```
TITLE  'CPM86+ DISK BIOS MODULE FOR S100 Computers IDE CONTROLLER'

;      06/14/2011   V1.0          ; Initial version taken from old XCOMP BIOS

                                  ; Control characters Console I/O Board MAY recognize
SCROLL        EQU    01H          ; Set scrool direction UP.
BELL          EQU    07H
SPACE         EQU    20H
TAB           EQU    09H          ; TAB ACROSS (8 SPACES FOR SD-BOARD)
CR            EQU    0DH
LF            EQU    0AH
FF            EQU    0CH
QUIT          EQU    11H          ; Turns off any screen enhancements (flashing, underline etc).
ESC           EQU    1BH
DELETE        EQU    7FH
BACKS         EQU    08H
CLEAR         EQU    1AH          ; TO CLEAR SCREEN
NO_ENHANCE    EQU    17H          ; Turn off Screen enhancements (Flashing etc)

;Propeller Console IO S-100 board or SD SYSTEMS VIDIO BOARD FOR CONSOLE I/O(<---These must configured for your hardware)

KEYSTAT       EQU    0H           ;These are ONLY use for displaying IDE Board errors (Bypassing CPM86+)
KEYIN         EQU    01H          ;Console input port. Normally the Propeller Driven S-100 Console-IO Board
KEYOUT        EQU    01H          ;Console output port. Normally the Propeller Driven S-100 Console-IO Board

;Ports for 8255 chip. Change these to specify where the 8255 is addressed,
;and which of the 8255's ports are connected to which IDE signals.
;The first three control which 8255 ports have the control signals,
;upper and lower data bytes.  The fourth one is for mode setting for the
;8255 to configure its ports, which must correspond to the way that
;the first three lines define which ports are connected.

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
IDECtrlPort   EQU    033H         ;8255 configuration port
IDEDrivePort  EQU    034H         ;Bit 0 = 0 for Drive A: = 1 for F:

IDE_Reset_Delay     EQU    020          ;Time delay for reset/initilization (~66 uS, with 8MHz 8086, 1 I/O wait state)

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

;IDE control lines for use with IDEportC.  Change these 8
;constants to reflect where each signal of the 8255 each of the
;IDE control signals is connected.  All the control signals must
;be on the same port, but these 8 lines let you connect them to
;whichever pins on that port.
```

```
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 pins

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
;
MAXSEC        EQU    3DH    ;Sectors per track for CF my Memory drive, Kingston CF 8G. (CPM format, 0-3D)
                            ;Note this will also work with a Seagate 6531 IDE drive
SECSIZ        EQU    512    ;Always 512 byte sectors
;
; 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


;CPM86+ obtains ALL its Disk Bios variables using the 8086 SS:[BP] register + an offset
```

```
;Here are the crital offset values:-

iopb_mcnt     equ     byte ptr 15 ;Sector count for multi sector R/W's
iopb_drive    equ     byte ptr 14 ;Selected Drive
iopb_track    equ     word ptr 12 ;Selected track
iopb_sector   equ     word ptr 10 ;Selected Sector
iopb_dmaseg   equ     word ptr 8  ;Selected DMA Segment
iopb_dmaoff   equ     word ptr 6  ;Selected Segment offset


       CSEG

       EXTRN  ?PMSG:NEAR, ?SMSG:NEAR, @sysdat:word
       EXTRN  ?CO:NEAR, ?CSTS:NEAR, ?CI:NEAR


;------HARD DISK INITILIZATION ROUTINE---------------------------------

       ;;;;; HDINIT:
HDINIT0:RET                           ;Pospone until Logon
HDINIT1:RET


;------HARD DISK LOGIN ROUTINE-----------------------------------------
;      Entry  BX  =  DPH OFFSET
;             CL  =  SELECTED DISK
;      Exit   BX  =  0 IF ILLEGAL SELECT
;                 =  OFFSET OF DPH RELATIVE TO O.S. DATA SEGMENT
; Note Login is called only once (or again after a ^C). Normal drive selection should be done
; in the actual Sector R/W Routines

HDLOGIN0:
       PUSH   BX
       XOR    AL,AL                  ;For our CPM86+ system will be only 0 or 5
       OUT    IDEDrivePort,AL               ;Select IDE Drive 0 (CPM A:)
       MOV    SI,Offset SAYSIGNON0
       CALL   ?SMSG                  ;Announce on speaker (If present)
       CALL   COMMON$INIT
       JB     LOGIN_ERROR
       POP    BX
       RET

HDLOGIN1:
       PUSH   BX
       MOV    AL,1
       OUT    IDEDrivePort,AL               ;Select IDE Drive 1 (CPM F: in our case)
       MOV    SI,Offset SAYSIGNON1
       CALL   ?SMSG                  ;Announce on speaker (If present)
       CALL   COMMON$INIT
       JB     LOGIN_ERROR
```

```
       POP    BX
       RET

LOGIN_ERROR:
       MOV    SI,Offset HD_LOGIN_ERR           ;"Drive Login Error"
       CALL   ?SMSG
       POP    BX                        ;Lets CPM86+ know there is a problem
       MOV    BX,0
       RET



;;;;; HDRD
;-------- ROUTINE READS N SECTORS FROM THE DISK:------------------------------
;      Entry  BX  = DPH offset
;      Exit:  AL  = 0 IF NO ERROR
;                 = 1 IF PHYSICAL SECTOR READ ERROR
;-------------------------------------------------------------------------------

HDRD:  CALL   wrlba               ;Setup Drive, Track, Sector.
       CALL   IDEwaitnotbusy             ;Make sure drive is ready
       JNB    L_19               ;Carry flag set if problem
       JMP    SHOWerrors         ;Returned to CPM86 with AL=1 (DS: valid for messages)

L_19:  MOV    DH,COMMANDread
       MOV    DL,REGcommand
       CALL   IDEwr8D                   ;Send Sec read command to drive.
       CALL   IDEwaitdrq          ;Wait until it's got the data
       JNB    L_20               ;Carry flag set if problem
       JMP    SHOWerrors         ;Returned to CPM86 with AL=1 (DS: valid for messages)

L_20:  push   ES                 ;ES is altered here (not used in this BIOS, but save anyway)
       PUSH   DI
       les    DI,DWORD PTR iopb_dmaoff [BP]
       MOV    CH,0               ;ALWAYS read 512 bytes to [CH] (256X2 bytes)
       CLI                       ;Just in case

MoreRD16:
       MOV    AL,REGdata         ;REG regsiter address
       OUT    IDEportC,AL

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

       IN     AL,IDEportA        ;Read the lower byte first
       MOV    ES:Byte Ptr [DI],AL
       INC    DI
       IN     AL,IDEportB        ;THEN read the upper byte
       MOV    ES:Byte Ptr [DI],AL
       INC    DI
```

```
       MOV    AL,REGdata          ;Deassert RD line
       OUT    IDEportC,AL
       DEC    CH                  ;256 times for 512 bytes
       JNZ    MoreRD16

       STI                        ;Turn back on
       POP    DI
       POP    ES                  ;Get back ES from above

       MOV    DL,REGstatus
       CALL   IDErd8D                    ;Returned data in DH
       MOV    AL,DH
       AND    AL,1H               ;Ret AL=0 for all OK
       JZ     NEXT_SECTOR_RD
       JMP    SHOWerrors          ;Returned to CPM86 with AL=1 (DS: valid for messages)

NEXT_SECTOR_RD:                           ;We have done one sector, are there more
       dec    iopb_mcnt [BP]             ;Decrease multi-sector count
       jz     RD_SEC_DONE         ;Stay here until all sectors read
       mov    ax,iopb_sector [BP]
       inc    ax
       cmp    ax,MAXSEC
       jb     SameRtrak           ;Still on same track
       inc    iopb_track [BP]            ;Next track
       xor    ax,ax               ;Note how sectors are numbered in CPM86+ 0,1,2,3...
SameRtrak:
       mov    iopb_sector [BP],ax
       add    iopb_dmaoff [BP],secsiz    ;Note how we inc. the DMA address
       jmp    HDRD                ;CPM86+ says it will never cross a segment boundry

RD_SEC_DONE:
       xor    al,al
       ret




;-------- ROUTINE WRITES N SECTORS TO THE DISK:-------------------------------
;      Entry  BX  = DPH offset
;      Exit:  AL  = 0 IF NO ERROR
;                 = 1 IF PHYSICAL ERROR
;------------------------------------------------------------------------------

HDWRT: CALL   wrlba               ;Setup Drive, Track, Sector.
       CALL   IDEwaitnotbusy              ;Make sure drive is ready
       JNB    L_22                ;Carry flag set if problem
       JMP    SHOWerrors          ;Returned to CPM86 with AL=1 (DS: valid for messages)

L_22:  MOV    DH,COMMANDwrite
```

```
        MOV     DL,REGcommand
        CALL    IDEwr8D                 ;tell drive to write a sector
        CALL    IDEwaitdrq              ;wait unit it wants the data
        JNB     L_23                    ;Carry flag set if problem
        JMP     SHOWerrors              ;Returned to CPM86 with AL=1 (DS: valid for messages)

L_23:   PUSH    DS                      ;Save Local BIOS [DS] value
        PUSH    SI
        lds     SI,DWORD PTR iopb_dmaoff [BP]

        MOV     AL,WRITEcfg8255
        OUT     IDECtrlPort,AL
        MOV     CH,0                    ;Read 512 bytes to [CH] (256X2 bytes)
        CLI                             ;Just in case

MoreWR16:
        MOV     AL,DS:Byte Ptr [SI]
        INC     SI
        OUT     IDEportA,AL             ;Write the lower byte first
        MOV     AL,DS:Byte Ptr [SI]
        INC     SI
        OUT     IDEportB,AL             ;THEN High byte on B

        MOV     AL,REGdata
        PUSH    AX
        OUT     IDEportC,AL             ;Send write command
        OR      AL,IDEwrline            ;Send WR pulse
        OUT     IDEportC,AL
        POP     AX
        OUT     IDEportC,AL             ;Send write command
        DEC     CH                      ;256 times for 512 bytes
        JNZ     MoreWR16

        STI                             ;Turn back on ints
        POP     SI
        POP     DS                      ;Get back Local BIOS [DS] value
        MOV     AL,READcfg8255                  ;Set 8255 back to read mode
        OUT     IDECtrlPort,AL

        MOV     DL,REGstatus
        CALL    IDErd8D                 ;Data returned in DH
        MOV     AL,DH
        AND     AL,1H
        JZ      NEXT_SECTOR_WR
        JMP     SHOWerrors              ;Returned to CPM86 with AL=1 (DS: valid for messages)

NEXT_SECTOR_WR:
        dec     iopb_mcnt [BP]                  ;Decrease multi-sector count
        jz      WR_SEC_DONE             ;Stay here until all sectors read
        mov     ax,iopb_sector [BP]
```

```
       inc    ax
       cmp    ax,MAXSEC
       jb     SameWtrak           ;Still on same track
       inc    iopb_track [BP]
       xor    ax,ax               ;Note how sectors are numbered in CPM86+ 0,1,2,3...
SameWtrak:
       mov    iopb_sector [BP],ax
       add    iopb_dmaoff [BP],secsiz    ;Note how we inc. the DMA address
       jmp    HDWRT               ;CPM86+ says it will never cross a segment boundry

WR_SEC_DONE:
       xor    al,al
       ret



;============================== IDE DRIVE SUPPORT ROUTINES ======================================

; Initilize the IDE Board.
COMMON$INIT:                       ;Initilze the 8255, then do a hard reset on the CURRENT Selected drive,
       MOV    AL,READcfg8255             ;10010010b
       OUT    IDECtrlPort,AL             ;Config 8255 chip, READ mode

       MOV    AL,IDErstline
       OUT    IDEportC,AL         ;Hard reset the disk drive

       MOV    CH,IDE_Reset_Delay  ;Time delay for reset/initilization (~66 uS, with 8MHz 8086, 1 I/O wait state)
ResetDelay:
       DEC    CH
       JNZ    ResetDelay          ;Delay (IDE reset pulse width)
       XOR    AL,AL
       OUT    IDEportC,AL         ;No IDE control lines asserted

       CALL   DELAY_32            ;Allow time for CF/Drive to recover

       MOV    DH,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 the non-LBA mode. (Common for old hard disks).
       MOV    DL,REGshd           ;00001110,(0EH) for CS0,A2,A1,
       CALL   IDEwr8D                    ;Write byte to select the MASTER device

       MOV    CH,080H                    ;<<< May need to adjust delay time
WaitInit:
       MOV    DL,REGstatus        ;Get status after initilization
       CALL   IDErd8D                    ;Check Status (info in [D])
       MOV    AL,DH
       AND    AL,80H
       JZ     DoneInit            ;Return if ready bit is zero
```

```
       PUSH   CX
       MOV    CX,0FFFFH
DELAY2:       MOV    DH,2                ;May need to adjust delay time to allow cold drive to
DELAY1:       DEC    DH                  ;to speed
       JNZ    DELAY1
       DEC    CX
       MOV    AL,CL
       OR     AL,CH
       JNZ    DELAY2
       POP    CX
       DEC    CH
       JNZ    WaitInit
       MOV    SI,Offset DR_INIT_ERROR ;"Drive Initilization Failed"
       CALL   ?PMSG
       CALL   SHOWerrors          ;Ret with Carry flag set if error (probably no drive)
       RET
DoneInit:
       XOR    AL,AL               ;Clear Carry bit
       RET

DELAY_32:
       MOV    AL,40               ;DELAY ~32 MS (DOES NOT SEEM TO BE CRITICAL)
DELAY3:       MOV    BL,0
M0:    DEC    BL
       JNZ    M0
       DEC    AL
       JNZ     DELAY3
       RET


; Setup Drive R/W for correct Drive/Track/Sector
wrlba: MOV    AL,iopb_drive [BP]  ;Check we have the correct current drive
       OR     AL,AL
       JZ     DriveOK
       MOV    AL,1                ;Must be second Drive on the board. (Note for speed, no CPM BIOS check is done)
DriveOK:
       OUT    IDEDrivePort,AL           ;This is actully faster than checking current drive in RAM flage etc.

       mov    ax,iopb_sector [BP] ;LBA mode, Low sectors go directly to hardware
       INC    AX                  ;Sectors are numbered 1 -- MAXSEC (even in LBA mode)
       MOV    DH,AL               ;Only need low byte
       MOV    DL,REGsector        ;Send info to drive
       CALL   IDEwr8D
                                  ;Note: For drive we will have 0 - MAXSEC sectors only
       MOV    AX,IOPB_TRACK [BP]  ;GET REQUESTED TRACK
       MOV    DH,AL               ;Send Low TRK#
       MOV    DL,REGcylinderLSB
       CALL   IDEwr8D                   ;AH is unaltered

       MOV    DH,AH               ;Send High TRK#
```

```
        MOV     DL,REGcylinderMSB
        CALL    IDEwr8D

        MOV     DH,1                ;For now, one sector at a time
        MOV     DL,REGseccnt
        CALL    IDEwr8D
        RET

IDEwaitnotbusy:                         ;ie Drive READY if 01000000
        MOV     CH,0FFH
        MOV     AL,0FFH                 ;Delay, must be above 80H for 4MHz Z80. Leave longer for slower drives
        PUSH    BX
        MOV     Byte Ptr DELAYStore,AL
MoreWait:
        MOV     DL,REGstatus        ;wait for RDY bit to be set
        CALL    IDErd8D                 ;Returns with data in DH
        MOV     AL,DH
        AND     AL,11000000B
        XOR     AL,01000000B
        JZ      DoneNotbusy
        DEC     CH
        JNZ     MoreWait
        MOV     AL,Byte Ptr DELAYStore      ;Check timeout delay
        DEC     AL
        MOV     Byte Ptr DELAYStore,AL
        JNZ     MoreWait
        STC                         ;Set carry to indicate an error
        POP     BX
        RET
DoneNotBusy:
        OR      AL,AL               ;Clear carry it indicate no error
        POP     BX
        RET

                                    ;Wait for the drive to be ready to transfer data.
                                    ;Returns the drive's status in Acc
IDEwaitdrq:
        MOV     CH,0FFH
        MOV     AL,0FFH                 ;Delay, must be above 80H for 4MHz Z80. Leave longer for slower drives
        PUSH    BX
        MOV     Byte Ptr DELAYStore,AL

MoreDRQ:
        MOV     DL,REGstatus        ;wait for DRQ bit to be set
        CALL    IDErd8D                 ;Return with data in DH
        MOV     AL,DH
        AND     AL,10001000B
        CMP     AL,00001000B
        JZ      DoneDRQ
        DEC     CH
```

```
        JNZ     MoreDRQ
        MOV     AL,Byte Ptr DELAYStore     ;Check timeout delay
        DEC     AL
        MOV     Byte Ptr DELAYStore,AL
        JNZ     MoreDRQ
        STC                         ;Set carry to indicate error
        POP     BX
        RET
DoneDRQ:
        OR      AL,AL               ;Clear carry
        POP     BX
        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     AL,DL
        OUT     IDEportC,AL         ;drive address onto control lines

        OR      AL,IDErdline        ;RD pulse pin (40H)
        OUT     IDEportC,AL         ;assert read pin

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

        MOV     AL,DL               ;<---Ken Robbins suggestion
        OUT     IDEportC,AL         ;drive address onto control lines

        XOR     AL,AL
        OUT     IDEportC,AL         ;Zero all port C lines
        RET

IDEwr8D:                            ;WRITE Data in [DH] to IDE register in [AL]
        MOV     AL,WRITEcfg8255           ;Set 8255 to write mode
        OUT     IDECtrlPort,AL

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

        MOV     AL,DL               ;select IDE register
        OUT     IDEportC,AL

        OR      AL,IDEwrline        ;lower WR line
        OUT     IDEportC,AL
```

```
        MOV     AL,DL               ;<-- Ken Robbins suggestion, raise WR line
        OUT     IDEportC,AL         ;deassert RD pin

        XOR     AL,AL               ;Deselect all lines including WR line
        OUT     IDEportC,AL

        MOV     AL,READcfg8255              ;Config 8255 chip, read mode on return
        OUT     IDECtrlPort,AL
        RET



; Show errors returned from IDE Board directly of Console
SHOWerrors:
        CALL    CRLF
        MOV     DL,REGstatus        ;Get status in status register
        CALL    IDErd8D
        MOV     AL,DH
        TEST    AL,1H
        JNZ     MoreError           ;Go to  REGerr register for more info about the error
                                    ;All OK if 01000000
        TEST    AL,80H
        JZ      NOT7
        MOV     SI,Offset DRIVE_BUSY       ;Drive Busy (bit 7) stuck high.   Status =
        CALL    ?PMSG
        JMP     DONEERR

NOT7:   TEST    AL,40H
        JNZ     NOT6
        MOV     SI,Offset DRIVE_NOT_READY ;Drive Not Ready (bit 6) stuck low.  Status =
        CALL    ?PMSG
        JMPS    DONEERR

NOT6:   TEST    AL,20H
        JNZ     NOT5
        MOV     SI,Offset DRIVE_WR_FAULT ;Drive write fault.    Status =
        CALL    ?PMSG
        JMPS    DONEERR

NOT5:   MOV     SI,Offset UNKNOWN_ERROR
        CALL    ?PMSG
        JMPS    DONEERR

MoreError:                          ;Get here if bit 0 of the status register indicted a problem
        MOV     DL,REGerr           ;Get error code in REGerr
        CALL    IDErd8D
        MOV     AL,DH

        TEST    AL,10H
        JZ      NOTE4
```

```
       MOV    SI,Offset SEC_NOT_FOUND
       CALL   ?PMSG
       JMPS   DONEERR

NOTE4: TEST   AL,80H
       JZ     NOTE7
       MOV    SI,Offset BAD_BLOCK
       CALL   ?PMSG
       JMPS   DONEERR

NOTE7: TEST   AL,40H
       JZ     NOTE6
       MOV    SI,Offset UNRECOVER_ERR
       CALL   ?PMSG
       JMPS   DONEERR

NOTE6: TEST   AL,4H
       JZ     NOTE2
       MOV    SI,Offset INVALID_CMD
       CALL   ?PMSG
       JMPS   DONEERR

NOTE2: TEST   AL,2H
       JZ     NOTE1
       MOV    SI,Offset TRK0_ERR
       CALL   ?PMSG
       JMPS   DONEERR

NOTE1: MOV    SI,Offset UNKNOWN_ERROR1
       CALL   ?PMSG
       JMPS   DONEERR

DONEERR:CALL  AL_binout           ;Show bit pattern
       XOR    AL,AL
       INC    AL                  ;Set NZ flag and return wiyh AL=1 (For Sector R/W's)
       STC                        ;Set Carry flag (For Drive initilization routine)
       RET


;      BINARY OUTPUT              ;Send what is in [al] in bits
AL_binout:                        ;No registers altered (except AL)
       push   cx
       mov    cx,8
binout1: push cx
       shl    al,1
       jb     bout1
       mov    cl,'0'
       push   ax
       call   ?CO
       pop    ax
```

```
       jmp    binend
bout1: mov    cl,'1'
       push   ax
       call   ?CO
       pop    ax
binend:       pop    cx
       loop   binout1
       pop    cx
       ret

;      AX_HEXOUT                 ;Output the 4 hex digits in [AX] (For debugging Only)
AX_hexout:
       PUSH   AX
       MOV    AL,AH
       CALL   AL_hexout
       POP    AX
       CALL   AL_hexout
       RET



;      AL_HEXOUT                 ;Output the 2 hex digits in [AL]
AL_hexout:                       ;No registers altered (except AL)
       push   cx
       push   ax
       mov    cl,4               ;first isolate low nibble
       shr    al,cl
       call   hexdigout
       pop    ax
       call   hexdigout          ;get upper nibble
       pop    cx
       ret

hexdigout:                       ;Convert nibble to ascii
       and    al,0fh
       add    al,90h
       daa
       adc    al,40h
       daa
       mov    cl,al
       call   ?CO
       ret

;      SIMPLE SEND CRLF
CRLF:  push   cx
       push   bx
       mov    cl,cr
       call   ?CO
       mov    cl,lf
       call   ?CO
```

```
        pop    bx
        pop    cx
        ret



        DSEG

        PUBLIC @DPHA, @DPHF

;------------------------- DATA STORAGE AREA --------------------------------


;------      Disk Parameter Header Equates --------------------------------
;
;       +------+------+------+------+------+------+------+------+
;  00h  |     XLT     |                    | DOPEN|             |
;       +------+------+------+------+------+------+------+------+
;  08h         |     DPB     |     CSV     |     ALV     |    DIRBCB   |
;       +------+------+------+------+------+------+------+------+
;  10h  |    DATBCB   |    HSHTBL   |     INIT    |    LOGIN    |
;       +------+------+------+------+------+------+------+------+
;  18h  |     READ    |    WRITE    | UNIT | CHNNL|NFLAGS|
;       +------+------+------+------+------+------+------+
;
;       Use two different tables (in case we wish to modify later for a HD or something)

@DPHA           EQU    OFFSET $
                DW     0000H         ;TRANSLATE TABLE (NONE FOR HARD DISK\CFCARD 0)
                DB     0,0,0H        ;SCRATCH AREA
                DB     0H            ;DOOR OPEN FLAG
                DB     0,0H          ;SCRATCH AREA
                DW     DPB0          ;DISK PARAMETER TABLE
                DW     0FFFFH,0FFFFH ;CHECKSUM, ALLOCATION VECTOR
                DW     0FFFFH        ;DIRECTORY BCB
                DW     0FFFFH        ;DATA BCB
                DW     0FFFFH        ;HASH TABLE
                DW     HDINIT0
                DW     HDLOGIN0
                DW     HDRD
                DW     HDWRT
                DB     0             ;UNIT 0 ON THIS CONTROLLER
                DB     0             ;ALWAYS CHANNEL 0
                DB     0             ;ZERO FLAGS AT THE MOMENT

@DPHF           EQU    OFFSET $
                DW     0000H         ;TRANSLATE TABLE (NONE FOR HARD DISK\CF CARD 1)
                DB     0,0,0H        ;SCRATCH AREA
                DB     0H            ;DOOR OPEN FLAG
                DB     0,0H          ;SCRATCH AREA
```

```
            DW      DPB1            ;DISK PARAMETER TABLE
            DW      0FFFFH,0FFFFH   ;CHECKSUM, ALLOCATION VECTOR
            DW      0FFFFH          ;DIRECTORY BCB
            DW      0FFFFH          ;DATA BCB
            DW      0FFFFH          ;HASH TABLE
            DW      HDINIT1
            DW      HDLOGIN1
            DW      HDRD
            DW      HDWRT
            DB      0               ;UNIT 0 ON THIS CONTROLLER
            DB      0               ;ALWAYS CHANNEL 0
            DB      0               ;ZERO FLAGS AT THE MOMENT

;
;             DISKDEF 0,0,63,0,2048,3996,1024,0,2
;           DPB     512,61,256,2048,1024,1,8000H

DPB0        EQU     OFFSET $        ;DISK PARAMETER BLOCK
            DW      61              ;PHYSICAL SECTORS/TRACK (Note CPM3 uses LOGICAL 128 byte sectors)
            DB      4H              ;BLOCK SHIFT
            DB      0FH             ;BLOCK MASK (15)
            DB      0H              ;EXTNT MASK
            DW      0F2FH           ;DISK SIZE - 1 (3887)
            DW      3FFH            ;DIRECTORY MAX (1023)
            DB      0FFH            ;ALLOC0
            DB      0FFH            ;ALLOC1
            DW      8000H           ;CHECK SIZE (NONE)
            DW      1H              ;OFFSET
            DB      02H             ;PHYSICAL RECORD SHIFT FACTOR
            DB      03H             ;PHYSICAL RECORD MASK
;
DPB1        EQU     OFFSET $        ;DISK PARAMETER BLOCK
            DW      61              ;SECTORS PER TRACK (244/4=61)
            DB      4H              ;BLOCK SHIFT
            DB      0FH             ;BLOCK MASK
            DB      0H              ;EXTNT MASK
            DW      0F2FH           ;DISK SIZE - 1
            DW      3FFH            ;DIRECTORY MAX (1023)
            DB      0FFH            ;ALLOC0
            DB      0FFH            ;ALLOC1
            DW      8000H           ;CHECK SIZE (NONE)
            DW      1H              ;OFFSET
            DB      02H             ;PHYSICAL RECORD SHIFT FACTOR
            DB      03H             ;PHYSICAL RECORD MASK


DELAYStore  DB      0H              ;For Software time delay

SAYSIGNON0          DB      '8  MEGA BYTE I D E HARD DISK ON DRIVE A',0
SAYSIGNON1          DB      '8  MEGA BYTE I D E HARD DISK ON DRIVE F',0
```

```
HD_LOGIN_ERR        DB     'Disk Drive Login Error.',0
DR_INIT_ERROR       DB     BELL,CR,LF,'Drive Initilization on Dual IDE Board Failed!',0
DRIVE_BUSY          DB     'Drive Busy (bit 7) stuck high.   Status = ',0
DRIVE_NOT_READY     DB     'Drive Ready (bit 6) stuck low.  Status = ',0
DRIVE_WR_FAULT      DB     'Drive write fault.    Status = ',0
UNKNOWN_ERROR       DB     'Unknown error in status register.   Status = ',0
SEC_NOT_FOUND       DB     'Sector not found. Error Register = ',0
BAD_BLOCK           DB     'Bad Sector ID.    Error Register = ',0
UNRECOVER_ERR       DB     'Uncorrectable data error.  Error Register = ',0
INVALID_CMD         DB     'Invalid Command. Error Register = ',0
TRK0_ERR            DB     'Track Zero not found. Error Register = ',0
UNKNOWN_ERROR1      DB     'Unknown Error. Error Register = ',0

       END
```