```
    Page 60,132 ;SI     SCCSID = @(#)msboot.asm     1.1 85/05/13
TITLE BOOT  SECTOR 1 OF TRACK 0 - BOOT LOADER

;   Rev 1.0 ChrisP, AaronR and others.  2.0 format boot
;
;   Rev 3.0 MarkZ   PC/AT enhancements
;           2.50 in label
;   Rev 3.1 MarkZ   3.1 in label due to vagaries of SYSing to IBM drive D's
;           This resulted in the BPB being off by 1.  So we now trust
;           2.0 and 3.1 boot sectors and disbelieve 3.0.
;
;   Rev 3.2 LeeAc   Modify layout of extended BPB for >32M support
;           Move PHYDRV to 3rd byte from end of sector
;           so that it won't have to be moved again
;           FORMAT and SYS count on PHYDRV being in a known location
;
;   Rev. 3.3 D.C. L. Changed Sec 9 EOT field from 15 to 18. May 29, 1986.
;
;   Rev 3.31 MarkT  The COUNT value has a bogus check (JBE????) to determine
;           if we've loaded in all the sectors of IBMBIO. This will
;           cause too big of a load if the sectors per track is high
;           enough, causing either a stack overflow or the boot code
;           to be overwritten.
;
;   Rev 4.00 J. K.  For DOS 4.00 Modified to handle the extended BPB, and
;           32 bit sector number calculation to enable the primary
;           partition be started beyond 32 MB boundary.
;
;
; The ROM in the IBM PC starts the boot process by performing a hardware
; initialization and a verification of all external devices.  If all goes
; well, it will then load from the boot drive the sector from track 0, head 0,
; sector 1.  This sector is placed at physical address 07C00h.  The initial
; registers are set up as follows:  CS=DS=ES=SS=0.  IP=7C00h, SP=0400H.
;
; The code in this sector is responsible for locating the MSDOS device drivers
; (IBMBIO) and for placing the directory sector with this information at
; physical address 00500h.  After loading in this sector, it reads in the
; entirety of the BIOS at BIOSEG:0 and does a long jump to that point.
;
; If no BIOS/DOS pair is found an error message is displayed and the user is
; prompted to reinsert another disk.  If there is a disk error during the
; process, a message is displayed and things are halted.
;
; At the beginning of the boot sector, there is a table which describes the
; MSDOS structure of the media.  This is equivalent to the BPB with some
; additional information describing the physical layout of the driver (heads,
; tracks, sectors)
;
;==============================================================================
;REVISION HISTORY:
;AN000 - New for DOS Version 4.00 - J.K.
;AC000 - Changed for DOS Version 4.00 - J.K.
;AN00x - PTM number for DOS Version 4.00 - J.K.
;==============================================================================
;AN001; d52 Make the fixed positioned variable "CURHD" to be local.  7/6/87 J.K.
;AN002; d48 Change head settle at boot time.                 7/7/87 J.K.
;AN003; P1820 New message SKL file                  10/20/87 J.K.
;AN004; D304 New structrue of Boot record for OS2.         11/09/87 J.K.
;==============================================================================

ORIGIN      EQU 7C00H           ; Origin of bootstrap LOADER
BIOSEG      EQU 70H         ; destingation segment of BIOS
BioOff      EQU 700H            ; offset of bios
cbSec       EQU 512
cbDirEnt    EQU 32
DirOff      EQU 500h
IBMLOADSIZE equ 3           ;J.K. Size of IBMLOAD module in sectors
ROM_DISKRD  equ 2
include version.inc

;
; Define the destination segment of the BIOS, including the initialization
; label
;
SEGBIOS SEGMENT AT BIOSEG
BIOS    LABEL   BYTE
SEGBIOS ENDS

CODE    SEGMENT
    ASSUME CS:CODE,DS:NOTHING,ES:NOTHING,SS:NOTHING

;    ORG     DirOff + 1Ch
;BiosFS  LABEL   WORD

    ORG ORIGIN

DSKADR  =   1EH*4           ;POINTER TO DRIVE PARAMETERS

Public $START
$START:
    JMP START
;-----------------------------------------------------------
;
;   THE FOLLOWING DATA CONFIGURES THE BOOT PROGRAM
;   FOR ANY TYPE OF DRIVE OR HARDFILE
;
;J.K. Extened_BPB

if ibmcopyright
      DB      "IBM  "
else
      DB      "MSDOS"
endif
      DB      "4.0"                 ;AN005;
ByteSec   DW      cbSec         ; SIZE OF A PHYSICAL SECTOR
      DB      8          ; SECTORS PER ALLOCATION UNIT
cSecRes   DW      1          ; NUMBER OF RESERVED SECTORS
cFat      DB      2          ; NUMBER OF FATS
DirNum    DW      512            ; NUMBER OF DIREC ENTRIES
cTotSec   DW      4*17*305-1         ; NUMBER OF SECTORS - NUMBER OF HIDDEN SECTORS
                    ;  (0 when 32 bit sector number)
MEDIA     DB      0F8H           ; MEDIA BYTE
cSecFat   DW      8          ; NUMBER OF FAT SECTORS
SECLIM    DW      17             ; SECTORS PER TRACK
HDLIM     DW      4          ; NUMBER OF SURFACES
Ext_cSecHid label dword
cSecHid_L DW      1          ;AN000; NUMBER OF HIDDEN SECTORS
cSecHid_H dw      0          ;AN000; high order word of Hiden Sectors
Ext_cTotSec label dword
ctotsec_L dw      0          ;AN000; 32 bit version of NUMBER OF SECTORS
ctotsec_H dw      0          ;AN000; (when 16 bit version is zero)
;
Phydrv    db     80h             ;AN004;
Curhd     db     0h              ;AN004; Current Head
Ext_Boot_Sig    db    41         ;AN000;
Boot_Serial dd    0          ;AN000;
Boot_Vol_Label  db    'NO NAME    '      ;AN000;
Boot_System_id  db    'FAT12   '         ;AN000;

;J.K. Danger!!! If not 32 bit sector number calculation, FORMAT should
;set the value of cSecHid_h and Ext_cTotSec to 0 !!!
;

;
Public UDATA
UDATA   LABEL   byte
Sec9      equ   byte ptr UDATA+0    ;11 byte diskette parm. table
BIOS$_L   EQU   WORD PTR UDATA+11
BIOS$_H   equ   word ptr UDATA+13   ;AN000;
CURTRK    EQU   WORD PTR UDATA+15
CURSEC    EQU   BYTE PTR UDATA+17
DIR$_L    EQU   WORD PTR UDATA+18
Dir$_H    equ   word ptr UDATA+20   ;AN000;
START:
```

```
;
; First thing is to reset the stack to a better and more known place.  The ROM
; may change, but we'd like to get the stack in the correct place.
;
    CLI                 ;Stop interrupts till stack ok
    XOR AX,AX
    MOV SS,AX               ;Work in stack just below this routine
    ASSUME  SS:CODE
    MOV SP,ORIGIN
    PUSH    SS
    POP ES
    ASSUME  ES:CODE
;
; We copy the disk parameter table into a local area.  We scan the table above
; for non-zero parameters.  Any we see get changed to their non-zero values.
;
;J.K. We copy the disk parameter table into a local area (overlayed into the
;code), and set the head settle time to 1, and End of Track to SECLIM given
;by FORMAT.

    MOV BX,DSKADR
    LDS SI,DWORD PTR SS:[BX]    ; get address of disk table
    PUSH    DS          ; save original vector for possible
    PUSH    SI          ; restore
    PUSH    SS
    PUSH    BX
    MOV DI,offset Sec9
    MOV CX,11
    CLD
if  $ le BIOS$_L
    %OUT Don't destroy unexcuted code yet!!!
endif
    repz    movsb           ;AN000;
    push    es          ;AN000;
    pop ds          ;AN000; DS = ES = code = 0.
    assume  ds:code         ;AN000;
;    mov     byte ptr [di-2], 1  ;AN000; Head settle time
;J.K. Change the head settle to 15 ms will slow the boot time quite a bit!!!
    mov byte ptr [di-2], 0fh    ;AN002; Head settle time
    mov cx, SECLIM      ;AN004;
    mov byte ptr [di-7], cl ;AN000; End of Track
;
; Place in new disk parameter table vector.
;
    MOV [BX+2],AX
    MOV [BX],offset SEC9
;
; We may now turn interrupts back on.  Before this, there is a small window
; when a reboot command may come in when the disk parameter table is garbage
;
    STI                 ;Interrupts OK now
;
; Reset the disk system just in case any thing funny has happened.
;
    INT 13H         ;Reset the system
;    JC  RERROR
    jc  CKErr           ;AN000;
;
; The system is now prepared for us to begin reading.  First, determine
; logical sector numbers of the start of the directory and the start of the
; data area.

    xor ax,ax           ;AN000;
    cmp cTotSec,ax      ;AN000; 32 bit calculation?
    je  Dir_Cont        ;AN000;
    mov cx,cTotSec      ;AN000;
    mov cTotSec_L,cx        ;AN000; cTotSec_L,cTotSec_H will be used for calculation
Dir_Cont:               ;AN000;
    MOV AL,cFat         ;Determine sector dir starts on
    MUL cSecFat         ;DX;AX
    ADD AX,cSecHid_L
    adc DX,cSecHid_H        ;AN000;
    ADD AX,cSecRes
    ADC DX,0
    MOV [DIR$_L],AX     ; DX;AX = cFat*cSecFat + cSecRes + cSecHid
    mov [DIR$_H],DX     ;AN000;
    MOV [BIOS$_L],AX
    mov [BIOS$_H],DX        ;AN000;
;
; Take into account size of directory (only know number of directory entries)
;
    MOV AX,cbDirEnt     ; bytes per directory entry
    MUL DirNum          ; convert to bytes in directory
    MOV BX,ByteSec      ; add in sector size
    ADD AX,BX
    DEC AX          ; decrement so that we round up
    DIV BX          ; convert to sector number
    ADD [BIOS$_L],AX        ; Start sector # of Data area
    adc [BIOS$_H],0     ;AN000;

;
; We load in the first directory sector and examine it to make sure the the
; BIOS and DOS are the first two directory entries.  If they are not found,
; the user is prompted to insert a new disk.  The directory sector is loaded
; into 00500h
;
    MOV BX,DirOff       ; sector to go in at 00500h
    mov dx,[DIR$_H]     ;AN000;
    MOV AX,[DIR$_L]     ; logical sector of directory
    CALL    DODIV           ; convert to sector, track, head
    jc  CKErr           ;AN000; Overflow? BPB must be wrong!!
;    MOV     AX,0201H         ; disk read 1 sector
    mov al, 1           ;AN000; disk read 1 sector
    CALL    DOCALL          ; do the disk read
    JB  CKERR           ; if errors try to recover
;
; Now we scan for the presence of IBMBIO  COM and IBMDOS  COM.  Check the
; first directory entry.
;
    MOV DI,BX
    MOV CX,11
    MOV SI,OFFSET BIO       ; point to "ibmbio  com"
    REPZ    CMPSB           ; see if the same
    JNZ CKERR           ; if not there advise the user
;
; Found the BIOS.  Check the second directory entry.
;
    LEA DI,[BX+20h]
    MOV SI,OFFSET DOS       ; point to "ibmdos  com"
    MOV CX,11
    REPZ    CMPSB
    JZ  DoLoad

;
; There has been some recoverable error.  Display a message and wait for a
; keystroke.
;
CKERR:  MOV SI,OFFSET SYSMSG    ; point to no system message
ErrOut: CALL    WRITE           ; and write on the screen
    XOR AH,AH           ; wait for response
    INT 16H         ; get character from keyboard
    POP SI          ; reset disk parameter table back to
    POP DS          ; rom
    POP [SI]
    POP [SI+2]
    INT 19h         ; Continue in loop till good disk

Load_Failure:
    pop ax          ;adjust the stack
    pop ax
    pop ax
    jmp short Ckerr     ;display message and reboot.

;J.K. We don't have the following error message any more!!!
;J.K. Sysmsg is fine.  This will save space by eliminating DMSSG message.
;RERROR: MOV     SI,OFFSET DMSSG     ; DISK ERROR MESSAGE
```

```
;    JMP     ErrOut

;
; We now begin to load the BIOS in.  Compute the number of sectors needed.
; J.K. All we have to do is just read in sectors contiguously IBMLOADSIZE
; J.K. times.  We here assume that IBMLOAD module is contiguous.  Currently
; J.K. we estimate that IBMLOAD module will not be more than 3 sectors.

DoLoad:
    mov BX,BioOff       ;offset of ibmbio(IBMLOAD) to be loaded.
    mov CX,IBMLOADSIZE      ;# of sectors to read.
    mov AX, [BIOS$_L]       ;Sector number to read.
    mov DX, [BIOS$_H]       ;AN000;
Do_While:               ;AN000;
    push    AX          ;AN000;
    push    DX          ;AN000;
    push    CX          ;AN000;
    call    DODIV           ;AN000; DX;AX = sector number.
    jc  Load_Failure        ;AN000; Adjust stack. Show error message
    mov al, 1           ;AN000; Read 1 sector at a time.
                    ;This is to handle a case of media
                    ;when the first sector of IBMLOAD is the
                    ;the last sector in a track.
    call    DOCALL          ;AN000; Read the sector.
    pop CX          ;AN000;
    pop DX          ;AN000;
    pop AX          ;AN000;
    jc  CkErr           ;AN000; Read error?
    add AX,1            ;AN000; Next sector number.
    adc DX,0            ;AN000;
    add BX,ByteSec      ;AN000; adjust buffer address.
    loop    Do_While        ;AN000;


;    MOV     AX,BiosFS       ; get file size
;    XOR     DX,DX           ; presume < 64K
;    DIV     ByteSec         ; convert to sectors
;    INC     AL          ; reading in one more can't hurt
;    MOV     COUNT,AL        ; Store running count
;    MOV     AX,BIOS$        ; get logical sector of beginning of BIOS
;    MOV     BIOSAV,AX       ; store away for real bios later
;    MOV     BX,BioOff       ; Load address from BIOSSEG
;
; Main read-in loop.
;   ES:BX points to area to read.
;   Count is the number of sectors remaining.
;   BIOS$ is the next logical sector number to read
;
;LOOPRD:
;    MOV     AX,BIOS$        ; Starting sector
;    CALL    DODIV
;
; CurHD is the head for this next disk request
; CurTrk is the track for this next request
; CurSec is the beginning sector number for this request
;
; Compute the number of sectors that we may be able to read in a single ROM
; request.
;
;    MOV     AX,SECLIM
;    SUB     AL,CURSEC
;    INC     AX
;
; AX is the number of sectors that we may read.
;

;
;New code for Rev 3.31
;*****************************************************************************

;    CMP     COUNT,AL      ;Is sectors we can read more than we need?
;    JAE     GOT_SECTORS       ;No, it is okay
;    MOV     AL,COUNT      ;Yes, only read in what is left
```

```
;GOT_SECTORS:

;*******************************************************************************
;End of change
;


;    PUSH    AX
;    CALL    DOCALL
;    POP     AX
;    JB  RERROR          ; If errors report and go to ROM BASIC
;    SUB     COUNT,AL        ; Are we finished?
;
;Old code replaced by Rev 3.3
;*********************************************************************
;   JBE DISKOK          ; Yes -- transfer control to the DOS
;*********************************************************************
;New code for Rev 3.3
;

;    JZ  DISKOK          ; Yes -- transfer control to the DOS

;*********************************************************************
;End of change
;
;    ADD     BIOS$,AX        ; increment logical sector position
;    MUL     ByteSec         ; determine next offset for read
;    ADD     BX,AX           ; (BX)=(BX)+(SI)*(Bytes per sector)
;    JMP     LOOPRD          ; Get next track
;
; IBMINIT requires the following input conditions:
;
;   DL = INT 13 drive number we booted from
;   CH = media byte
;J.K.I1. BX was the First data sector on disk (0-based)
;J.K.I1. IBMBIO init routine should check if the boot record is the
;J.K.I1. extended one by looking at the extended_boot_signature.
;J.K.I1. If it is, then should us AX;BX for the starting data sector number.

DISKOK:
    MOV CH,Media
    MOV DL,PhyDrv
    MOV bx,[BIOS$_L]        ;AN000; J.K.I1.Get bios sector in bx
    mov ax,[BIOS$_H]        ;AN000; J.K.I1.
    JMP FAR PTR BIOS        ;CRANK UP THE DOS

WRITE:  LODSB               ;GET NEXT CHARACTER
    OR  AL,AL           ;clear the high bit
    JZ  ENDWR           ;ERROR MESSAGE UP, JUMP TO BASIC
    MOV AH,14           ;WILL WRITE CHARACTER & ATTRIBUTE
    MOV BX,7            ;ATTRIBUTE
    INT 10H         ;PRINT THE CHARACTER
    JMP WRITE

; convert a logical sector into Track/sector/head.  AX has the logical
; sector number
; J.K. DX;AX has the sector number. Because of not enough space, we are
; going to use Simple 32 bit division here.
; Carry set if DX;AX is too big to handle.
;

DODIV:
    cmp dx,Seclim       ;AN000; To prevent overflow!!!
    jae DivOverFlow     ;AN000; Compare high word with the divisor.
    DIV SECLIM          ;AX = Total tracks, DX = sector number
    INC DL          ;Since we assume SecLim < 255 (a byte), DH =0.
                    ;Cursec is 1-based.
    MOV CURSEC, DL      ;save it
    XOR DX,DX
    DIV HDLIM
    MOV CURHD,DL        ;Also, Hdlim < 255.
    MOV CURTRK,AX
    clc             ;AN000;
    ret             ;AN000;
DivOverFlow:                ;AN000;
    stc             ;AN000;
EndWR:
    ret

;
;J.K.We don't have space for the following full 32 bit division.
; convert a logical sector into Track/sector/head.  AX has the logical
; sector number
; J.K. DX;AX has the sector number.
;DODIV:
;   push    ax
;   mov ax,dx
;    xor     dx,dx
;    div     SecLim
;    mov     Temp_H,ax
;    pop     ax
;    div     SecLim          ;J.K.Temp_H;AX = total tracks, DX=sector
;    INC     DL          ;Since we assume SecLim < 255 (a byte), DH =0.
;                       ;Cursec is 1-based.
;    MOV     CURSEC, DL      ;save it
;    push    ax
;    mov     ax,Temp_H
;    XOR     DX,DX
;    DIV     HDLIM
;    mov     Temp_H,ax
;    pop     ax
;    div     HdLim           ;J.K.Temp_H;AX=total cyliners,DX=head
;    MOV     CURHD,DL        ;Also, Hdlim < 255.
;    cmp     Temp_H,0
;    ja  TooBigToHandle
;    cmp     ax, 1024
;    ja  TooBigToHandle
;    MOV     CURTRK,AX
;ENDWR:  RET
;TooBigToHandle:
;    stc
;    ret

;
; Issue one read request.  ES:BX have the transfer address, AL is the number
; of sectors.
;
DOCALL: MOV AH,ROM_DISKRD       ;AC000;=2
    MOV DX,CURTRK
    MOV CL,6
    SHL DH,CL
    OR  DH,CURSEC
    MOV CX,DX
    XCHG    CH,CL
    MOV DL, PHYDRV
    mov dh, curhd
    INT 13H
    RET

;    include ibmbtmes.inc
    include boot.cl1            ;AN003;


    IF IBMCOPYRIGHT
BIO DB  "IBMBIO  COM"
DOS DB  "IBMDOS  COM"
    ELSE
BIO DB  "IO      SYS"
DOS DB  "MSDOS   SYS"
    ENDIF

Free    EQU (cbSec - 4) - ($-$start)        ;AC000;
;Free    EQU (cbSec - 5) - ($-$start)
if Free LT 0
    %out FATAL PROBLEM:boot sector is too large
```

```
endif

	org origin + (cbSec - 2)	;AN004;
;	org	origin + (cbSec - 5)

;Warning!! Do not change the position of following unless
;Warning!! you change BOOTFORM.INC (in COMMON subdirectory) file.
;Format should set this EOT value for IBMBOOT.
;FEOT	db	12h		;AN000; set by FORMAT. AN004;Use SecLim in BPB instead.
; FORMAT and SYS count on CURHD,PHYDRV being right here
;J.K. CURHD has been deleted since it is not being used by anybody.
;CURHD	DB	?		;AN001;Unitialized (J.K. Maybe don't need this).
;PHYDRV	db	0		;AN000;moved into the header part.
; Boot sector signature
	db	55h,0aah

CODE	ENDS
	END
SUB
```