

```

;
; Test Program to interact with the CPM3 type BIOS for the S100Computers IDE intreface 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 accomidate two CF cards (Master/Slave) & better menu options
;                               ;note I still have more work to do with this but what is here seem OK.
;      V2.1   2/5/2011     ;Menu driven, and added code to copy & restore disk data from another
;                               ;"backup" partition on disk
;      V2.2   2/13/2011    ;Added Sec++ & Sec--
;      V2.3   2/15/2011    ;Re-did drive initilization (Pulse CF cards twice on reset line)
;      V2.4   2/16/2011    ;Correct small error for end of drive track check
;      V2.5   3/14/2011    ;Added BOOT CPM option, cleaned up some areas.
;      V2.6   3/15/2011    ;Re-dis reset line pulse. Correct CPM boot to Track 0 sector 1

;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 (8255)
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 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
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)

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 actualy 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"
BUFFER$ORG EQU 3000H ;<----- Will place all sector data here

CPM$BOOT$COUNT EQU 12 ;Allow up to 12 CPM sectors for CPMLDR
CPMLDR$ADDRESS EQU BUFFER$ORG ;We cannot load the CPMLDR at 100H in RAM since this is where this program
resides

RDCON EQU 1 ;For CP/M I/O
WRCON EQU 2
RESET$DISK EQU 0DH ;Reset all CPM disks
PRINT EQU 9
CONST EQU 11 ;CONSOLE STAT
BDOS EQU 5

FALSE EQU 0
TRUE EQU NOT FALSE

CPM EQU TRUE ; TRUE if output via CPM, FALSE if direct to hardware
DEBUG EQU TRUE
CPM$TRANSLATE EQU TRUE ;Translate Trk,Sec,Head to CPM TRACK# & SEC# on display

;-----
ORG 100H ;<--- For CPM

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

;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:

```

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

```

ABORT:

```

IF CPM
MVI C,RESET$DISK ;Reset All disks in CPM
CALL BDOS
JMP 0H ;Jump to CP/M cold start address
ELSE
JMP 0F000H ;Else jump to monitor
ENDIF

```

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

; print the drive's serial number
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 ;Set DMA address to 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
LDA     @DisplayFlag    ;Do we have detail sector data display flag on or off
ORA     A                ;NZ = on (Initially 0FFH so detailed sector display on)
JNZ     Display1        ;List command options (Turn display option to on)
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 UC character Input (Note, no fancy checking)
CPI     ESC              ;Abort if ESC
JZ      ABORT
CALL    upper
CALL    ZCRLF

SBI     '@'              ;Adjust to 0,1AH

ADD     A                ;X2
LXI     H,TBL            ;Get menu selection
ADD     L
MOV     L,A
MOV     A,M
INX    HL
MOV     H,M
MOV     L,A              ;Jump to table pointer
PCHL

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
        STA    @DisplayFlag ;flip it
        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,CPM$BOOT$COUNT ;Count of CPMLDR sectors (12)
        STA    @SECTOR$COUNT
        LXI    H,CPMLDR$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    @SECTOR$COUNT

```

```

DCR    A
STA    @SECTOR$COUNT
JZ     LOAD$DONE

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

LOAD$DONE:
MVI    E,REGstatus   ;Check the R/W status when done
CALL   IDErd8D
BIT    0,D
JNZ    CPMLoadErr    ;Z if no errors
LXI    H,CPMLDR$ADDRESS
MOV    A,M
CPI    31H           ;EXPECT TO HAVE 31H @80H IE. LD SP,80H
JNZ    CPMLoadErr1   ;Z if no errors

LXI    D,MOVE$REQUEST ;Ask if we can move data to 100H overwriting this program
CALL   PSTRING
CALL   ZCI
CALL   upper
CPI    'Y'
JNZ    MAINLOOP

LXI    H,CPM$MOVE$CODE      ;Need to put memory move code out of the way.
LXI    D,0H
LXI    B,(CPM$MOVE$CODE$END-CPM$MOVE$CODE)
LDIR
JMP    0H                  ;Now jump here where the code will move the CPMLDR (@3000H) to 100H

CPMLoadErr1:
LXI    D,CPM$ERROR1       ;Drive data error
CALL   PSTRING
JMP    MAINLOOP

CPMLoadErr:
LXI    D,CPM$ERROR        ;Drive Read Error
CALL   PSTRING
JMP    MAINLOOP

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

STA    @SECTOR$COUNT     ;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    @SECTOR$COUNT
DCR    A
STA    @SECTOR$COUNT
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
INX    H                  ;Bump to next track

```

```

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: ;Write N sectors
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 @SECTOR$COUNT ;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 @SECTOR$COUNT
DCR A
STA @SECTOR$COUNT
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
        LHL    @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
        LHL    @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 -----

driveid:CALL   IDEwaitnotbusy ;Do the IDentify drive command, and return with the buffer
                ;filled with info about the drive
        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

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]
INX    H          ;Text is low byte high byte format
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                    ;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
                                ;All OK if 01000000
    PUSH   PSW              ;save for return below
    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$READY ;Drive Not Ready (bit 6) stuck low.  Status =
    CALL   PSTRING
    JMP    DONEERR
NOT6:  ANI    20H
    JNZ    NOT5
    LXI    D,DRIVE$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                   ;Set Carry flag
        RET

ENDIF

;-----
; Print a 16 bit number in RAM located @ [HL]
; (Note Special Low Byte First. Used only for Drive ID)

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:
        LXI    D,ENTER$SECL ;get CPM style Track# & Sector# data and convert to LBA format
        CALL   PSTRING      ;Enter sector number
        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
CPI    'z'+1            ;must be >= lowercase a
RNC                                         ; else go back...
CPI    'a'-'A'          ;must be <= lowercase z
RNC                                         ; else go back...
SUI    'a'-'A'          ;subtract lowercase bias
RET

ASBIN: SUI    30H
CPI    0AH
RM
SUI    07H
RET

HEXDUMP:
PUSH   PSW              ;Print a hexdump of the data in the 512 byte buffer @[HL]
PUSH   B                ;Save everything

```

```

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 @BYTE$COUNT

SF172: CALL ZCRLF
LHLD @BYTE$COUNT
MOV A,H
CALL PHEX ;Print byte count in sector
MOV A,L
CALL PHEX
PUSH D
LXI D,16
DAD D
POP D
SHLD @BYTE$COUNT ;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:
    MVI    A,READcfg8255    ;Initilze the 8255 and drive then do a hard reset on the drive,
    OUT    IDEportCtrl     ;Config 8255 chip (10010010B), read mode on return
                                ;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
                                ;I have a (negative)pulse of 2.7uSec. (10Mz Z80, two IO wait states).
                                ;Which seem to work for the 5 different CF cards I have.
    MVI    B,20H

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$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 3 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 very 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
;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

```



```

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 Robbins 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,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

MOV A,E           ;<-- Ken Robbins suggestion, raise WR line
OUT IDEportC     ;deassert RD pin

XRA    A
OUT     IDEportC   ;Deselect all lines including WR line

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

CPM$MOVE$CODE     ;This code is written to reside at 0H. Will be relocated by this
LXI     H,BUFFER   ;this program to move the boot CPMLDR to 100H in RAM (Overwriting this program)
LXI     D,100H
LXI     B,(12*512)
LDIR
JMP     100H
CPM$MOVE$CODE$END:

SIGN$ON:          DB     CR,LF,'IDE Disk Drive Test Program 3/15/2011 (V2.6) '
                  DB     '(Using CPM3 BIOS Routines)',CR,LF
                  DB     'CPM Track,Sectors --> LBA mode',LF,CR,'$'
INIT$ERROR:       DB     'Initilizing Drive Error.',CR,LF,'$'
ID$ERROR:         DB     'Error obtaining Drive ID.',CR,LF,'$'
INIT$DR$OK:       DB     'Drive Initilized 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 (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
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 (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
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$RW$ERROR 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 = $'
READ$ID$ERROR DB      'Error setting up to read Drive ID',CR,LF,'$'
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:- $'
msgErr       DB      CR,LF,'Sorry, that was not a valid menu option!$'
FormatDone   DB      CR,LF,'Disk Format Complete.',CR,LF,'$'
backupDone   DB      CR,LF,'Disk partition copy complete.',CR,LF,'$'
CopyMsg      DB      CR,LF,'Copy disk partition to a second area on disk (CF card).'
DB      CR,LF,'>>> This assumes that tracks greater than MAXTRK '
DB      '(for CPM, 0FFH) are unused <<<'
DB      CR,LF,'>>> on this disk. Be sure you have nothing in this '
DB      '"Backup partition area". <<<'
DB      CR,LF,BELL,'Warning: This will change data in the partition area, '
DB      'are you sure? (Y/N)...$ '

AtEnd        DB      CR,LF,'At end of disk partition!',CR,LF,'$'
RBackup$MSG  DB      'Reading track: $'
WBackup$MSG  DB      'H. Writing track: $'
H$Msg        DB      'H$'

RestoreMsg   DB      CR,LF,'Restore disk with data from backup partition on disk (CF card).'
DB      CR,LF,BELL,'Warning: This will change data on disk, '
DB      'are you sure? (Y/N)...$ '
DB      CR,LF,'Restore of disk data from backup partition complete.',CR,LF,'$'

RestoreDone  DB      CR,LF,'Sector value out of range.',CR,LF,'$'
RANGE$MSG    DB      CR,LF,'Error reading CPMLDR.',CR,LF,'$'
CPM$ERROR    DB      CR,LF,'Data error reading CPMLDR. (The first byte loaded was not 31H).',CR,LF,'$'
CPM$ERROR1   DB      CR,LF,'The CPMLDR image is now at 3000H in RAM. '
MOVE$REQUEST DB      'To boot CPM you will have to'
DB      CR,LF,'overwrite this program at 100H. Do you wish to do so (Y/N)...$'

```

```
; ----- RAM usage -----
```

```

RAMAREA      DB      '          RAM STORE AREA ----->'          ;useful for debugging
@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
@SEC1       DW      0H          ;For disk partition copy
@TRK1       DW      0H
@SEC2       DW      0H
@TRK2       DW      0H
@StartLineHex DW      0H
@StartLineASCII DW      0H      0H
@BYTE$COUNT DW      0H
@SECTOR$COUNT DW      0H
@DELAYStore DB      0H
;
          DS      40H
STACK      DW      0H
;
IDbuffer   DB      '          Start of ID buffer-->'
          DS      512
          DB      '<--End of ID buffer          '
;
          ORG      BUFFER$ORG          ;<--- In case we wish to use ZSID etc.
BUFFER:    DB      76H          ;put a Z80 HALT instruction here in case we
          ;jump to a sector in error
          DB      '<--Start buffer area'          ;a 512 byte buffer
          DS      476
          DB      'End of buffer-->'
;
;END

```