

```

;
; ROM Program for S100Computers IDE intreface board (Based on MYIDE.ASM)
;=====
;
;      V1.0   3/13/2011      ;Initial version
;      V1.1   3/15/2011      ;Corrected Init Drive routine and boot CPM sector number

;Ports for 8255 chip. Change these to specify where your 8255 is addressed,
;The first three control which 8255 ports have the control signals,
;upper and lower data bytes. The last one (IDEportCtrl), is for mode setting for the
;8255 to configure its actual I/O ports (A,B & C).
;
;Note most drives these days dont use the old Head,Track, Sector terminology. Instead
;we use "Logical Block Addressing" or LBA. This is what we use below. LBA treats the drive
;as one continous set of sectors, 0,1,2,3,... 3124,...etc. However as seen below we need to
;convert this LBA to heads,tracks and sectors to be compatible with CPM & MSDOS.

      ; INCLUDE Z-80 MACRO LIBRARY:
      MACLIB Z80                      ;For the Z80 DJNZ opcode

IDEportA      EQU    030H              ;lower 8 bits of IDE interface
IDEportB      EQU    031H              ;upper 8 bits of IDE interface
IDEportC      EQU    032H              ;control lines for IDE interface
IDEportCtrl   EQU    033H              ;8255 configuration port

READcfg8255   EQU    10010010b        ;Set 8255 IDEportC to output, IDEportA/B input
WRITEcfg8255  EQU    10000000b        ;Set all three 8255 ports to output mode

;IDE control lines for use with IDEportC.

IDEa0line     EQU    01H              ;direct from 8255 to IDE interface
IDEa1line     EQU    02H              ;direct from 8255 to IDE interface
IDEa2line     EQU    04H              ;direct from 8255 to IDE interface
IDEcs0line    EQU    08H              ;inverter between 8255 and IDE interface
IDEcs1line    EQU    10H              ;inverter between 8255 and IDE interface
IDEwrline     EQU    20H              ;inverter between 8255 and IDE interface
IDERdline     EQU    40H              ;inverter between 8255 and IDE interface
IDERstline    EQU    80H              ;inverter between 8255 and IDE interface
;
;Symbolic constants for the IDE Drive registers, which makes the
;code more readable than always specifying the address bits

REGdata       EQU    IDEcs0line
REGerr        EQU    IDEcs0line + IDEa0line
REGsecCnt     EQU    IDEcs0line + IDEa1line
REGsector     EQU    IDEcs0line + IDEa1line + IDEa0line
REGcylinderLSB EQU    IDEcs0line + IDEa2line
REGcylinderMSB EQU    IDEcs0line + IDEa2line + IDEa0line
REGshd        EQU    IDEcs0line + IDEa2line + IDEa1line          ; (0EH)
REGcommand    EQU    IDEcs0line + IDEa2line + IDEa1line + IDEa0line ; (0FH)
REGstatus     EQU    IDEcs0line + IDEa2line + IDEa1line + IDEa0line
REGcontrol    EQU    IDEcs1line + IDEa2line + IDEa1line
REGastatus    EQU    IDEcs1line + IDEa2line + IDEa1line + IDEa0line

;IDE Command Constants. These should never change.

COMMANDrecal  EQU    10H
COMMANDread   EQU    20H
COMMANDwrite  EQU    30H
COMMANDinit   EQU    91H
COMMANDid     EQU    0ECH
COMMANDspindown EQU    0E0H
COMMANDspinup EQU    0E1H
;
;
; IDE Status Register:
; bit 7: Busy 1=busy, 0=not busy
; bit 6: Ready 1=ready for command, 0=not ready yet
; bit 5: DF 1=fault occured insIDE drive
; bit 4: DSC 1=seek complete
; bit 3: DRQ 1=data request ready, 0=not ready to xfer yet
; bit 2: CORR 1=correctable error occured
; bit 1: IDX vendor specific
; bit 0: ERR 1=error occured
;
;
;Equates for display on SD Systems Video Board (Used In CPM Debugging mode only)

```

```

SCROLL      EQU    01H    ;Set scrool direction UP.
LF          EQU    0AH
CR          EQU    0DH
BS          EQU    08H    ;Back space (required for sector display)
BELL       EQU    07H
SPACE      EQU    20H
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
TAB        EQU    09H    ;TAB ACROSS (8 SPACES FOR SD-BOARD)
ESC        EQU    1BH
CLEAR      EQU    1CH    ;SD Systems Video Board, Clear to EOL. (Use 80 spaces if EOL not available
;on other video cards)

;Critical variables stored in low RAM (unused area by CPM).
@DMA       EQU    40H    ;Store DMA address at 40 & 41H in RAM
@DRIVE$SEC EQU    43H
@DRIVE$TRK EQU    44H
@SEC       EQU    46H
@TRK       EQU    48H
@SEC1      EQU    4AH    ;For disk partition copy
@TRK1      EQU    4CH
@SEC2      EQU    4EH
@TRK2      EQU    50H
@StartLineHex EQU 52H
@StartLineASCII EQU 54H
@ByteCount EQU    56H
@SecCount  EQU    58H
@DisplayFlag EQU 5AH    ;Display of sector data initially ON
@DELAY$Store EQU 5BH

BUFFER     EQU    100H   ;<----- Will write/read sector data to/from here
IDBuffer   EQU    100H   ;<----- Will write drive ID data here also can be change if required.
ABORT      EQU    0F000H ;Abort location.

SEC$SIZE EQU 512        ;Assume sector size as 512. (Not tested for other sizes)
MAXSEC EQU 3DH         ;Sectors per track for CF my Memory drive, Kingston CF 8G. (For CPM format, 0-3CH)
;This translates to LBA format of 1 to 3D sectors, for a total of 61 sectors/track.
;This CF card actully has 3F sectors/track. Will use 3D for my CPM3 system because
;my Seagate drive has 3D sectors/track. Don't want different CPM3.SYS files around
;so this program as is will also work with a Seagate 6531 IDE drive
MAXTRK EQU 0FFH       ;CPM3 allows up to 8MG so 0-256 "tracks"

CPM$ADDRESS EQU 100H   ;For BOOTCPM Command, will place the CPMLDR.COM Loader here with
;CPMLDR.COM will ALWAYS be on TRK 0,SEC2, (LBA Mode)
SEC$COUNT EQU 12     ;CPMLDR.COM requires (currently) 10, 512 byte sectors
;Add extra just in case

Z80PORT EQU 0D0H     ;4 PORTS ON Z80 CPU BOARD FOR MEMORY MANAGEMENT & INT Controller
;Can be ignored in other systems

ORG 0F000H          ;<--- EPROM Addresss set ot F000H

BEGIN:
LXI SP,AHEAD-4     ;SETUP A FAKE STACK
JMP MEMSIZE        ;RETURNS WITH TOP OF RAM IN [HL]
DW AHEAD           ;Ret will pick up this address
AHEAD: SPHL        ;[HL] CONTAINS TOP OF RAM - WORKAREA
PUSH H
POPIX              ;Store stack pointer for below in [IX]

LXI D,SIGN$ON      ;print a welcome message
CALL PSTRING
JMP OVER$TBL

;----- CONSOL I/O (Only HARDWARE DEPENDENT CODE) -----

ZCSTS: IN 0H        ;Get Consol Status
ANI 02H
RZ                ;Return Z if nothing there

```

```

MVI    A,01H
ORA    A
RET

```

```

ZCO:   PUSH    PSW                ;Write character that is in [C]
ZCO1:  IN      0H                  ;Show Character
      ANI     04H
      JZ      ZCO1
      MOV     A,C
      OUT    1H
      POP    PSW
      RET

```

```

ZCI:   IN      0H                  ;Get Character in [A]
      ANI     02H
      JZ      ZCI
      IN     01H
      ANI     7FH                  ;Strip parity
      RET

```

```

;-----
;COMMAND BRANCH TABLE
TBL:   DW  ERROR    ; "A"
      DW  BACKUP   ; "B" Backup partition
      DW  CPMBOOT  ; "C" LOAD CPM (If present)
      DW  DISPLAY  ; "D" Sector contents display:- ON/OFF
      DW  ERROR    ; "E"
      DW  FORMAT   ; "F" Format current disk
      DW  RESTORE  ; "G" Restore backup
      DW  ERROR    ; "H"
      DW  NEXT$SECT ; "I" Next Sector
      DW  PREV$SEC  ; "J" Previous sector
      DW  ERROR    ; "K"
      DW  SET$LBA  ; "L" Set LBA value (Set Track,sector)
      DW  ERROR    ; "M"
      DW  POWER$DOWN; "N" Power down hard disk command
      DW  ERROR    ; "O"
      DW  ERROR    ; "P"
      DW  ERROR    ; "Q"
      DW  READ$SEC ; "R" Read sector to data buffer
      DW  SEQ$RD   ; "S" Sequential sec read and display contents
      DW  ERROR    ; "T"
      DW  POWER$UP ; "U" Power up hard disk command
      DW  N$RD$SEC ; "V" Read N sectors
      DW  WRITE$SEC ; "W" Write data buffer to current sector
      DW  N$WR$SEC ; "X" Write N sectors
      DW  ERROR    ; "Y"
      DW  ERROR    ; "Z"

```

```

OVER$TBL:
      LXI    D,SP$MSG                ;Print Current Stack Location
      CALL   PSTRING

      PUSHIX                ;SP is stored from above in [IX]
      POP    H
      PUSH  H
      MOV   A,H
      CALL  PHEX              ;Print HL/SP
      POP  H
      CALL  PHEX
      CALL  ZCRLF              ;Then CRLF
      CALL  ZCSTS              ;CHECK IF GARBAGE AT KEYBOARD
      CNZ  ZCI                  ;If so flush it

      MVI   A,04H              ;SETUP MEMORY MANAGEMENT TO OVERLAP WITH
      OUT   Z80PORT+3          ;CURRENT RAM in 64K Space
      XRA   A
      OUT   Z80PORT+2
      OUT   Z80PORT+1          ;ALSO KILL THE INTERSYSTEMS Z80 CPU BOARD INT CONTROLLER (If present)

      DCR   A
      STA   @DisplayFlag      ;Set detail display flag on initially

      CALL  IDEinit            ;initialize the board and drive. If there is no drive abort
      JZ    INIT$OK            ;Setup for main menu commands

      LXI   D,INIT$ERROR

```

```

CALL    PSTRING
CALL    SHOWerrors
JMP     ABORT

INIT$OK:
CALL    driveid      ;Get the drive ID info. If there is no drive, abort
JZ      INIT$OK1

LXI     D, ID$ERROR
CALL    PSTRING
CALL    SHOWerrors
JMP     ABORT

INIT$OK1:            ;print the drive's model number
LXI     D, msgmdl
CALL    PSTRING
LXI     H, IDbuffer + 54
MVI     B, 10        ;character count in words
CALL    printname    ;Print [HL], [B] X 2 characters
CALL    ZCRLF

LXI     D, msgsn
CALL    PSTRING
LXI     H, IDbuffer + 20
MVI     B, 5         ;Character count in words
CALL    printname
CALL    ZCRLF
                                ;Print the drive's firmware revision string

LXI     D, msgrev
CALL    PSTRING
LXI     H, IDbuffer + 46
MVI     B, 2
CALL    printname    ;Character count in words
CALL    ZCRLF
                                ;print the drive's cylinder, head, and sector specs

LXI     D, msgcy
CALL    PSTRING
LXI     H, IDbuffer + 2
CALL    printparm
LXI     D, msghd
CALL    PSTRING
LXI     H, IDbuffer + 6
CALL    printparm
LXI     D, msgsc
CALL    PSTRING
LXI     H, IDbuffer + 12
CALL    printparm
CALL    ZCRLF
                                ;Default position will be first block

LXI     H, 0
SHLD   @SEC          ;Default to Track 0, Sec 0
SHLD   @TRK
LXI     H, buffer
SHLD   @DMA

CALL    IDEinit      ;For some reason this need to be here after getting the drive ID.
                                ;otherwise sector #'s are off by one!

mainloop:           ;A 1 line prompt
LXI     D, CMD$STRING1 ;List command options (Turn display option to on)
CALL    PSTRING

CALL    wrlba        ;Update LBA on drive
CALL    DISPLAYposition ;Display current Track, sector, head#

LXI     D, Prompt
CALL    PSTRING

CALL    GETCMD       ;Simple character Input (Note, no fancy checking)
CPI     ESC          ;Abort if ESC
JZ      ABORT
CALL    upper
CALL    ZCRLF

SBI     '@'          ;Adjust to 0, 1AH

ADD     A
LXI     H, TBL
ADD     L
                                ;X2
                                ;Get menu selection

```

```

MOV     L,A
MOV     A,M
INX     HL
MOV     H,M
MOV     L,A           ;Jump to table pointer
PCHL                    ;JMP (HL)

READ$SEC:                ;Read Sector @ LBA to the RAM buffer
LXI     H,buffer       ;Point to buffer
SHLD   @DMA

CALL    READSECTOR

JZ      main1b         ;Z means the sector read was OK
CALL    ZCRLF
JMP     mainloop
main1b: LXI     D, msgrd ;Sector read OK
CALL    PSTRING

LDA     @DisplayFlag   ;Do we have detail sector data display flag on or off
ORA     A              ;NZ = on
JZ      mainloop
LXI     H,buffer       ;Point to buffer. Show sector data flag is on
SHLD   @DMA
CALL    HEXDUMP        ;Show sector data
JMP     mainloop

WRITE$SEC:                ;Write data in RAM buffer to sector @ LBA
LXI     D,msgsure      ;Are you sure?
CALL    PSTRING
CALL    ZCI
CALL    upper
CPI     'Y'
JNZ     main2c
CALL    ZCRLF

LXI     H,buffer       ;Point to buffer
SHLD   @DMA

CALL    WRITESECTOR

JZ      main2b         ;Z means the sector write was OK
CALL    ZCRLF
JMP     mainloop
main2b: LXI     D, msgwr ;Sector written OK
CALL    PSTRING
main2c: JMP     mainloop

SET$LBA:                  ;Set the logical block address
LXI     D,GET$LBA
CALL    PSTRING
CALL    ghex32lba      ;Get new CPM style Track & Sector number and put them in RAM at @SEC & @TRK
JC      main3b         ;Ret C set if abort/error
CALL    wrlba          ;Update LBA on drive
main3b: CALL    ZCRLF
        jmp     mainloop

NEXT$SECT:
LDA     @SEC
INR     A
CPI     MAXSEC-1
JNC    RANGE$ERROR
STA     @SEC
CALL    wrlba          ;Update LBA on drive
CALL    ZCRLF
jmp     mainloop

RANGE$ERROR:
LXI     D,RANGE$MSG
CALL    PSTRING
jmp     mainloop

PREV$SEC:
LDA     @SEC
ORA     A
JZ      RANGE$ERROR
DCR     A

```

```

    STA    @SEC
    CALL   wrlba          ;Update LBA on drive
    CALL   ZCRLF
    jmp    mainloop

POWER$UP:
                ;Set the drive to spin up (for hard disk connections)
    CALL   spinup
    jmp    mainloop

POWER$DOWN:
                ;Set the drive to spin down (for hard disk connections)
    CALL   spindown
    jmp    mainloop

DISPLAY:
                ;Do we have detail sector data display flag on or off
    LDA    @DisplayFlag
    CMA                    ;flip it
    STA    @DisplayFlag
    jmp    mainloop      ;Update display and back to next menu command

SEQ$RD:
                ;Do sequential reads
    CALL   SequentialReads
    jmp    mainloop

CPMBOOT:
                ;Boot CPM from IDE system tracks -- if present
    MVI    A,0             ;Load from track 0,sec 1, head 0 (Always)
    STA    @SEC           ;Remember sectors are numbered +1
    XRA    A
    STA    @TRK+1
    STA    @TRK

    MVI    A,SEC$COUNT   ;Count of sectors (12)
    STA    @SecCount
    LXI    H,CPM$ADDRESS  ;DMA address where the CPMLDR resides in RAM (100H)
    SHLD  @DMA

NextRCPM:
    CALL   wrlba          ;Update LBA on drive
    CALL   DISPLAYposition ;Display current Track,sector,head#
    CALL   ZCRLF

    LHLD  @DMA
    CALL  READSECTOR     ;read a sector
    SHLD  @DMA

    LDA    @SecCount
    DCR   A
    STA    @SecCount
    JZ    LOAD$DONE

    LHLD  @SEC
    INX   H
    SHLD  @SEC          ;Note we assume we always will stay on track 0 in this special case
    JMP   NextRCPM

LOAD$DONE:
    MVI   E,REGstatus    ;Check the R/W status when done
    CALL  IDErd8D
    BIT   0,D
    JNZ   CPMLoadErr     ;Z if no errors
    LXI   H,CPM$ADDRESS
    MOV   A,M
    CPI   31H            ;EXPECT TO HAVE 31H @80H IE. LD SP,80H
    JZ    CPM$ADDRESS    ;AS THE FIRST INSTRUCTION. IF OK JP to 100H in RAM
                    ;Else fall through to error msg

CPMLoadErr:
    LXI   D,CPM$ERROR    ;Drive R/W Error
    CALL  PSTRING
    JMP   CPMBOOT

N$RD$SEC:
                ;Read N sectors >>>> NOTE no check is made to not overwrite
                ;CPM etc. in high RAM
    LXI   D,ReadN$MSG
    CALL  PSTRING
    CALL  GETHEX
    JC    mainloop      ;Abort if ESC (C flag set)

    STA    @SecCount     ;store sector count
    LXI   H,buffer      ;Point to buffer

```

```

        SHLD    @DMA

NextRSec:
        LXI    D,ReadingN$MSG
        CALL   PSTRING
        CALL   wrlba          ;Update LBA on drive
        CALL   DISPLAYposition ;Display current Track,sector,head#

        LHLD   @DMA
        CALL   READSECTOR
        SHLD   @DMA

        LDA    @SecCount
        DCR    A
        STA    @SecCount
        JZ     mainloop

        LHLD   @SEC
        INX    H
        SHLD   @SEC
        MOV    A,L          ;0 to 62 CPM Sectors
        CPI    MAXSEC-1
        JNZ    NextRSec

        LXI    H,0          ;Back to CPM sector 0
        SHLD   @SEC
        LHLD   @TRK        ;Bump to next track
        INX    H
        SHLD   @TRK
        MOV    A,L          ;0-FFH tracks (only)
        JNZ    NextRSec

        LXI    D,AtEnd      ;Tell us we are at end of disk
        CALL   PSTRING
        JMP    mainloop

N$WR$SEC:
        LXI    D,msgsure    ;Write N sectors
        CALL   PSTRING      ;Are you sure?
        CALL   ZCI
        CALL   upper
        CPI    'Y'
        JNZ    main2c

        LXI    D,WriteN$MSG
        CALL   PSTRING
        CALL   GETHEX
        JC     mainloop     ;Abort if ESC (C flag set)

        STA    @SecCount    ;store sector count
        LXI    H,buffer     ;Point to buffer
        SHLD   @DMA

NextWSec:
        LXI    D,WritingN$MSG
        CALL   PSTRING
        CALL   wrlba        ;Update LBA on drive
        CALL   DISPLAYposition ;Display current Track,sector,head#

        LHLD   @DMA
        CALL   WRITESECTOR  ;Actully, Sector/track values are already updated
        SHLD   @DMA        ;above in wrlba, but WRITESECTOR is used in multiple places.
                           ;A repeat does no harm -- speed is not an issue here

        LDA    @SecCount
        DCR    A
        STA    @SecCount
        JZ     mainloop

        LHLD   @SEC
        INX    H
        SHLD   @SEC
        MOV    A,L          ;0 to 62 CPM Sectors
        CPI    MAXSEC-1
        JNZ    NextWSec

        LXI    H,0          ;Back to CPM sector 0
        SHLD   @SEC
        LHLD   @TRK        ;Bump to next track

```

```

INX      H
SHLD    @TRK
MOV     A,L          ;0-FFH tracks (only)
ORA     A
JNZ     NextWSec

LXI     D,AtEnd      ;Tell us we are at end of disk
CALL    PSTRING
JMP     mainloop

FORMAT:                                     ;Format (Fill sectors with E5's for CPM directory empty)
LXI     D,FORMAT$MSG
CALL    PSTRING
LXI     D,msgsure    ;Are you sure?
CALL    PSTRING
CALL    ZCI
CALL    upper
CPI     'Y'
JNZ     mainloop
LXI     H,buffer      ;Fill buffer with 0E5's (512 of them)
MVI     B,0
Fill0:  MVI     A,0E5H ;<-- Sector fill character (0E5's for CPM)
MOV     M,A
INX     H
MOV     M,A
INX     H
DJNZ   Fill0
CALL    ZCRLF
;
NEXT$FORMAT:
LXI     H,buffer
SHLD   @DMA
CALL    WRITESECTOR ;Will return error if there was one
JZ     main9b       ;Z means the sector write was OK
CALL    ZCRLF
JMP     mainloop
main9b: CALL    ZEOL      ;Clear line cursor is on
CALL    DISPLAYposition ;Display actual current Track,sector,head#
CALL    ZCSTS        ;Any keyboard character will stop display
CPI     01H         ;CPM Says something there
JNZ     WRNEXTSEC1
CALL    ZCI          ;Flush character
LXI     D,CONTINUE$MSG
CALL    PSTRING
CALL    ZCI
CPI     ESC
JZ     mainloop
CALL    ZCRLF
WRNEXTSEC1:
LHLD   @SEC
INX     H
SHLD   @SEC         ;0 to MAXSEC CPM Sectors
MOV     A,L
CPI     MAXSEC
JNZ     NEXT$FORMAT

LXI     H,0          ;Back to CPM sector 0
SHLD   @SEC
LHLD   @TRK        ;Bump to next track
INX     H
SHLD   @TRK
MOV     A,L          ;0-FFH tracks (only)
CPI     MAXTRK
JNZ     NEXT$FORMAT

LXI     D,FormatDone ;Tell us we are all done.
CALL    PSTRING
JMP     mainloop

BACKUP:                                     ;Backup the CPM partition to another area on the SAME CF-card/disk
LXI     D,CopyMsg
CALL    PSTRING
CALL    ZCI
CALL    upper
CPI     'Y'
JNZ     mainloop

```



```

LXI    H,0            ;Start with CPM sector 0
SHLD   @SEC
SHLD   @SEC1
SHLD   @SEC2         ;and on second partition
SHLD   @TRK          ;and track 0
SHLD   @TRK1
LXI    H,MAXTRK+0200H+1 ;<<<<< VIP This assumes CPM3 is on tracks 0-MAXTRK. (0-FFH) >>>>
SHLD   @TRK2         ;It skips an area to be safe. However if you have other stuff on this
                                ;CF card at that location (eg DOS partition) change this value

CALL   ZCRLF
CALL   ZCRLF

```

NextCopy1:

```

CALL   ZEOL          ;Clear line cursor is on
LXI    D,RBackup$MSG ;for each track update display
CALL   PSTRING
LDA    @TRK1+1       ;High TRK byte
CALL   phex
LDA    @TRK1         ;Low TRK byte
CALL   phex
LXI    D,WBackup$MSG
CALL   PSTRING
LDA    @TRK2+1       ;High TRK byte
CALL   phex
LDA    @TRK2         ;Low TRK byte
CALL   phex
LXI    D,H$Msg
CALL   PSTRING

```

NextCopy:

```

LDA    @SEC1
STA    @SEC
LHLD   @TRK1
SHLD   @TRK
CALL   wrlba         ;Update LBA on "1st" drive

LXI    H,buffer      ;Point to buffer
SHLD   @DMA
CALL   READSECTOR    ;Get sector data to buffer

LDA    @SEC2
STA    @SEC
LHLD   @TRK2
SHLD   @TRK
CALL   wrlba         ;Update LBA on "2nd" drive

LXI    H,buffer      ;Point to buffer
SHLD   @DMA
CALL   WRITESECTOR   ;Write buffer data to sector

CALL   ZCSTS         ;Any keyboard character will stop display
CPI    01H           ;CPM Says something there
JNZ    BKNEXTSEC1
CALL   ZCI           ;Flush character
LXI    D,CONTINUE$MSG
CALL   PSTRING
CALL   ZCI
CPI    ESC
JZ     mainloop

```

BKNEXTSEC1:

```

LHLD   @SEC
INX    H
SHLD   @SEC1
SHLD   @SEC2
MOV    A,L           ;0 to 62 CPM Sectors
CPI    MAXSEC-1
JNZ    NextCopy

LXI    H,0           ;Back to CPM sector 0
SHLD   @SEC1
SHLD   @SEC2

LHLD   @TRK1        ;Bump to next track
INX    H
SHLD   @TRK1

LHLD   @TRK2        ;Bump to next track
INX    H

```

```

SHLD  @TRK2

LHLD  @TRK1      ;Check if we are done
MOV   A,L        ;0-FFH tracks (only)
CPI   MAXTRK
JNZ   NextCopy1

LXI   D,BackupDone ;Tell us we are all done.
CALL  PSTRING
JMP   mainloop

```

```

RESTORE:                ;Restore disk from backup partition

```

```

LXI   D,RestoreMsg
CALL  PSTRING
CALL  ZCI
CALL  upper
CPI   'Y'
JNZ   mainloop

LXI   H,0          ;Start with CPM sector 0
SHLD  @SEC
SHLD  @SEC1
SHLD  @SEC2        ;and on second partition
SHLD  @TRK         ;and track 0
SHLD  @TRK1
LXI   H,MAXTRK+0200H+1 ;<<<<< VIP This assumes CPM3 is on tracks 0-MAXTRK. (0-FFH) >>>>
SHLD  @TRK2        ;It skips an area to be safe. However if you have other stuff on this
                          ;CF card at that location (eg DOS partition) change this value

CALL  ZCRLF
CALL  ZCRLF

```

```

NextRestore1:

```

```

CALL  ZEOL         ;Clear line cursor is on
LXI   D,RBackup$MSG ;for each track update display
CALL  PSTRING
LDA   @TRK2+1      ;High TRK byte
CALL  phex
LDA   @TRK2        ;Low TRK byte
CALL  phex
LXI   D,WBackup$MSG
CALL  PSTRING
LDA   @TRK1+1      ;High TRK byte
CALL  phex
LDA   @TRK1        ;Low TRK byte
CALL  phex
LXI   D,H$Msg
CALL  PSTRING

```

```

NextRestore:

```

```

LDA   @SEC2        ;Point to backup partition
STA   @SEC
LHLD  @TRK2
SHLD  @TRK
CALL  wrlba        ;Update LBA on "1st" drive

LXI   H,buffer     ;Point to buffer
SHLD  @DMA
CALL  READSECTOR   ;Get sector data to buffer

LDA   @SEC1
STA   @SEC
LHLD  @TRK1
SHLD  @TRK
CALL  wrlba        ;Update LBA on "2nd" drive

LXI   H,buffer     ;Point to buffer
SHLD  @DMA
CALL  WRITESECTOR  ;Write buffer data to sector

CALL  ZCSTS        ;Any keyboard character will stop display
CPI   01H         ;CPM Says something there
JNZ   RESNEXTSEC1
CALL  ZCI          ;Flush character
LXI   D,CONTINUE$MSG
CALL  PSTRING
CALL  ZCI
CPI   ESC

```

```

        JZ      mainloop

RESNEXTSEC1:
        LHL    @SEC
        INX    H
        SHLD   @SEC1
        SHLD   @SEC2
        MOV    A,L          ;0 to 62 CPM Sectors
        CPI    MAXSEC-1
        JNZ    NextRestore

        LXI    H,0          ;Back to CPM sector 0
        SHLD   @SEC1
        SHLD   @SEC2

        LHL    @TRK1       ;Bump to next track
        INX    H
        SHLD   @TRK1

        LHL    @TRK2       ;Bump to next track
        INX    H
        SHLD   @TRK2

        LHL    @TRK2       ;Check if we are done
        MOV    A,L          ;0-FFH tracks (only)
        CPI    MAXTRK
        JNZ    NextRestore1

        LXI    D,RestoreDone ;Tell us we are all done.
        CALL   PSTRING
        JMP    mainloop

ERROR: LXI    D, msgErr     ;CMD error msg
        CALL   PSTRING
        JMP    mainloop

;----- Support Routines -----
;Find top of RAM to setup a stack

MEMSIZE:LXI    H,0FFFFH    ;START FROM THE TOP DOWN
MEMSZ2: MOV    A,M
        CMA
        MOV    M,A
        CMP    M
        CPMA          ;PUT BACK WHAT WAS THERE
        MOV    M,A
        JZ     GOTTOP
        DCX    H          ;TRY 100H BYTES LOWER
        JMP    MEMSZ2     ;KEEP LOOKING FOR RAM
GOTTOP: POP    B          ;RESTORE [BC], Note we draw an extra word from stack
        RET              ;Will end up at "AHEAD" near start of ROM

driveid:CALL   IDEwaitnotbusy ;Do the IDentify drive command, and return with the buffer
;filled with info about the drive
;If Busy return NZ
        RC
        MVI    D,COMMANDid
        MVI    E,REGcommand
        CALL   IDEwr8D      ;issue the command

        CALL   IDEwaitdrq   ;Wait for Busy=0, DRQ=1
        JC     SHOWerrors

        MVI    B,0          ;256 words
        LXI    H,IDbuffer   ;Store data here
        CALL   MoreRD16     ;Get 256 words of data from REGdata port to [HL]
        RET

spinup:
spup2: MVI    D,COMMANDspinup
        MVI    E,REGcommand
        CALL   IDEwr8D
        CALL   IDEwaitnotbusy
        JC     SHOWerrors
        ORA    A          ;Clear carry
        ret

```

```

spindown:                ;Tell the drive to spin down
    CALL IDEwaitnotbusy
    JC SHOWerrors
    MVI D,COMMANDspindown
    jmp spup2

SequentialReads:
    CALL IDEwaitnotbusy ;sequentially read sectors one at a time from current position
    JC SHOWerrors
    CALL ZCRLF

NEXTSEC:
    LXI H,buffer        ;Point to buffer
    SHLD @DMA

    CALL READSECTOR    ;If there are errors they will show up in READSECTOR
    JZ SEQOK
    LXI D,CONTINUE$MSG
    CALL PSTRING
    CALL ZCI
    CPI ESC            ;Abort if ESC
    RZ

SEQOK: CALL ZEOL        ;Clear line cursor is on
    CALL DISPLAYposition ;Display current Track,sector,head#

    LXI H,buffer        ;Point to buffer
    SHLD @DMA

    LDA @DisplayFlag    ;Do we have detail sector data display flag on or off
    ORA A              ;NZ = on
    CNZ HEXDUMP
    CALL ZCRLF
    CALL ZCRLF
    CALL ZCRLF

    CALL ZCSTS          ;Any keyboard character will stop display
    CPI 01H            ;CPM Says something there
    JNZ NEXTSEC1
    CALL ZCI            ;Flush character
    LXI D,CONTINUE$MSG
    CALL PSTRING
    CALL ZCI
    CPI ESC
    RZ
    CALL ZCRLF

NEXTSEC1:
    LHLD @SEC
    INX H
    SHLD @SEC
    MOV A,L            ;0 to 62 CPM Sectors
    CPI MAXSEC-1
    JNZ NEXTSEC

    LXI H,0            ;Back to CPM sector 0
    SHLD @SEC
    LHLD @TRK          ;Bump to next track
    INX H
    SHLD @TRK
    JMP NEXTSEC        ;Note will go to last sec on disk unless stopped
;
;
DISPLAYposition:        ;Display current track,sector & head position
    LXI D,msgCPMTRK    ;Display in LBA format
    CALL PSTRING        ;---- CPM FORMAT ----
    LDA @TRK+1         ;High TRK byte
    CALL phex
    LDA @TRK            ;Low TRK byte
    CALL phex

    LXI D,msgCPMSEC
    CALL PSTRING        ;SEC = (16 bits)
    LDA @SEC+1         ;High Sec
    CALL phex
    LDA @SEC            ;Low sec
    CALL phex
;---- LBA FORMAT ----
    LXI D, msgLBA
    CALL PSTRING        ;(LBA = 00 (<-- Old "Heads" = 0 for these drives).

```

```

LDA    @DRIVE$TRK+1    ;High "cylinder" byte
CALL   phex
LDA    @DRIVE$TRK      ;Low "cylinder" byte
CALL   phex
LDA    @DRIVE$SEC
CALL   phex
LXI    D, MSGBracket  ;)$
CALL   PSTRING
RET

;
printname:                ;Send text up to [B]
                ;Text is low byte high byte format
    INX    H
    MOV    C,M
    CALL   ZCO
    DCX    H
    MOV    C,M
    CALL   ZCO
    INX    H
    INX    H
    DCR    B
    JNZ    printname
    ret

;
ZCRLF:
    PUSH   PSW
    MVI    C,CR
    CALL   ZCO
    MVI    C,LF
    CALL   ZCO
    POP    PSW
    RET

;
ZEOL:                ;CR and clear current line
    MVI    C,CR
    CALL   ZCO
    MVI    C,CLEAR    ;Note hardware dependent, (Use 80 spaces if necessary)
    CALL   ZCO
    RET

;
;
;    ;Print a string in [DE] up to '$'
PSTRING:
    PUSH   B
    PUSH   D
    PUSH   H
    XCHG
PSTRX:  MOV    A,M
    CPI    '$'
    JZ     DONEP
    MOV    C,A
    CALL   ZCO
    INX    H
    JMP    PSTRX
DONEP:  POP    H
    POP    D
    POP    B
    RET

;
;
SHOWerrors:
    CALL   ZCRLF
    MVI    E,REGstatus    ;Get status in status register
    CALL   IDErd8D
    MOV    A,D
    ANI    1H
    JNZ    MoreError    ;Go to REGerr register for more info
                ;All OK if 01000000
    PUSH   PSW
    ANI    80H
    JZ     NOT7
    LXI    D,DRIVE$BUSY    ;Drive Busy (bit 7) stuck high.    Status =
    CALL   PSTRING
    JMP    DONEERR
NOT7:   ANI    40H
    JNZ    NOT6
    LXI    D,DRIVE$NOT$READY    ;Drive Not Ready (bit 6) stuck low.    Status =
    CALL   PSTRING

```

```

NOT6:  JMP     DONEERR
      ANI     20H
      JNZ     NOT5
      LXI     D,DRIVE$WR$FAULT ;Drive write fault.   Status =
      CALL    PSTRING
      JMP     DONEERR
NOT5:  LXI     D,UNKNOWN$ERROR
      CALL    PSTRING
      JMP     DONEERR

MoreError:                                ;Get here if bit 0 of the status register indicted a problem
      MVI     E,REGerr                      ;Get error code in REGerr
      CALL    IDErD8D
      MOV     A,D
      PUSH    PSW

      ANI     10H
      JZ      NOTE4
      LXI     D,SEC$NOT$FOUND
      CALL    PSTRING
      JMP     DONEERR

NOTE4: ANI     80H
      JZ      NOTE7
      LXI     D,BAD$BLOCK
      CALL    PSTRING
      JMP     DONEERR
NOTE7: ANI     40H
      JZ      NOTE6
      LXI     D,UNRECOVER$ERR
      CALL    PSTRING
      JMP     DONEERR
NOTE6: ANI     4H
      JZ      NOTE2
      LXI     D,INVALID$CMD
      CALL    PSTRING
      JMP     DONEERR
NOTE2: ANI     2H
      JZ      NOTE1
      LXI     D,TRK0$ERR
      CALL    PSTRING
      JMP     DONEERR
NOTE1: LXI     D,UNKNOWN$ERROR1
      CALL    PSTRING
      JMP     DONEERR

DONEERR:POP    PSW
      PUSH    PSW
      CALL    ZBITS
      CALL    ZCRLF
      POP     PSW
      ORA     A                          ;Set Z flag
      STC                                ;Set Carry flag
      RET

;-----
; Print a 16 bit number in RAM located @ [HL] (Note Special Low Byte First)
;
printparm:
      INX     H                          ;Index to high byte first
      MOV     a,M
      CALL    PHEX
      DCX     H                          ;Now low byte
      MOV     a,M
      CALL    PHEX
      RET

;
; Print an 8 bit number, located in [A]

PHEX:  PUSH    PSW
      PUSH    B
      PUSH    PSW
      RRC
      RRC
      RRC
      RRC
      CALL    ZCONV
      POP     PSW
      CALL    ZCONV

```

```

        POP      B
        POP      PSW
        RET
;
ZCONV: ANI      0FH          ;HEX to ASCII and print it
        ADI      90H
        DAA
        ACI      40H
        DAA
        MOV      C,A
        CALL     ZCO
        RET

;DISPLAY BIT PATTERN IN [A]
;
ZBITS:  PUSH     PSW
        PUSH     B
        PUSH     D
        MOV      E,A
        MVI      B,8
BQ2:    DB      0CBH,23H
        SLAR     E          ;Z80 Op code for SLA A,E
        MVI      A,18H
        ADC      A
        MOV      C,A
        CALL     ZCO
        DJNZ     BQ2
        POP      D
        POP      B
        POP      PSW
        RET

ghex32lba:                                ;get CPM style Track# & Sector# data and convert to LBA format
        LXI      D,ENTER$SECL ;Enter sector number
        CALL     PSTRING
        CALL     GETHEX          ;get 2 HEX digits
        RC
        STA      @SEC           ;Note: no check data is < MAXSEC, sectors start 0,1,2,3....
        CALL     ZCRLF

        LXI      D,ENTER$TRKH ;Enter high byte track number
        CALL     PSTRING
        CALL     GETHEX          ;get 2 HEX digits
        RC
        STA      @TRK+1
        CALL     ZCRLF

        LXI      D,ENTER$TRKL ;Enter low byte track number
        CALL     PSTRING
        CALL     GETHEX          ;get 2 more HEX digits
        RC
        STA      @TRK
        CALL     ZCRLF
        XRA      A
        ORA      A              ;To return NC
        RET
;
;
GETHEX: CALL     GETCMD          ;Get a character from keyboard & ECHO
        CPI      ESC
        JZ       HEXABORT
        CPI      '/'
        JC       HEXABORT      ;check 0-9, A-F
        CPI      'F'+1
        JNC      HEXABORT
        CALL     ASBIN          ;Convert to binary
        RLC
        RLC                    ;Shift to high nibble
        RLC
        RLC
        MOV      B,A            ;Store it
        CALL     GETCMD          ;Get 2nd character from keyboard & ECHO
        CPI      ESC
        JZ       HEXABORT
        CPI      '/'
        JC       HEXABORT      ;check 0-9, A-F

```

```

CPI      'F'+1
JNC      HEXABORT
CALL     ASBIN          ;Convert to binary
ORA      B              ;add in the first digit
ORA      A              ;To return NC
RET

HEXABORT:
STC      ;Set Carry flag
RET

;
;
GETCMD:  CALL  ZCI          ;GET A CHARACTER, convert to UC, ECHO it
CALL     UPPER
CPI      ESC
RZ              ;Don't echo an ESC
PUSH     PSW          ;Save it
PUSH     B
MOV      C,A
CALL     ZCO          ;Echo it
POP      B
POP      PSW          ;get it back
RET

;
;
UPPER:   CPI      'a'          ;Convert LC to UC
RC              ;must be >= lowercase a
CPI      'z'+1          ; else go back...
RNC              ;must be <= lowercase z
SUI      'a'-'A'        ; else go back...
RET          ;subtract lowercase bias

;
;
ASBIN:   SUI      30H          ;ASCII TO BINARY CONVERSION ROUTINE
CPI      0AH
RM
SUI      07H
RET

;
;
HEXDUMP:                ;Print a hexdump of the data in the 512 byte buffer (@DMA)
PUSH     PSW          ;Save everything
PUSH     B
PUSH     D
PUSH     H

CALL     ZCRLF        ;CR/LF first
MVI     D,32          ;Print 32 lines total
MVI     B,16          ;16 characters across
SHLD    @StartLineHex ;Save the buffer location for ASCII display below
LXI     H,0
SHLD    @ByteCount

SF172:  CALL     ZCRLF
LHLD    @ByteCount
MOV     A,H
CALL    PHEX          ;Print byte count in sector
MOV     A,L
CALL    PHEX
PUSH    D
LXI     D,16
DAD     D
POP     D
SHLD    @ByteCount    ;store for next time
CALL    BLANK
LHLD    @StartLineHex
SHLD    @StartLineASCII ;Store for ASCII display below

SF175:  MOV     A,M
CALL    LBYTE          ;Display [A] on CRT/LCD
INX     H
DJNZ   SF175
SHLD    @StartLineHex ;Save for next line later
CALL    ShowAscii      ;Now translate to ASCII and display
MVI     B,16          ;16 characters across for next line
DCR     D
JNZ     SF172          ;Have we done all 32 lines

;
CALL    ZCRLF
POP     H              ;Get back original registers
POP     D

```



```

POP      B
POP      PSW
RET

ShowAscii:                ;Now show as ascii info
      LHLD   @StartLineASCII
      MVI   B,16           ;16 ASCII characters across
XF172: CALL   BLANK        ;send a space character
      CALL   BLANK
XF175:  MOV   A,M
      ANI   7FH
      CPI   ' '           ;FILTER OUT CONTROL CHARACTERS
      JNC   XT33
XT22:  MVI   A,'.'
XT33:  CPI   07CH
      JNC   XT22
      MOV   C,A           ;SET UP TO SEND
      PUSH  B
      CALL  ZCO
      POP   B
      INX  H             ;Next position in buffer
      DJNZ XF175
      RET

;
BLANK: PUSH  B
      PUSH  H
      MVI  C,' '
      CALL  ZCO
      POP  H
      POP  B
      RET

;
LBYTE: PUSH  PSW
      RRC
      RRC
      RRC
      RRC
      CALL  SF598
      POP  PSW
SF598: CALL  ZCONV
      RET

;
;
;=====
;
;   IDE Drive BIOS Routines written in a format that can be used directly with CPM3
;
;=====
;
IDEinit:                ;Initilze the 8255 and drive then do a hard reset on the drive,
      MVI   A,READcfg8255 ;Config 8255 chip (10010010B), read mode on return
      OUT  IDEportCtrl    ;Config 8255 chip, READ mode

                                ;Hard reset the disk drive
                                ;For some reason some CF cards need to the RESET line
                                ;pulsed very carefully. You may need to play around
                                ;with the pulse length. Symptoms are: incorrect data comming
      MVI   A,IDERstline  ;back from a sector read (often due to the wrong sector being read)
      OUT  IDEportC

      MVI   B,020H        ;<<<<< fine tune later

ResetDelay:
      DCR  B
      JNZ  ResetDelay    ;Delay (reset pulse width)

      XRA  A
      OUT  IDEportC      ;No IDE control lines asserted (just bit 7 of port C)
      CALL DELAY$32

      MVI  D,11100000b   ;Data for IDE SDH reg (512bytes, LBA mode,single drive,head 0000)
                                ;For Trk,Sec,head (non LBA) use 10100000
                                ;Note. Cannot get LBA mode to work with an old Seagate Medalist 6531 drive.
                                ;have to use teh non-LBA mode. (Common for old hard disks).

      MVI  E,REGshd      ;00001110,(0EH) for CS0,A2,A1,
      CALL IDEwr8D       ;Write byte to select the MASTER device

;
      MVI  B,0FFH        ;<<< May need to adjust delay time for hard disks

WaitInit:
      MVI  E,REGstatus   ;Get status after initilization

```

```

CALL    IDErd8D      ;Check Status (info in [D])
MOV     A,D
ANI     80H
JZ      DoneInit    ;Return if ready bit is zero
MVI     A,2
CALL    DELAYX      ;Long delay, drive has to get up to speed
DCR     B
JNZ     WaitInit
CALL    SHOWerrors   ;Ret with NZ flag set if error (probably no drive)
RET

DoneInit:
XRA    A
RET

DELAYX: STA    @DELAYStore
PUSH   B
LXI    B,0FFFFH    ;<<< May need to adjust delay time to allow cold drive to
DELAY2: LDA    @DELAYStore ; get up to speed.
DELAY1: DCR    A
JNZ    DELAY1
DCX    B
MOV    A,C
ORA    B
JNZ    DELAY2
POP    B
RET

DELAY$FF:XRA    A      ;Longer Delay
JR     DELAY3

DELAY$32:
MVI    A,40        ;DELAY ~32 MS (DOES NOT SEEM TO BE CRITICAL)
DELAY3: MVI    B,0
M0:    DJNZ   M0
DCR    A
JNZ    DELAY3
RET

;Read a sector, specified by the 4 bytes in LBA
;Z on success, NZ call error routine if problem
READSECTOR:
CALL    wrlba      ;Tell which sector we want to read from.
;Note: Translate first in case of an error otherwise we
;will get stuck on bad sector
CALL    IDEwaitnotbusy ;make sure drive is ready
JC      SHOWerrors ;Returned with NZ set if error

MVI    D,COMMANDread
MVI    E,REGcommand
CALL    IDEwr8D    ;Send sec read command to drive.
CALL    IDEwaitdrq ;wait until it's got the data
JC      SHOWerrors

LHLD   @DMA        ;DMA address
MVI    B,0        ;Read 512 bytes to [HL] (256X2 bytes)

MoreRD16:
MVI    A,REGdata   ;REG regsiter address
OUT    IDEportC

ORI    IDErdline   ;08H+40H, Pulse RD line
OUT    IDEportC

IN     IDEportA    ;Read the lower byte first (Note early versions had high byte then low byte
MOV    M,A        ;this made sector data incompatable with other controllers).
INX   H
IN     IDEportB    ;THEN read the upper byte
MOV    M,A
INX   H

MVI    A,REGdata   ;Deassert RD line
OUT    IDEportC
DJNZ  MoreRD16

MVI    E,REGstatus
CALL   IDErd8D
MOV    A,D
ANI    1H

```

```

CNZ      SHOWerrors      ;If error display status
RET

;Write a sector, specified by the 3 bytes in LBA (@ IX+0)",
;Z on success, NZ to error routine if problem

WRITESECTOR:
CALL     wrlba           ;Tell which sector we want to read from.
;Note: Translate first in case of an error otherwise we
;will get stuck on bad sector

CALL     IDEwaitnotbusy ;make sure drive is ready
JC       SHOWerrors

MVI      D,COMMANDwrite
MVI      E,REGcommand
CALL     IDEwr8D        ;tell drive to write a sector
CALL     IDEwaitdrq    ;wait unit it wants the data
JC       SHOWerrors

;

LHLD     @DMA
MVI      B,0           ;256X2 bytes

MVI      A,WRITEcfg8255
OUT      IDEportCtrl

WRSEC1:  MOV     A,M
INX      H
OUT      IDEportA      ;Write the lower byte first (Note early versions had high byte then low byte
MOV      A,M           ;this made sector data incompatable with other controllers).
INX      H
OUT      IDEportB      ;THEN High byte on B
MVI      A,REGdata
PUSH     PSW
OUT      IDEportC      ;Send write command
ORI      IDEwrline     ;Send WR pulse
OUT      IDEportC
POP      PSW
OUT      IDEportC
DJNZ    WRSEC1

MVI      A,READcfg8255 ;Set 8255 back to read mode
OUT      IDEportCtrl

MVI      E,REGstatus
CALL     IDErd8D
MOV      A,D
ANI      1H
CNZ      SHOWerrors    ;If error display status
RET

;
;wrlba:
;Write the logical block address to the drive's registers
;Note we do not need to set the upper nibble of the LBA
;It will always be 0 for these small drives

LDA      @SEC          ;LBA mode, Low sectors go directly
INR      A             ;Sectors are numbered 1 -- MAXSEC (even in LBA mode)
STA      @DRIVE$SEC    ;For Diagnostic Display Only
MOV      D,A
MVI      E,REGsector   ;Send info to drive
CALL     IDEwr8D       ;Note: For drive we will have 0 - MAXSEC sectors only

LHLD     @TRK
MOV      A,L
STA      @DRIVE$TRK
MOV      D,L           ;Send Low TRK#
MVI      E,REGcylinderLSB
CALL     IDEwr8D

MOV      A,H
STA      @DRIVE$TRK+1
MOV      D,H           ;Send High TRK#
MVI      E,REGcylinderMSB
CALL     IDEwr8D

MVI      D,1           ;For now, one sector at a time
MVI      E,REGsecct
CALL     IDEwr8D
RET
;

```

```

;
IDEwaitnotbusy:                ;ie Drive READY if 01000000
    MVI    B,0FFH
    MVI    A,0FFH                ;Delay, must be above 80H for 4MHz Z80. Leave longer for slower drives
    STA    @DELAYStore

MoreWait:
    MVI    E,REGstatus           ;wait for RDY bit to be set
    CALL   IDErd8D
    MOV    A,D
    ANI    11000000B
    XRI    01000000B
    JZ     DoneNotbusy
    DCR    B
    JNZ    MoreWait
    LDA    @DELAYStore           ;Check timeout delay
    DCR    A
    STA    @DELAYStore
    JNZ    MoreWait
    STC                                ;Set carry to indicqate an error
    ret

DoneNotBusy:
    ORA    A                    ;Clear carry it indicate no error
    RET

                                ;Wait for the drive to be ready to transfer data.
                                ;Returns the drive's status in Acc

IDEwaitdrq:
    MVI    B,0FFH
    MVI    A,0FFH                ;Delay, must be above 80H for 4MHz Z80. Leave longer for slower drives
    STA    @DELAYStore

MoreDRQ:
    MVI    E,REGstatus           ;wait for DRQ bit to be set
    CALL   IDErd8D
    MOV    A,D
    ANI    10001000B
    CPI    00001000B
    JZ     DoneDRQ
    DCR    B
    JNZ    MoreDRQ
    LDA    @DELAYStore           ;Check timeout delay
    DCR    A
    STA    @DELAYStore
    JNZ    MoreDRQ
    STC                                ;Set carry to indicate error
    RET

DoneDRQ:
    ORA    A                    ;Clear carry
    RET

;-----
; Low Level 8 bit R/W to the drive controller.  These are the routines that talk
; directly to the drive controller registers, via the 8255 chip.
; Note the 16 bit I/O to the drive (which is only for SEC R/W) is done directly
; in the routines READSECTOR & WRITESECTOR for speed reasons.
;
IDErd8D:                        ;READ 8 bits from IDE register in [E], return info in [D]
    MOV    A,E
    OUT    IDEportC              ;drive address onto control lines

    ORI    IDErdline             ;RD pulse pin (40H)
    OUT    IDEportC              ;assert read pin

    IN     IDEportA
    MOV    D,A                    ;return with data in [D]

    MOV    A,E                    ;<---Ken's suggestion
    OUT    IDEportC              ;deassert RD pin

    XRA    A
    OUT    IDEportC              ;Zero all port C lines
    ret

IDEwr8D:                        ;WRITE Data in [D] to IDE register in [E]
    MVI    A,WRITEcfig8255       ;Set 8255 to write mode
    OUT    IDEportCtrl

```

```

MOV     A,D                      ;Get data put it in 8255 A port
OUT     IDEportA

MOV     A,E                      ;select IDE register
OUT     IDEportC

ORI     IDEwrline                ;lower WR line
OUT     IDEportC

MOV A,E                          ;<-- Kens Suggestion, raise WR line
OUT IDEportC

XRA     A                        ;Deselect all lines including WR line
OUT     IDEportC

MVI     A,READcfg8255            ;Config 8255 chip, read mode on return
OUT     IDEportCtrl
RET

;
;
SIGN$ON:  DB     SCROLL,QUIT,NO$ENHANCEMENT,FAST,BELL,CR,LF
          DB     'ROM IDE Diagnostic (V1.1)',CR,LF,'$'
INIT$ERROR: DB     'Init. Error.',CR,LF,'$'
ID$ERROR:  DB     'Drive ID Error.',CR,LF,'$'
msgmdl:   DB     CR,LF,'Model: $'
msgsn:    DB     'S/N:  $'
msgrev:   DB     'Rev:  $'
msgcy:    DB     'Cyl: $'
msghd:    DB     ', Heads: $'
msgsc:    DB     ', Sec: $'
msgCPMTRK: DB     'CPM TRK = $'
msgCPMSEC: DB     ' CPM SEC = $'
msgLBA:   DB     ' (LBA = 00$'
MSGBracket DB     ')$'
SP$MSG    DB     'Stack=$'

CMD$STRING1: DB     CR,LF,LF
          DB     '(L) Set LBA value (R) Read Sec to Buffer (W) Write Buffer '
          DB     'to Sector',CR,LF
          DB     '(D) Sec.Disp ON/OFF (S) Sequential Sec Read (F) Format Disk',CR,LF
          DB     '(V) Read N Sectors (X) Write N Sectors (B) Backup disk',CR,LF
          DB     '(G) Restore Backup (I) Next Sector (J) Previous Sector',CR,LF
          DB     '(U) Power Up (N) Power Down (C) Boot CPM',CR,LF
          DB     '(ESC) Quit',CR,LF,LF,'Current settings:- $'

Prompt:   db     CR,LF,LF,'Enter cmd >$'
msgsure:  DB     CR,LF,'Change data, (Y/N)...$'
msgrd:    DB     CR,LF,'Sec Read OK',CR,LF,'$'
msgwr:    DB     CR,LF,'Sec Write OK',CR,LF,'$'
GET$LBA:  DB     'Enter CPM style TRK & SEC values (in hex).',CR,LF,'$'
SEC$RW$ERROR DB     'Drive Error, Status = $'
ERR$REG$DATA DB     'Drive Error, Error = $'
ENTER$SECL DB     'Start Sec # (xxH) = $'
ENTER$TRKL DB     'Track # (LOW byte, xxH) = $'
ENTER$TRKH DB     'Track # (HIGH byte, xxH) = $'
ENTER$HEAD DB     'Head # (01-0f) = $'
ENTER$COUNT DB     'Number of sec to R/W = $'
DRIVE$BUSY DB     'Drive Busy. Status = $'
DRIVE$NOT$READY DB     'Drive Ready. Status = $'
DRIVE$WR$FAULT DB     'Drive write fault. Status = $'
UNKNOWN$ERROR DB     'Unknown error. Status = $'
BAD$BLOCK DB     'Bad Sector ID. Error = $'
UNRECOVER$ERR DB     'Data error. Error = $'
READ$ID$ERROR DB     'Error reading Drive ID',CR,LF,'$'
SEC$NOT$FOUND DB     'Sec not found. Error = $'
INVALID$CMD DB     'Invalid Command. Error = $'
TRK0$ERR DB     'Track 0 not found. Error = $'
UNKNOWN$ERROR1 DB     'Unknown Error. Error = $'
CONTINUE$MSG DB     CR,LF,'To Abort enter ESC.$'
FORMAT$MSG DB     'Fill sec with E5.$'
ReadN$MSG DB     CR,LF,'Read sectors from disk to RAM buffer.'
          DB     CR,LF,'How many sec (xx HEX):$'
WriteN$MSG DB     CR,LF,'Write sectors in RAM buffer to disk.'
          DB     CR,LF,'How many sec (xx HEX):$'
ReadingN$MSG DB     CR,LF,'Reading Sec at:- $'
WritingN$MSG DB     CR,LF,'Writing Sec at:- $'
msgErr DB     CR,LF,'Invalid option!$'
FormatDone DB     CR,LF,'Format done.',CR,LF,'$'

```

```
backupDone      DB      CR,LF,'Backup done.',CR,LF,'$'  
CopyMsg         DB      CR,LF,'Backup disk (Y/N)...$ '  
AtEnd           DB      CR,LF,'At end!',CR,LF,'$'  
RBackup$MSG    DB      'Reading Trk: $'  
WBackup$MSG    DB      'H. Writing Trk: $'  
H$Msg          DB      'H$'  
RestoreMsg     DB      CR,LF,'Get backup (Y/N)...$'  
RestoreDone    DB      CR,LF,'Backup done.',CR,LF,'$'  
RANGE$MSG      DB      CR,LF,'Range Err.',CR,LF,'$'  
CPM$ERROR      DB      CR,LF,'CPM Boot Err.',CR,LF,'$'  
;END
```