```
                                page    40,132
                        ;       T0DOSMBR/EBR is part of T0-DOS. Copyright (C) 2010 Ton Daas
                        ;       T0-DOS is free software: you can redistribute it and/or modify
                        ;       it under the terms of the GNU General Public License as published by
                        ;       the Free Software Foundation, either version 3 of the License,
                        ;       or any later version.
                        ;
                        ;       T0-DOS is distributed in the hope that it will be useful,
                        ;       but WITHOUT ANY WARRANTY; without even the implied warranty of
                        ;       MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
                        ;       See the GNU General Public License for more details.
                        ;
                        ;       You should have received a copy of the GNU General Public License
                        ;       along with this program.  If not, see <https://www.gnu.org/licenses/>.
                        ;_____________________________________________________________________________
                        ; Track 0 DOS (T0-DOS) MBR / DOS 1 Boot Sector at cyl=0, head=0, sect=1
                        ; Part of T0-DOS (C) 2010 Ton Daas
                        ; Upon execution reg values are: CS=0000h, IP=7C00h, DL=drive
                        ;-----------------------------------------------------------------------------
0000                    t0dos   segment

0200                            org     200h
= 0200                  FAT     equ     $

= 00EA                  jmpf    equ     0EAh    ;far jump opcode
03F8                            org     3F8h
= 03F8                  retmbr  equ     $       ;= jmp  far ptr ?,? ;address of T0DOS back to MBR code
                                                ;= jmp short to T0DOS init entrypoint

                        ; Address in segment 0 where ROM-BIOS loads and executes a boot record
7C00                            org     7C00h
= 7C00                  loadadr equ     $

                        ; Area to load IBMBIO.COM and IBMDOS.COM or to copy mbr temporarily into
```

```
0600                            org     600h
= 0600                  dosbios equ     $       ;For relocation of mbr this is offset in segment 0

                        ;-------following code is executed before relocation-----------------------------
                        ; Setup stack and relocate code to make room for chain-boot
0600  33 C9                     xor     cx,cx
0602  8E D9                     mov     ds,cx
0604  BE 7C00 R                 mov     si,offset loadadr       ;point ds:si to own code
0607  8E C1                     mov     es,cx   ;set destination segment
0609  BF 0600 R                 mov     di,offset dosbios ;and offset, where later DOS would be loaded
060C  FC                        cld
060D  FA                        cli     ;prevent interrupts during stack change and string move
060E  8E D1                     mov     ss,cx   ;set stack segment
0610  8B E6                     mov     sp,si   ;set stack to before this code.
0612  B5 01                     mov     ch,1    ;256 words
0614  F3/ A5                    rep movsw       ;move this code from address 7C00h to 600h cx=0 on exit
0616  FB                        sti             ;enable interrupts after string move is done
                                assume  cs:t0dos,ds:t0dos
0617  E9 904B R                 jmp     cont-loadadr+dosbios   ;near jump will do nicely

                        ;-------following code is used and executed after relocation!--------------------
= 0080                  bootfl  equ     80h     ;bootflag

                        ; Partition type list
= 0001                  fat12   equ     01h     ;11h if hidden, <32Mb
= 0004                  fat16   equ     04h     ;14h if hidden, >32Mb <500Mb
= 0005                  extend  equ     05h     ;15h if hidden
= 0006                  fat16b  equ     06h     ;16h if hidden, >32Mb <2Gb
= 0007                  ntfs    equ     07h     ;17h if hidden, also OS/2 HPFS
= 000B                  fat32   equ     0Bh     ;1Bh if hidden, <2Gb
= 000C                  f32lba  equ     0Ch     ;1Ch if hidden, C/H/S=unused, relative sector=LBA
= 000E                  f16lba  equ     0Eh     ;1Eh if hidden, C/H/S=unused, relative sector=LBA
= 000F                  extlba  equ     0Fh     ;1Fh if hidden, C/H/S=unused, relative sector=LBA

= 0010                  hidden  equ     10h     ;single bit is set in DOS partition types
```

```
                                ; Describe types that can apear hidden and/or require CHS boot
061A  05                        chstype db      extend          ;types that should boot with CHS info
061B  01 04 06 0B               dostype db      fat12,fat16,fat16b,fat32 ;chs types that can appear hidden
= 0005                          chslen  equ     $-chstype
061F  0E 0C                             db      f16lba,f32lba   ;lba types that can appear hidden
0621  07                                db      ntfs    ;ntfs supports both and can appear hidden
= 0007                          typelen equ     $-dostype

                                ; Routine to write text to screen. Exits with ds:si pointing at terminating 0
                                ; entrypoint is tty (or wrtty if AL already has first character to write)
                                ; on entry ds:si points to message to write (null terminates routine)
0622                            wrtty   proc    near
0622  B3 07                             mov     bl,7    ;white
0624  B4 0E                             mov     ah,0Eh  ;write teletype to active page
0626  CD 10                             int     10h     ;AL=character, BL=foreground color
0628  AC                        tty:    lodsb
0629  3C 00                             cmp     al,0
062B  75 F5                             jne     wrtty   ;end on nul
062D  4E                                dec     si      ;set pointer back to trailing 0
062E  C3                                ret
062F                            wrtty   endp

                                ; Routine to read or write (AL mod 10) sectors to ES:[BX]
                                ; AL/10=function (2=read or 3=write). Return with CF=1 if error
062F                            int13   proc    near
062F  B4 08                             mov     ah,100h shr 5  ;set retrycount 5
0631  8B F8                     int13r: mov     di,ax   ;save
0633  D4 0A                             aam             ;AH= AL/10 and AL= AL mod 10
0635  CD 13                             int     13h
0637  73 09                             jnc     int13x
0639  B4 00                             mov     ah,0    ;reset disk system
063B  CD 13                             int     13h
063D  97                                xchg    ax,di
063E  D0 E4                             shl     ah,1
```

```
0640  73 EF                    jnc     int13r  ;try ah shift times until CF=1
0642  C3                      int13x: ret
0643                          int13   endp

                              ; Return control to PC-BIOS
0643  CD 18                   basic:  int     18h     ; call ROM BASIC or next boot device

0645  BE 077B R               inval:  mov     si,offset inv_msg
0648  E9 0776 R                       jmp     abend

                              ; Find an active partition
064B  BE 07BE R               cont:   mov     si,offset table       ;load a valid offset in case we exit to DOS
064E  8B FE                           mov     di,si
0650  B1 04                           mov     cl,4    ;loop count (assume ch=0)
0652  B8 0080                         mov     ax,bootfl       ;bootflag
0655  2A 25                   check:  sub     ah,[di] ;get and test act. (AF=0, if act CF=1 else CF=0)
0657  74 09                           jz      next    ;jump if active flag clear
0659  32 C4                           xor     al,ah   ;validate bootflag and make future flags invalid
065B  75 E8                           jnz     inval   ;if not equal, then invalid bootflag field
065D  8B F7                           mov     si,di   ;save table entry offset
065F  98                              cbw             ;make ah=0 again (al=0)
0660  38 05                   chck_a: cmp     [di],al ;(opcode=038h) effective nop (AF=0, CF=0)
                              ;patch: mov     [di],al ;(opcode=088h) clear active flag (if patched with /A)
0662  8D 7D 10                next:   lea     di,[di+16]
0665  E2 EE                           loop    check   ;not all 4 partitions done?
0667  2F                              das             ;sets ZF=1 if an active partition found (assume AF=0)
0668  75 0D                   forc_a: jnz     lt0dos  ;(075h) go and load T0DOS if no partition is active
                              ;patch: jmp     lt0dos  ;(0EBh) jmp short unconditional (if patched with /A)

                              ; Check keystroke if we want to boot DOS before active partition is loaded
066A  B4 01                           mov     ah,1    ;read keyboard status
066C  CD 16                           int     16h     ;AL=ascii, AH=scancode
066E  74 05                           jz      isetac  ;continue load, if no keystroke in buffer
0670  80 FC 81                        cmp     ah,81h  ;if it is scancode for Alt-0,
0673  74 02                           je      lt0dos  ;then load T0DOS
```

```
0675  EB 7C                isetac: jmp    short setact   ;else continue with active partition DS:[SI]

                          ; Reserve 4K at top of memory
0677  CD 12                lt0dos: int    12h     ;get memory size in KB (min 64, max 640)
0679  2D 0004                      sub    ax,4    ;4 KB below end of memory
067C  B1 06                        mov    cl,6
067E  D3 E0                        shl    ax,cl   ;convert to segment value
0680  8E C0                        mov    es,ax   ;point to segment at top of memory
                                   assume es:nothing

                          ; Load T0-DOS boot routine from disk DL, track 0, sectors 2 and 3
0682  BB 0200 R                    mov    bx,offset FAT  ;point ES:BX to address for the T0DOS BIOS code
0685  B6 00                        mov    dh,0    ;head 0, dl still has current drive
0687  B1 02                        mov     cl,2   ;cyl 0, sector 2 (first and second FAT locations)
0689  B0 16                        mov    al,22   ;read 2 sectors
068B  E8 062F R                    call   int13
068E  72 B3                        jc     basic
0690  BF 03F8 R                    mov    di,offset retmbr       ;point at return jmp far from T0DOS
0693  B0 EA                        mov    al,jmpf ;get expected opcode
0695  AE                          scasb          ;is expected code present?
0696  75 AB                        jne    basic

                          ; Set our return address and jump to T0-DOS boot routine at top of memory
0698                      t0init  proc   far
0698  B8 06A2 R                    mov    ax,offset dosret
069B  AB                          stosw          ;pass our MBR return address
069C  8C C8                       mov    ax,cs
069E  AB                          stosw          ;and our segment
069F  06                          push   es
06A0  57                          push   di      ;entrypoint for T0-DOS initialization
06A1  CB                          ret            ;go to T0 init, CS, ES=top of mem, DS=0, SI=act.part.
06A2                      t0init  endp

                          ; Entrypoint on return from DOS. DS:SI=selected table entry, AH=request
                                   assume es:t0dos
```

```
176      06A2  F6 C4 04              dosret: test    ah,04h  ;was request for a primary partition?
177      06A5  75 9C                       jnz     basic   ;if not, exit to next device or start ROM-basic
178
179                                ; Unhide partition if selected partition is hidden dostype
180      06A7  BB 07BE R                   mov     bx,offset table       ;point to partition table
181      06AA  BF 061B R                   mov     di,offset dostype ;point at list of types that can be hidden
182      06AD  B9 0007                     mov     cx,typelen
183      06B0  8A 44 04                    mov     al,[si+4]       ;get partition type
184      06B3  84 C0                       test    al,al           ;if unused entry?
185      06B5  74 09                       jz      hide            ;then hide others
186      06B7  34 10                       xor     al,hidden       ;change to unhidden type
187      06B9  F2/ AE                      repne scasb             ;check if partition is in this list
188      06BB  75 21                       jne     chkchg
189      06BD  88 44 04                    mov     [si+4],al       ;unhide selected hidden partition
190
191                                ; Hide any unhidden dos partitions
192      06C0  3B DE               hide:   cmp     bx,si
193      06C2  74 10                       je      skphid  ;if selected partition, go unhide it
194      06C4  8A 47 04                    mov     al,[bx+4]       ;get type of other partition
195      06C7  BF 061B R                   mov     di,offset dostype ;point at list of types that can be hidden
196      06CA  B1 07                       mov     cl,typelen
197      06CC  F2/ AE                      repne scasb     ;if type is not in this list, then:
198      06CE  75 04                       jne     skphid  ;already hidden or non dos, so leave type as is
199      06D0  80 77 04 10                 xor     byte ptr[bx+4],hidden  ;hide partition
200      06D4  8D 5F 10            skphid: lea     bx,[bx+16]      ;skip to next table entry
201      06D7  80 FB FE                    cmp     bl,low(table-t0dos+64) ;not all entries done?
202      06DA  72 E4                       jb      hide    ;then repeat routine
203      06DC  EB 05                       jmp     short wrtchg
204
205                                ; Check for bootflag change request
206      06DE  F6 C4 80            chkchg: test    ah,80h  ;was bootflag change requested?
207      06E1  74 10                       jz      setact  ;if not, skip write of bootsector
208      06E3  F6 04 80            wrtchg: test    byte ptr [si],bootfl   ;(0F6h) effective nop
209                                ;patch: mov     byte ptr [si],bootfl   ;(0C6h) set flag (if patched with /A)
210
```

```
                                ; Write changed partition table back to master boot record
06E6  BB 0600 R                         mov     bx,offset dosbios       ;buffer address
06E9  B6 00                             mov     dh,0    ;set head=0, drive number is still in dl
06EB  B9 0001                           mov     cx,1    ;set sector=1 and cylinder=0
06EE  B0 1F                             mov     al,31   ;write back master boot record
06F0  E8 062F R                         call    int13   ;continue even with error

                                ; Analize partition type to decide on load method
06F3  C6 04 80                  setact: mov     byte ptr [si],bootfl    ;mark partition active to loaded OS
06F6  8A 44 04                  gettyp: mov     al,[si+4]       ;get partition type
06F9  84 C0                             test    al,al   ;is it unused entry?
06FB  74 6C                             jz      noboot  ;exit, since unused partition is not bootable
06FD  BF 061A R                         mov     di,offset chstype
0700  B9 0005                           mov     cx,chslen       ;assume CH=0
0703  F2/ AE                            repne scasb     ;is active partition chs type?
0705  74 47                             je      rdchs   ;if so, then load chs methode, else:

                                ; Check for extended int 13h support if partition is LBA type, ntfs or unknown
0707  BB 55AA                           mov     bx,55AAh        ;fill with request signature
070A  B4 41                             mov     ah,41h  ;get extended int 13 support info; DL still has drive
070C  CD 13                             int     13h
070E  72 3E                             jc      rdchs   ;extension not found, so attempt clasic CHS method
0710  81 FB AA55                        cmp     bx,0AA55h       ;signature, AH=major version, DH=extension ver.
0714  75 38                             jne     rdchs   ;requested support not installed
0716  F6 C1 01                          test    cl,01h  ;bit0=1 if int-13h (AH=42h) supported
0719  74 33                             jz      rdchs   ;jump if API subset not supported

                                ; Read bootrecord with extended int 13h
071B  8B DC                             mov     bx,sp   ;get bootsector load address
071D  B9 0005                           mov     cx,5    ;set retrycount
0720  56                        retrlb: push    si      ;save si
0721  33 C0                             xor     ax,ax
                                ; Build address request packet on stack
0723  50                                push    ax      ;sector 4th word
0724  50                                push    ax      ;sector 3rd word
```

```
0725  FF 74 0A                    push    [si+10] ;sector 2nd word
0728  FF 74 08                    push    [si+8]  ;sector low word
072B  06                          push    es      ;buffer segment
072C  53                          push    bx      ;buffer offset
072D  40                          inc     ax
072E  50                          push    ax      ;number of sectors (max.(7F)
072F  B0 10                       mov     al,10h  ;packet size
0731  50                          push    ax      ;high byte reserved (=0)
0732  8B F4                       mov     si,sp   ;DS:[SI] points to request address packet
0734  B4 42                       mov     ah,42h  ;extended disk read; DL has drive number
0736  CD 13                       int     13h
0738  72 04                       jc      skp_cc  ;if CF then AH=errorcode else AH=0
                                ; Check actualy count sectors read, as C is not set for sector not found error
073A  83 7C 02 01                 cmp     word ptr [si+2],1      ;check actual count
073E  8D 64 0E                skp_cc: lea     sp,[si+14]     ;purge address request packet-1w from stack
0741  58                          pop     ax      ;restore initial AX (=0)
0742  5E                          pop     si      ;restore si
0743  73 18                       jnc     readok  ;if count < 1 then return C=1, else C=0
0745  CD 13                       int     13h
0747  E2 D7                       loop    retrlb
0749  BE 0793 R                rdfail: mov     si,offset err_msg
074C  EB 28                       jmp     short abend

                                ; Read bootrecord of active partition
074E  8B DC                    rdchs:  mov     bx,sp   ;set ES:BX to buffer address 7C00h
0750  8A 74 01                    mov     dh,[si+1]      ;set head number, DL still has drive number
0753  8B 4C 02                    mov     cx,[si+2]      ;set sector & cyl
0756  B0 15                       mov     al,21   ;read partition bootsector
0758  E8 062F R                   call    int13
075B  72 EC                       jc      rdfail
075D  81 BF 01FE AA55          readok: cmp     word ptr [bx+bootid-dosbios],0AA55h   ;is it bootable?
0763  75 04                       jne     noboot
0765  FF D3                       call    bx      ;execute partition boot record. DS:SI= part.tab. entry
0767  EB 8D                       jmp     gettyp  ;if it returns, then asume EBR passed first logical back
```

```
0769  BE 0799 R              noboot: mov     si,offset mis_msg
076C  8B FE                         mov     di,si
076E  B8 694D                       mov     ax,'iM' ;modify message text
0771  AB                            stosw
0772  B8 7373                       mov     ax,'ss'
0775  AB                            stosw
0776  E8 0628 R              abend:  call    tty
0779  EB FB                         jmp     short abend    ;loop on last 0

077B  49 6E 76 61 6C 69      inv_msg db      'Invalid partition table',0
      64 20 70 61 72 74
      69 74 69 6F 6E 20
      74 61 62 6C 65 00
0793  45 72 72 6F 72 20      err_msg db      'Error '       ;concatenated with next unpatched text
0799  6C 6F 61 64 69 6E      mis_msg db      'loading operating system' ;first chars patched with 'Miss'
      67 20 6F 70 65 72
      61 74 69 6E 67 20
      73 79 73 74 65 6D
07B1     04 [                       db      dosbios+1B5h-$ dup (0) ;should give at least one byte 0!
               00
                  ]

07B5  7B 93 99                      db      low(inv_msg-dosbios),low(err_msg-dosbios),low(mis_msg-dosbios)
07B8  00 00 00 00                   dd      0       ;NT signature (enables the OS to identify specific disk)
07BC  0000                          dw      0       ;unused (usually 0)
07BE     40 [               table   db      4*16 dup (0)   ;area for partition table
               00
                  ]

07FE  AA55                  bootid  dw      0AA55h  ;should have signature AA55h
```

```
                                page
                        ;_____________________________________________________________________________
                        ; Track 0 DOS (T0DOS) primairy EBR code (same as TBOOTMGR (C) 2021 Ton Daas).
                        ; Upon execution reg. values are: CS=0000h, IP=7C00h, DL=drive,
                        ;       DS:[SI] points at our Primairy Partition Table entry
                        ; It will replace entry DS:[SI] with first entry in EBR partition table.
                        ; First sector LBA value is converted to absolute value
                        ; and partition type converted to LBA type if this EBR is LBA type.
                        ; It then will return from call to MBR boot code with initial SI and DL values.
                        ; On return AX, DI are destroyed.

                        ; Check if we received a call from T0-DOS (or TBOOTMGR) MBR code
7C00                            org     loadadr
7C00                    ebr     proc    near
7C00  33 C0                     xor     ax,ax
7C02  81 FC 7BFE R              cmp     sp,offset loadadr-2 ;if stack holds a return address,
7C06  74 15                     je      eptcpy  ;then continue with chainboot

                        ; else write boot failure text to screen. DS may have unexpected value
7C08  8E D8                     mov     ds,ax   ;make DS zero
7C0A  BE 7C58 R                 mov     si,offset ebrmsg        ;get message offset
7C0D  B3 07                     mov     bl,7    ;white
7C0F  AC                lp_tty: lodsb
7C10  84 C0                     test    al,al   ;if end of message,
7C12  74 06                     jz      no_tty  ;then stop
7C14  B4 0E             wr_tty: mov     ah,0Eh  ;write teletype to active page
7C16  CD 10                     int     10h     ;AL=character, BL=foreground color
7C18  EB F5                     jmp     short lp_tty

                        ; Infinite loop on final 0
7C1A  4E                no_tty: dec     si
7C1B  EB F2                     jmp     short lp_tty   ;loop on last 0

                        ; Copy first ept entry into ppt table at DS:[SI] and return
7C1D  56                eptcpy: push    si      ;save SI
```

```
346      7C1E  06                        push    es      ;save ES
347      7C1F  1E                        push    ds      ;copy DS
348      7C20  07                        pop     es      ;to ES
349      7C21  8E D8                     mov     ds,ax   ;set DS to 0
350      7C23  8D 7C 01                  lea     di,[si+1]       ;skip bootflag
351      7C26  BE 7DBF R                 mov     si,offset ept+1 ;our load address + table offset +1
352      7C29  A4                        movsb           ;copy first head
353      7C2A  A5                        movsw           ;copy first cyl/sec
354      7C2B  AC                        lodsb           ;load type in AL
355      7C2C  26: 80 3D 0F              cmp     byte ptr es:[di],extlba         ;if our type is not LBA,
356      7C30  75 12                     jne     ebrnlb  ;then continue with copy
357      7C32  3C 06                     cmp     al,fat16b       ;else replace CHS types to LBA
358      7C34  75 02                     jne     ebrn16
359      7C36  B0 0E                     mov     al,f16lba
360      7C38  3C 0B             ebrn16: cmp     al,fat32
361      7C3A  75 02                     jne     ebrn32
362      7C3C  B0 0C                     mov     al,f32lba
363      7C3E  3C 05             ebrn32: cmp     al,extend
364      7C40  75 02                     jne     ebrnlb
365      7C42  B0 0F                     mov     al,extlba
366      7C44  AA                ebrnlb: stosb           ;save type
367      7C45  A4                        movsb           ;copy last head
368      7C46  A5                        movsw           ;copy last cyl/sec
369      7C47  AD                        lodsw           ;get first sector LBA low word
370      7C48  26: 03 05                 add     ax,es:[di]      ;add to parent LBA
371      7C4B  AB                        stosw           ;store result
372      7C4C  AD                        lodsw           ;get first sector LBA high word
373      7C4D  26: 13 05                 adc     ax,es:[di]      ;add to parent LBA high word
374      7C50  AB                        stosw           ;store result
375      7C51  A5                        movsw           ;copy sector amount low word
376      7C52  A5                        movsw           ;copy sector amount high word
377      7C53  06                        push    es      ;move ES to DS
378      7C54  1F                        pop     ds
379      7C55  07                        pop     es      ;restore ES
380      7C56  5E                        pop     si      ;restore SI
```

```
7C57  C3                              ret             ;return control to T0DOS MBR
7C58                            ebr   endp
7C58  45 78 74 65 6E 64         ebrmsg db     'Extended partition not bootable',0
      65 64 20 70 61 72
      74 69 74 69 6F 6E
      20 6E 6F 74 20 62
      6F 6F 74 61 62 6C
      65 00
7DBE                                  org     loadadr+1BEh
7DBE     40 [                   ept   db      64 dup (0)     ;extended partition table
                00
                    ]
7DFE  AA55                            dw      0AA55h  ;last word should have this signature
7E00                            t0dos ends
                                      end
```