

```

; Test Program to interact with the CPM3 type BIOS for the S100Computers IDE interface board
;=====
;
;      V1.7      3/1/2010 ;Removed Z80 Code (so it can be translated to 8086 code later)
;      V2.0      1/23/2011      ;Updated to accomodate two CF cards (Master/Slave) & better menu options
;                               ;note I still have more work to do with this but what is here seem OK.
;
;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 continuous 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
REGsecnt 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 occurred 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 occurred
; bit 1: IDX vendor specific
; bit 0: ERR 1=error occurred
;
;
;Equates for display on SD Systems Video Board (Used In CPM Debugging mode only)
SCROLL EQU 01H ;Set scrool direction UP.
LF EQU 0AH
CR EQU 0DH
BS EQU 08H ;Back space (required for sector display)
BELL EQU 07H
SPACE EQU 20H
TAB EQU 09H ;TAB ACROSS (8 SPACES FOR SD-BOARD)
ESC EQU 1BH
CLEAR EQU 1CH ;SD Systems Video Board, Clear to EOL. (Use 80 spaces if EOL not available
;on other video cards)
;
;SECSIZE 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

```



```

JNZ Display1
LXI D,CMD$STRING1 ;List command options (Turn display option to on)
JP Display2
Display1:
LXI D,CMD$STRING2 ;List command options (Turn display option to off)
Display2:
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

main1: CPI 'R' ;Read Sector @ LBA to the RAM buffer
JNZ main2

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

main2: CPI 'W' ;Write data in RAM buffer to sector @ LBA
JNZ main3
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

main3: CPI 'L' ;Set the logical block address
JNZ main4
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

main4: CPI 'U' ;cause the drive to spin up (for hard disk connections)
JNZ main5
CALL spinup
jmp mainloop

main5: CPI 'N' ;cause the drive to spin down (for hard disk connections)
JNZ main6
CALL spindown
jmp mainloop

main6: CPI 'Q' ;quit (for hard disk connections)
JNZ main7
jmp 0

main7: CPI 'D'
JNZ main8
LDA @DisplayFlag ;Do we have detail sector data display flag on or off
CMA ;flip it
STA @DisplayFlag
jmp mainloop ;Update display and back to next menu command

```

```

main8:  CPI      'S'
        JNZ     main9
        CALL   SequentialReads
        JMP     mainloop

main9:  CPI      'v'                ;Read N sectors >>>> NOTE no check is made to not overwrite CPM etc. in high RAM
        JNZ     main10
        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#

        LHL   @DMA
        CALL   READSECTOR
        SHLD   @DMA

        LDA    SecCount
        DCR   A
        STA    SecCount
        JZ    mainloop

        LHL   @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
        LHL   @TRK
        INX   H
        SHLD   @TRK
        JP    NextRSec

main10: CPI      'X'                ;Write N sectors
        JNZ     main11

        LXI    D,msgsure;Are you sure?
        CALL   PSTRING
        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#

        LHL   @DMA
        CALL   WRITESECTOR
        SHLD   @DMA

        LDA    SecCount
        DCR   A
        STA    SecCount
        JZ    mainloop

        LHL   @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
        LHL   @TRK
        INX   H
        SHLD   @TRK
        JP    NextWSec

main11: CPI      'F'                ;Format (Fill sectors with E5's for CPM directory empty)

```

```

    jnz     mainloop
    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    WRNEXTSECL
    CALL   ZCI             ;Flush character
    LXI    D,CONTINUE$MSG
    CALL   PSTRING
    CALL   ZCI
    CPI    ESC
    JZ     mainloop
    CALL   ZCRLF
WRNEXTSECL:
    LHLD   @SEC
    INX    H
    SHLD   @SEC           ;0 to MAXSEC CPM Sectors
    MOV    A,L
    CPI    MAXSEC
    JNZ    NEXT$FORMAT

    LXI    H,0            ;Back to CPM sector 0
    SHLD   @SEC
    LHLD   @TRK          ;Bump to next track
    INX    H
    SHLD   @TRK
    JMP    NEXT$FORMAT    ;Note will go to last sec on disk unless stopped
                                ;Will actually hang if we get to end of disk!
;
;Do the IDentify drive command, and return with the buffer
;filled with info about the drive
driveid:
    CALL   IDEwaitnotbusy
    RC     ;If Busy return NZ
    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:
    MVI    D,COMMANDspinup
spup2:  MVI    E,REGcommand
    CALL   IDEwr8D
    CALL   IDEwaitnotbusy
    JC     SHOWerrors
    ORA    A              ;Clear carry
    ret

                                ;Tell the drive to spin down
spindown:
    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
;
;
;----- Support Routines -----
;
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

ZCSTS:
IF CPM
PUSH   B
PUSH   D
PUSH   H
MVI    C,CONST
CALL   BDOS    ;Returns with 1 in [A] if character at keyboard
POP    H
POP    D
POP    B
CPI    1
RET

ELSE
IN     0H    ;Get Character in [A]
ANI    02H
RZ
MVI    A,01H
ORA    A
RET

ENDIF

;
ZCO:   ;Write character that is in [C]
IF CPM
PUSH   PSW
PUSH   B
PUSH   D
PUSH   H
MOV    E,C
MVI    C,WRCON
CALL   BDOS
POP    H
POP    D
POP    B
POP    PSW
RET

ELSE
PUSH   PSW
ZCO1:  IN     0H    ;Show Character
ANI    04H
JZ     ZCO1
MOV    A,C
OUT    1H
POP    PSW
RET

ENDIF

ZCI:   ;Return keyboard character in [A]
IF CPM
PUSH   B
PUSH   D
PUSH   H
MVI    C,RDCON
CALL   BDOS
POP    H
POP    D
POP    B
RET

ELSE
ZCI1:  IN     0H    ;Get Character in [A]
ANI    02H
JZ     ZCI1
IN     01H
RET

ENDIF

;
;
; ;Print a string in [DE] up to '$'
PSTRING:
IF CPM
MVI    C,PRINT
JMP    BDOS    ;PRINT MESSAGE,

ELSE
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
ENDIF
;
;
SHOWerrors:
IF      NOT DEBUG
        ORA     A           ;Set NZ flag
        STC     A           ;Set Carry Flag
        RET
ELSE
        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
;
        PUSH   PSW           ;All OK if 01000000
        ANI     80H           ;save for return below
        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$READY ;Drive Not Ready (bit 6) stuck low.  Status =
        CALL    PSTRING
        JMP     DONEERR
NOT6:   ANI     20H
        JNZ     NOT5
        LXI     D,DRIVE$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    A           ;Set Carry flag
        RET
ENDIF
;
;-----
; 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           ;280 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     '/'
          ;check 0-9, A-F
          JC     HEXABORT
          CPI     'F'+1
          JNC    HEXABORT
          CALL    ASBIN        ;Convert to binary
          RLC                 ;Shift to high nibble
          RLC
          RLC
          RLC
          MOV     B,A         ;Store it

```

```

CALL GETCMD ;Get 2nd character from keyboard & ECHO
CPI ESC
JZ HEXABORT
CPI '/' ;check 0-9, A-F
JC HEXABORT
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
IF NOT CPM
PUSH PSW ;Save it
PUSH B
MOV C,A
CALL ZCO ;Echo it
POP B
POP PSW ;get it back
ENDIF
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 ;10010010b
OUT IDEportCtrl ;Config 8255 chip, READ mode

MVI a,IDERstline
OUT IDEportC ;Hard reset the disk drive

MVI B,020H ;<<<<< fine tune later
ResetDelay:
DCR B
JNZ ResetDelay ;Delay (reset pulse width)
XRA A
OUT IDEportC ;No IDE control lines asserted

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
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

;

```

```

;
;
;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

;
    LHL    @DMA                ;DMA address
    MVI     B,0                ;Read 512 bytes to [HL] (256X2 bytes)
MoreRD16:
    MVI     A,REGdata          ;REG register 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 incompatible 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 until it wants the data
    JC      SHOWerrors

;
    LHL    @DMA                ;256X2 bytes
    MVI     B,0

    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    M,A                ;this made sector data incompatible 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 Diaplay 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,REGsecnt
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:
MOV    A,E                ;READ 8 bits from IDE register in [E], return info in [D]
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]

XRA    A

```

```

OUT      IDEportC      ;Zero all port C lines
ret
;
;
IDEwr8D:                ;WRITE Data in [D] to IDE register in [E]
MVI      A,WRITEcfg8255 ;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
NOP

XRA      A              ;Deselect all lines including WR line
OUT      IDEportC

MVI      A,READcfg8255 ;Config 8255 chip, read mode on return
OUT      IDEportCtrl
RET
;
;
SIGN$ON: DB             CR,LF,'IDE Disk Drive Test Program (V2.0) (Using CPM3 BIOS Routines)',CR,LF
DB             'CPM Track,Sectors --> LBA mode',LF,CR,'$'
INIT$ERROR: DB         'Initializing Drive Error.',CR,LF,'$'
ID$ERROR:   DB         'Error obtaining Drive ID.',CR,LF,'$'
INIT$DR$OK: DB         'Drive Initialized OK.',CR,LF,LF,'$'
msgmdl:     DB         'Model: $'
msgsn:      DB         'S/N:  $'
msgrev:     DB         'Rev:  $'
msgcy:      DB         'Cylinders: $'
msghd:      DB         ', Heads: $'
msgsc:      DB         ', Sectors: $'
msgCPMTRK:  DB         'CPM TRK = $'
msgCPMSEC:  DB         ' CPM SEC = $'
msgLBA:     DB         ' (LBA = 00$'
MSGBracket  DB         ')$'

CMD$STRING1: DB        CR,LF,LF,'                MAIN MENU',CR,LF
DB             '(L) Set LBA value  (R) Read Sector to Buffer  (W) Write Buffer  '
DB             'to Sector',CR,LF
DB             '(D) Display ON      (S) Sequential Sec Read   (F) Format Disk',CR,LF
DB             '(V) Read N Sectors (X) Write N Sectors',CR,LF
DB             '(U) Power Up      (N) Power Down              (ESC) Quit',CR,LF,LF
DB             LF,'Current settings:- $'

CMD$STRING2: DB        CR,LF,LF,'                MAIN MENU',CR,LF
DB             '(L) Set LBA value  (R) Read Sector to Buffer  (W) Write Buffer  '
DB             'to Sector',CR,LF
DB             '(D) Display OFF     (S) Sequential Sec Read   (F) Format Disk',CR,LF
DB             '(V) Read N Sectors (X) Write N Sectors',CR,LF
DB             '(U) Power Up      (N) Power Down              (ESC) Quit',CR,LF,LF
DB             'Current settings:- $'

Prompt:     db         CR,LF,LF,'Please enter command >$'
msgsure:    DB         CR,LF,'Warning: this will change data on the drive, '
DB         'are you sure? (Y/N)...$'
msgrd:      DB         CR,LF,'Sector Read OK',CR,LF,'$'
msgwr:      DB         CR,LF,'Sector Write OK',CR,LF,'$'
GET$LBA:    DB         'Enter CPM style TRK & SEC values (in hex).',CR,LF,'$'
SEC$RW$ERR  DB         'Drive Error, Status Register = $'
ERR$REG$DATA DB        'Drive Error, Error Register = $'
ENTER$SECL  DB         'Starting sector number,(xxH) = $'
ENTER$TRKL  DB         'Track number (LOW byte, xxH) = $'
ENTER$TRKH  DB         'Track number (HIGH byte, xxH) = $'
ENTER$HEAD  DB         'Head number (01-0f) = $'
ENTER$COUNT DB        'Number of sectors to R/W = $'
DRIVE$BUSY  DB         'Drive Busy (bit 7) stuck high. Status = $'
DRIVE$NOT$READY DB      'Drive Ready (bit 6) stuck low. Status = $'
DRIVE$WR$FAULT DB      'Drive write fault. Status = $'
UNKNOWN$ERROR DB       'Unknown error in status register. Status = $'
BAD$BLOCK   DB         'Bad Sector ID. Error Register = $'
UNRECOVER$ERR DB       'Uncorrectable data error. Error Register = $'
READSID$ERROR DB       'Error setting up to read Drive ID',CR,LF,'$'
SEC$NOT$FOUND DB       'Sector not found. Error Register = $'
INVALID$CMD DB         'Invalid Command. Error Register = $'
TRK0$ERR    DB         'Track Zero not found. Error Register = $'
UNKNOWN$ERROR1 DB      'Unknown Error. Error Register = $'
CONTINUE$MSG DB        CR,LF,'To Abort enter ESC. Any other key to continue. $'
FORMAT$MSG  DB         'Fill sectors with 0H (e.g for CPM directory sectors).$'
ReadN$MSG   DB         CR,LF,'Read multiple sectors from current disk/CF card to RAM buffer.'
DB         CR,LF,'How many 512 byte sectores (xx HEX):$'
WriteN$MSG  DB         CR,LF,'Write multiple sectors RAM buffer current disk/CF card.'
DB         CR,LF,'How many 512 byte sectores (xx HEX):$'
ReadingN$MSG DB        CR,LF,'Reading Sector at:- $'
WritingN$MSG DB        CR,LF,'Writing Sector at:- $'
;
;
; ----- RAM usage -----

```

```
RAMAREA      DB      '          RAM AREA ----->'
@DMA         DW      buffer
@DRIVE$SEC   DB      0H
@DRIVE$TRK   DW      0H
@DisplayFlag DB      0FFH          ;Display of sector data initially ON
;
@SEC         DW      0H
@TRK         DW      0H
StartLineHex DW      0H
StartLineASCII DW     0H
ByteCount    DW      0H
SecCount     DW      0H

;
DELAYStore  DB      0H
;
STACK       DW      0H
;
IDbuffer    DS      512
;
buffer      DB      '<--Start buffer area'   ;a 512 byte buffer
            DS      476
            DB      'End of buffer-->'
;
;END
```