; test if byte ready RQM1
        JP          P,FDC_WRITEP0        ;
        ;
        AND         20H                   ;
        JP          Z,DSKOPEND            ; jump if in results mode
        ;
        LD          A,(HL)                 ;
        OUT         (FDATA),A             ;
        INC         HL                    ;

FDC_WRITEP1:
        IN          A,(FMSR)              ;
        OR          A                     ;
        JP          P,FDC_WRITEP1        ;
        ;
        AND         20H                   ;
        JP          Z,DSKOPEND            ;
        ;
        LD          A,(HL)                 ;
        OUT         (FDATA),A             ;
        INC         HL                    ;

FDC_WRITEP2:
        IN          A,(FMSR)              ;
        OR          A                     ;
        JP          P,FDC_WRITEP2        ;
        ;
        AND         20H                   ;
        JP          Z,DSKOPEND            ;
        ;
        LD          A,(HL)                 ;
        OUT         (FDATA),A             ;
        INC         HL                    ;

FDC_WRITEP3:
        IN          A,(FMSR)              ; 11
        OR          A                     ; 4
        JP          P,FDC_WRITEP3        ; 10
        ;
        AND         20H                   ; 7
        JP          Z,DSKOPEND            ; 10
        ;
        LD          A,(HL)                 ;
        OUT         (FDATA),A             ;
        INC         HL                    ; 11
        ;
        DEC         C                     ; 4
        JP          NZ,FDC_WRITEP0        ; 11
        JP          DSKOPEND              ; 10

;__SETTRACK__
;
;      SEEK TO A TRACK ON GIVEN UNIT
;      A: TRACK #
;
;
SETTRACK:
        ; ANY INTERRUPT PENDING
        ; IF YES FIND OUT WHY/CLEAR
        CALL        CHECKINT              ; CHECK INTERRUPT STATUS, MAKE SURE IT IS CLEAR
        CP          0FFH                   ; DID IT RETURN WITH ERROR CODE?
        JP          Z,SETTRKEXIT          ;
        ;
        LD          A,(TRACK)              ; GET TRACK
        OR          A                     ; SET FLAGS
        JP          Z,RECAL                ; IF 0 PERFORM RECAL INSTEAD OF SEEK
        LD          A,0FH                  ; SEEK COMMAND

```

```

        CALL    PFDATA          ; PUSH COMMAND
        LD      A,(UNIT)        ; SAY WHICH UNIT
        CALL    PFDATA          ; SEND THAT
        LD      A,(TRACK)       ; TO WHAT TRACK
        CALL    PFDATA          ; SEND THAT TOO
        JP      WAIT           ; WAIT FOR INTERRUPT SAYING DONE

;*****
RECAL:
        LD      A,07H           ; RECAL TO TRACK 0
        CALL    PFDATA          ; SEND IT
        LD      A,(UNIT)       ; WHICH UNIT
        CALL    PFDATA          ; SEND THAT TOO

;
WAIT:
;
        CALL    DELAYHSEC       ; DELAY TO LET HEADS SETTLE BEFORE READ
;
; WAIT HERE FOR INTERRUPT SAYING DONE
; LOOP TIL INTERRUPT
        CALL    CHECKINT        ; CHECK INTERRUPT STATUS
;
SETTRKEXIT:
        RET

;__PFDATAS__
;
; WRITE A COMMAND OR PARAMETER SEQUENCE
;
; TRANSFERS ARE SYNCHRONIZED BYT MSR D7 <RQM> AND D6 <DIO>
;
;   RQM  DIO
;   0      0      BUSY
;   1      0      WRITE TO DATA REGISTER PERMITTED
;   1      1      BYTE FOR READ BY HOST PENDING
;   0      1      BUSY
;
;
;
PFDATAS:
        PUSH    AF              ; STORE AF
PFDS1:
        IN      A,(FMSR)        ; READING OR WRITING IS KEYS TO D7 RQM
        AND     80H             ; MASK OFF RQM BIT
        JP      Z,PFDS1        ; WAIT FOR RQM TO BE TRUE.
        IN      A,(FMSR)        ; READ STATUS
        AND     40H             ; WAITING FOR INPUT?
;*****
        CALL    NZ,ERRORT       ; NO, SIGNAL ERROR
        POP     AF              ; RESTORE AF
        OUT     (FDATA),A       ; OUTPUT A TO CONTROLLER
        RET

;__PFDATA__
;
; WRITE A COMMAND OR PARAMETER SEQUENCE
;
; TRANSFERS ARE SYNCHRONIZED BYT MSR D7 <RQM> AND D6 <DIO>
;
;   RQM  DIO
;   0      0      BUSY
;   1      0      WRITE TO DATA REGISTER PERMITTED
;   1      1      BYTE FOR READ BY HOST PENDING
;   0      1      BUSY
;
;
;
PFDATA:
        PUSH    AF              ; STORE AF
PFD1:
        IN      A,(FMSR)        ; READING OR WRITING IS KEYS TO D7 RQM
        AND     80H             ; MASK OFF RQM BIT
        JP      Z,PFD1         ; WAIT FOR RQM TO BE TRUE.
        IN      A,(FMSR)        ; READ STATUS
        AND     40H             ; WAITING FOR INPUT?

```

```

;*****
CALL    NZ,ERRORT      ; NO, SIGNAL ERROR
POP     AF              ; RESTORE AF
OUT     (FDATA),A      ; OUTPUT A TO CONTROLLER
NOP     ; WAIT 24 US BEFORE READING FMSR AGAIN
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
NOP     ;
RET

;_CHECKINT_
;
; CHECK FOR ACTIVE FDC INTERRUPTS BEFORE GIVING I8272 COMMANDS
; POLL RQM FOR WHEN NOT BUSY AND THEN SEND FDC
; SENSE INTERRUPT COMMAND. IF IT RETURNS WITH NON ZERO
; ERROR CODE, PASS BACK TO CALLING ROUTINE FOR HANDLING
;
;
CHECKINT:
PUSH    AF              ; STORE AF
WTDONE:
IN      A,(FMSR)        ; READING OR WRITING IS KEYS TO D7 RQM
AND     80H             ; MASK OFF RQM BIT
JP      Z,WTDONE        ; WAIT FOR RQM TO BE TRUE. WAIT UNTIL DONE
IN      A,(FMSR)        ; READ STATUS
AND     40H             ; WAITING FOR INPUT?
;*****
CALL    NZ,ERRORT      ; NO, SIGNAL ERROR
POP     AF              ; RESTORE AF
CALL    SENDINT         ; SENSE INTERRUPT COMMAND
PUSH    AF              ; STORE AF
;
CP      %00000000       ; NORMAL TERMINATION CASE
LD      DE,MSG_NORMAL_TERM ; SET POINTER TO NORMAL TERM MESSAGE
JP      Z,EXITCI        ; EXIT
;
CP      %01000000       ; ABNORMAL TERMINATION OF COMMAND CASE
LD      DE,MSG_ABNORMAL_TERM_CMD ; SET POINTER TO COMMAND STARTED BUT NOT SUCCESSFULLY COMPL
ETED
JP      Z,EXITCI        ; EXIT
;
CP      %11000000       ; ABNORMAL TERMINATION OF COMMAND CASE
LD      DE,MSG_READY_CHANGED ; SET POINTER TO READY SIGNAL FROM FDD CHANGED STATE
JP      Z,EXITCI        ; EXIT
;
CP      %10000000       ; INVALID COMMAND
LD      DE,MSG_INVALID_COMMAND ; SET POINTER TO INVALID COMMAND MESSAGE
;
EXITCI:
LD      C,09H           ; CP/M WRITE END STRING TO CONSOLE CALL
CALL    0005H           ;
POP     AF              ; RESTORE AF
CALL    PRINTRESULTS    ; PRINT RESULTS OF COMMAND

```

```

/
RET
;

; _DELAYHSEC_
/
;
; DELAY FOR 1/2 SECOND
/
/
/
DELAYHSEC:
LD HL,00000H ; 65536
DELDM:
NOP ; (4 T)
NOP ; (4 T)
NOP ; (4 T)
NOP ; (4 T)
DEC L ; (6 T)
JP NZ,DELDM ; (10 T) 24 T 8 MICROSECONDS AT 4 MHZ
DEC H ; (6 T)
JP NZ,DELDM ; (10 T) (8 US * 256) * 256 524288 US .5 SECONDS
RET

; _ERRORT_
/
;
; ERROR HANDLING
/
/
/
ERRORT:
POP AF
ERRORT1:
; CODE BY HR
LD DE,MSG_ERRORX ; POINT TO ERROR STRING
; *****
LD C,09H ; CP/M WRITE ERROR STRING TO CONSOLE CALL
CALL 0005H
;
ERRCLR:
IN A,(FDATA) ; CLEAR THE JUNK OUT OF DATA REGISTER
IN A,(FMSR) ; CHECK WITH RQM
AND 80H ; IF STILL NOT READY, READ OUT MORE JUNK

JP Z,ERRCLR ;

LD A,0FFH ; RETURN ERROR CODE -1
;
RET

; _SENDINT_
/
;
; SENSE INTERRUPT COMMAND
/
/
/
SENDINT:
LD A,08H ; SENSE INTERRUPT COMMAND
CALL PFDATA ; SEND IT
CALL GFDATA ; GET RESULTS
LD (ST0),A ; STORE THAT
AND 0C0H ; MASK OFF INTERRUPT STATUS BITS
CP 80H ; CHECK IF INVALID COMMAND
JP Z,ENDSENDINT ; YES, EXIT
CALL GFDATA ; GET ANOTHER (STATUS CODE 1)
LD (ST1),A ; SAVE THAT
LD A,(ST0) ; GET FIRST ONE
AND 0C0H ; MASK OFF ALL BUT INTERRUPT CODE 00 IS NORMAL
ENDSENDINT:
RET ; ANYTHING ELSE IS AN ERROR

; _GFDATA_
/
;
; GET DATA FROM FLOPPY CONTROLLER

```

```

; TRANSFERS ARE SYNCHONIZED BYT MSR D7 <RQM> AND D6 <DIO>
;
;      RQM   DIO
;      0           0           BUSY
;      1           0           WRITE TO DATA REGISTER PERMITTED
;      1           1           BYTE FOR READ BY HOST PENDING
;      0           1           BUSY
;
;_____
;
;***** GFDATA *****
;*****
GFDATA:
      IN      A, (FMSR)      ; READ STATUS BYTE
      AND     80H           ; MASK OFF RQM
      JP      Z, GFDATA     ; LOOP WHILE BUSY
      IN      A, (FMSR)     ; READ STTUS BUTE
      AND     40H           ; MASK OFF DIO
      CALL    Z, ERRORRT1   ; IF WRITE EXPECTED RUN ERRORRT
      IN      A, (FDATA)    ; READ DATA
      NOP     ; WAIT 24 US BEFORE READING FMSR AGAIN
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      RET

;__DUMP_____
;
;      Print a Memory Dump (LOW)
;_____
;
DUMP_BUFFER:
      CALL    CRLF          ;
BLKRD:
      CALL    PHL           ; PRINT START LOCATION
      LD      C, 16         ; SET FOR 16 LOCS
      PUSH    HL            ; SAVE STARTING HL
NXTONE:
      LD      A, (HL)       ; GET BYTE
      CALL    HXOUT         ; PRINT IT
      CALL    SPACE        ;
UPDH:
      INC     HL            ; POINT NEXT
      DEC     C             ; DEC. LOC COUNT
      JR      NZ, NXTONE    ; IF LINE NOT DONE
                          ; NOW PRINT 'DECODED' DATA TO RIGHT OF DUMP
PCRLF:
      CALL    SPACE        ; SPACE IT
      LD      C, 16         ; SET FOR 16 CHARS
      POP     HL            ; GET BACK START
PCRLF0:
      LD      A, (HL)       ; GET BYTE
      AND     060H          ; SEE IF A 'DOT'

```

```

        LD      A,(HL)          ; O.K. TO GET
        JR      NZ,PDOT         ;
DOT:
        LD      A,2EH           ; LOAD A DOT
PDOT:
        CALL    COUT            ; PRINT IT
        INC     HL              ;
        LD      A,D             ;
        CP      H               ;
        JR      NZ,UPDH1        ;
        LD      A,E             ;
        CP      L               ;
        JP      Z,DUMP_END      ;
;
;IF BLOCK NOT DUMPED, DO NEXT CHARACTER OR LINE
UPDH1:
        DEC     C               ; DEC. CHAR COUNT
        JR      NZ,PCRLF0       ; DO NEXT
CONTD:
        CALL    CRLF            ;
        JP      BLKRD           ;
DUMP_END:
        RET                    ;

;_HXOUT_
;
;
;      PRINT THE ACCUMULATOR CONTENTS AS HEX DATA ON THE SERIAL PORT
;
;
HXOUT:
        PUSH    BC              ; SAVE BC
        LD      B,A             ;
        RLC     A               ; DO HIGH NIBBLE FIRST
        RLC     A               ;
        RLC     A               ;
        RLC     A               ;
        AND     0FH             ; ONLY THIS NOW
        ADD     A,30H           ; TRY A NUMBER
        CP      3AH             ; TEST IT
        JR      C,OUT1          ; IF CY SET PRINT 'NUMBER'
        ADD     A,07H           ; MAKE IT AN ALPHA
OUT1:
        CALL    COUT            ; SCREEN IT
        LD      A,B             ; NEXT NIBBLE
        AND     0FH             ; JUST THIS
        ADD     A,30H           ; TRY A NUMBER
        CP      3AH             ; TEST IT
        JR      C,OUT2          ; PRINT 'NUMBER'
        ADD     A,07H           ; MAKE IT ALPHA
OUT2:
        CALL    COUT            ; SCREEN IT
        POP     BC              ; RESTORE BC
        RET                    ;

;_SPACE_
;
;
;      PRINT A SPACE CHARACTER ON THE SERIAL PORT
;
;
SPACE:
        PUSH    AF              ; Store AF
        LD      A,20H           ; LOAD A "SPACE"
        CALL    COUT            ; SCREEN IT
        POP     AF              ; RESTORE AF
        RET                    ; DONE

;_CRLF_
;
;
;      PRINT A cr/lf

```



```

;
;
CRLF:
    PUSH    AF                ; Store AF
    LD      A,0DH             ; LOAD A 0DH
    CALL    COUT              ; SCREEN IT
    LD      A,0AH             ; LOAD A 0AH
    CALL    COUT              ; SCREEN IT
    POP     AF                ; RESTORE AF
    RET                     ; DONE

;__COUT__
;
;    PRINT CONTENTS OF A
;
;
COUT:
    PUSH    BC                ;
    PUSH    AF                ;
    PUSH    HL                ;
    PUSH    DE                ;

    LD      (COUT_BUFFER),A ;
    LD      DE,COUT_BUFFER ;
    LD      C,09H             ; CP/M WRITE START STRING TO CONSOLE CALL
    CALL    0005H
    POP     DE                ;
    POP     HL                ;
    POP     AF                ;
    POP     BC                ;
    RET                     ; DONE

;__PHL__
;
;    PRINT THE HL REG ON THE SERIAL PORT
;
;
PHL:
    LD      A,H               ; GET HI BYTE
    CALL    HXOUT             ; DO HEX OUT ROUTINE
    LD      A,L               ; GET LOW BYTE
    CALL    HXOUT             ; HEX IT
    CALL    SPACE             ;
    RET                     ; DONE

;__GETLN__
;
;    Read a line(80) of text from the Serial Port, handle <BS>, term on <CR>.
;    Exit if too many chars.  Store result in HL.  Char count in C.
;
;
GETLN:
    LD      C,00H             ; ZERO CHAR COUNTER
    PUSH    DE                ; STORE DE
GETLNLOP:
    CALL    KIN               ; GET A KEY
    CP      CR                ; IS <CR>?
    JR      Z,GETLNDONE       ; YES, EXIT
    CP      BS                ; IS <BS>?
    JR      NZ,GETLNSTORE     ; NO, STORE CHAR
    LD      A,C               ; A=C
    CP      0                 ;
    JR      Z,GETLNLOP        ; NOTHING TO BACKSPACE, IGNORE & GET NEXT KEY
    DEC     HL                ; PERFORM BACKSPACE
    DEC     C                 ; LOWER CHAR COUNTER
    LD      A,0               ;

```

```

        LD      (HL),A      ; STORE NULL IN BUFFER
        LD      A,20H      ; BLANK OUT CHAR ON TERM
        CALL    COUT        ;
        LD      A,BS       ;
        CALL    COUT        ;
        JR      GETLNLOP    ; GET NEXT KEY
GETLNSTORE:
        LD      (HL),A      ; STORE CHAR IN BUFFER
        INC     HL          ; INC POINTER
        INC     C           ; INC CHAR COUNTER
        LD      A,C         ; A=C
        CP      4DH         ; OUT OF BUFFER SPACE?
        JR      NZ,GETLNLOP ; NOPE, GET NEXT CHAR
GETLNDONE:
        LD      (HL),00H    ; STORE NULL IN BUFFER
        POP     DE          ; RESTORE DE
        RET

```

```

;__KIN__

```

```

;
;      Read from the Serial Port and Echo & convert input to UCASE
;

```

```

;
KIN:
        PUSH    BC          ;
        PUSH    HL          ;
        PUSH    DE          ;
        LD      C,01H       ; CP/M console input call
        CALL    0005H       ; GET K.B. DATA
        POP     DE          ;
        POP     HL          ;
        POP     BC          ;
        RET

```

```

;__LDHL__

```

```

;
;      GET ONE WORD OF HEX DATA FROM BUFFER POINTED TO BY HL SERIAL PORT, RETURN IN HL
;

```

```

;
LDHL:
        PUSH    DE          ; STORE DE
        CALL    HEXIN       ; GET K.B. AND MAKE HEX
        LD      D,A         ; THATS THE HI BYTE
        CALL    HEXIN       ; DO HEX AGAIN
        LD      L,A         ; THATS THE LOW BYTE
        LD      H,D         ; MOVE TO HL
        POP     DE          ; RESTORE BC
        RET                ; GO BACK WITH ADDRESS

```

```

;__HEXIN__

```

```

;
;      GET ONE BYTE OF HEX DATA FROM BUFFER IN HL, RETURN IN A
;

```

```

;
HEXIN:
        PUSH    BC          ; SAVE BC REGS.
        CALL    NIBL        ; DO A NIBBLE
        RLC     A           ; MOVE FIRST BYTE UPPER NIBBLE
        RLC     A           ;
        RLC     A           ;
        RLC     A           ;
        LD      B,A         ; SAVE ROTATED BYTE
        CALL    NIBL        ; DO NEXT NIBBLE
        ADD     A,B         ; COMBINE NIBBLES IN ACC.
        POP     BC          ; RESTORE BC
        RET                ; DONE
NIBL:
        LD      A,(HL)      ; GET K.B. DATA
        INC     HL          ; INC KB POINTER

```

```

        CP      40H      ; TEST FOR ALPHA
        JR      NC,ALPH  ;
        AND     0FH      ; GET THE BITS
        RET
ALPH:
        AND     0FH      ; GET THE BITS
        ADD     A,09H     ; MAKE IT HEX A-F
        RET

;_FILL_MEM_
;
;      Function : fill memory with a value
;      Input    : HL = start address block
;                : BC = length of block
;                : A = value to fill with
;      Uses     : DE, BC
;
FILL_MEM:
;      HL = start address of block
;      DE = HL + 1
        ld      e,l      ;
        ld      d,h      ;
        inc     de       ;
;      initialise first byte of block
;      with data byte (&00)
;      with data byte in A
        ld      (hl),A   ;
        ldir     ;
        RET          ; return to caller

;-----
; TEXT CONSTANTS

MSG_START:
        DEFB    LF,CR    ; LINE FEED AND CARRIAGE RETURN
        DEFM    "FLOPPY TEST PROGRAM"
        DEFB    LF, CR   ; LINE FEED AND CARRIAGE RETURN

        DEFM    "1->START MOTOR"
        DEFB    LF, CR   ; LINE FEED AND CARRIAGE RETURN
        DEFM    "2->STOP MOTOR"
        DEFB    LF, CR   ; LINE FEED AND CARRIAGE RETURN
        DEFM    "3->SELECT TRACK"
        DEFB    LF, CR   ; LINE FEED AND CARRIAGE RETURN
        DEFM    "4->FORMAT TRACK"
        DEFB    LF, CR   ; LINE FEED AND CARRIAGE RETURN
        DEFM    "5->READ SECTOR"
        DEFB    LF, CR   ; LINE FEED AND CARRIAGE RETURN
        DEFM    "6->WRITE SECTOR"
        DEFB    LF, CR   ; LINE FEED AND CARRIAGE RETURN
        DEFM    "7->DUMP BUFFER"
        DEFB    LF, CR   ; LINE FEED AND CARRIAGE RETURN
        DEFM    "8->SENSE INT"
        DEFB    LF, CR   ; LINE FEED AND CARRIAGE RETURN
        DEFM    "9->CLEAR BUFFER"
        DEFB    LF, CR   ; LINE FEED AND CARRIAGE RETURN
        DEFM    "Q->Quit"
        DEFB    LF, CR   ; LINE FEED AND CARRIAGE RETURN
        DEFB    ENDS     ; LINE TERMINATOR

MSG_NORMAL_TERM:
        DEFB    LF,CR    ; LINE FEED AND CARRIAGE RETURN
        DEFM    "INTERRUPT NORMAL TERMINATION"
        DEFB    LF,CR    ; LINE FEED AND CARRIAGE RETURN
        DEFB    ENDS     ; LINE TERMINATOR

;
MSG_ABNORMAL_TERM_CMD:
        DEFB    LF,CR    ; LINE FEED AND CARRIAGE RETURN
        DEFM    "ABNORMAL TERMINATION OF COMMAND: STARTED BUT RETURNED ERROR"
        DEFB    LF,CR    ; LINE FEED AND CARRIAGE RETURN
        DEFB    ENDS     ; LINE TERMINATOR

```

```

;
MSG_INVALID_COMMAND:
    DEFB    LF,CR                ; LINE FEED AND CARRIAGE RETURN
    DEFM    "INVALID COMMAND"
    DEFB    LF,CR                ; LINE FEED AND CARRIAGE RETURN
    DEFB    ENDS                ; LINE TERMINATOR
;
MSG_READY_CHANGED:
    DEFB    LF,CR                ; LINE FEED AND CARRIAGE RETURN
    DEFM    "ABNORMAL TERMINATION OF COMMAND: FDD READY STATE CHANGED"
    DEFB    LF,CR                ; LINE FEED AND CARRIAGE RETURN
    DEFB    ENDS                ; LINE TERMINATOR
;
MSG_ERROR:
    DEFB    LF,CR                ; LINE FEED AND CARRIAGE RETURN
    DEFM    "TERMINATION ERROR"
    DEFB    LF,CR                ; LINE FEED AND CARRIAGE RETURN
    DEFB    ENDS                ; LINE TERMINATOR
; CODE BY HR
MSG_ERRORX:
    DEFB    LF,CR                ; LINE FEED AND CARRIAGE RETURN
    DEFM    "TERMINATION ERRORX"
    DEFB    LF,CR                ; LINE FEED AND CARRIAGE RETURN
    DEFB    ENDS                ; LINE TERMINATOR
;*****

MSG_ENTER_TRACK_NUMBER:
    DEFB    LF, CR                ; LINE FEED AND CARRIAGE RETURN
    DEFM    "ENTER TRACK NUMBER (00):"
    DEFB    ENDS                ; LINE TERMINATOR
MSG_ENTER_SECTOR_NUMBER:
    DEFB    LF, CR                ; LINE FEED AND CARRIAGE RETURN
    DEFM    "ENTER SECTOR NUMBER (00):"
    DEFB    ENDS                ; LINE TERMINATOR
MSG_BEGIN_FORMAT:
    DEFB    LF, CR                ; LINE FEED AND CARRIAGE RETURN
    DEFM    "FORMAT BEGINNING. . ."
    DEFB    ENDS                ; LINE TERMINATOR
MSG_BEGIN_READ:
    DEFB    LF, CR                ; LINE FEED AND CARRIAGE RETURN
    DEFM    "READ/WRITE BEGINNING. . ."
    DEFB    ENDS                ; LINE TERMINATOR
MSG_END_OPERATION:
    DEFB    LF, CR                ; LINE FEED AND CARRIAGE RETURN
    DEFM    "OPERATION ENDED. . ."
    DEFB    ENDS                ; LINE TERMINATOR

MSG_END:
    DEFB    LF, CR                ; LINE FEED AND CARRIAGE RETURN
    DEFM    "END FLOPPY TEST PROGRAM"
    DEFB    LF, CR                ; LINE FEED AND CARRIAGE RETURN
    DEFB    ENDS                ; LINE TERMINATOR
; CODE BY HR
MSG_YES_NO:
    DEFB    LF, CR                ; LINE FEED AND CARRIAGE RETURN
    DEFM    "YES or NO"
    DEFB    LF, CR                ; LINE FEED AND CARRIAGE RETURN
    DEFB    ENDS                ; LINE TERMINATOR
;*****

COUT_BUFFER:
    DEFB    00
    DEFB    "$"
FLATCH_STORE:
    DEFB    00
;
; DISK COMMAND BLOCK
;
CMD:    DEFB    0                ; COMMAND READ OR WRITE,
UNIT:    DEFB    0                ; PHYSICAL DRIVE 0->3
TRACK:    DEFB    0                ; PHYSICAL TRACK 0->256
HEAD:    DEFB    0                ; HEAD SEL 0 OR 1
SECTOR:    DEFB    1                ; PHYSICAL SECTOR, ALWAYS 1 TO MAX SECTOR
DENS:    DEFB    2                ; DENSITY

```

```

EOTSEC:    DEFB    09          ; LAST SECTOR OF TRACK
GAP:       DEFB    1BH        ; VALUE FOR IRG <GAP3>
SECSIZ:    DEFB    080H       ; HOW MANY BYTES TO TRANSFER/4
DTL:       DEFB    0FFH       ; SIZE OF SECTOR
;
SRTHUT:    DEFB    7FH        ; STEP RATE AND HEAD UNLOAD TIME
HLT:       DEFB    05H        ; HEAD LOAD TIME
;
MIN:       DEFB    MINI       ; LATCH BIT PATTERN FOR FDC9229 MINITRUE
; CODE BY HR
; D1 instead of D
; D1 instead of D because the Assembler does not allow D
D1:        DEFB    0E5H       ; FILLER BYTE FOR FORMAT
;*****
FMTGAP:    DEFB    054H       ; GAP FOR FORMAT
PRE:       DEFB    PRECOMP    ; LATCH BIT PATTERN FOR FDC9229 PRECOMP125NS
;
; STATUS RESULT STORAGE
;
ST0:       DEFB    0          ; STORE STATUS 0
ST1:       DEFB    0          ; ST1
ST2:       DEFB    0          ; ST2
SCYL:      DEFB    0          ; TRACK
SHEAD:     DEFB    0          ; HEAD 0 OR 1
SREC:      DEFB    0          ; SECTOR
SNBIT:     DEFB    0          ; DENSITY
; CODE BY HR
KEYBUF:    DEFS     80,20h     ; 80 x SPACE
;*****
; ;KEYBUF:    DEFM "
KEYEND:
TKEYEND:   EQU    KEYEND-KEYBUF
;*****
ORG        2500h
SECTOR_BUFFER:    DEFS     0200h

END

```